Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский авиационный институт
(национальный исследовательский университет)»

На правах рукописи

ГИНЗБУРГ Илья Борисович

АВТОНОМНЫЕ ОТКАЗОУСТОЙЧИВЫЕ ВЕБ-ПРИЛОЖЕНИЯ ДЛЯ ГЕОИНФОРМАЦИОННЫХ СИСТЕМ С ИСПОЛЬЗОВАНИЕМ ДАННЫХ ДИСТАНЦИОННОГО ЗОНДИРОВАНИЯ ЗЕМЛИ

Специальность 05.13.01
Системный анализ, управление и обработка информации
(авиационная и ракетно-космическая техника)

ДИССЕРТАЦИЯ
на соискание ученой степени кандидата технических наук

Научный руководитель
доктор технических наук,
профессор С. Н. Падалко

Москва – 2016

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность темы исследования

В «Концепции развития российской космической системы дистанционного зондирования Земли на период до 2025 года» отдельно говорится (глава 7) о необходимости создания современного наземного комплекса приема, обработки и распространения космических данных, получаемых от космических аппаратов дистанционного зондирования Земли (ДЗЗ), как неотъемлемого компонента отечественной космической инфраструктуры. В настоящее время данный комплекс создается в виде Единой территориально-распределенной информационной системы ДЗЗ (ЕТРИС ДЗЗ). Задачей ЕТРИС ДЗЗ является объединение разрозненных систем, обеспечивающих сбор, хранение, обработку и доступ к данным ДЗЗ разных тематик, и обеспечение доступа широкого круга пользователей к этим данным, используя как ведомственные или иные линии связи, так и сеть Интернет посредством веб-приложений.

В настоящее время существующие системы мониторинга земной поверхности и геоинформационные системы (ГИС), использующие данные ДЗЗ, обеспечивают доступ к своим ресурсам на основе традиционных веб-приложений, которые недостаточно учитывают специфику веб-представления геопространственных данных. В то же время эта специфика согласуется с возможностями, предоставляемыми недавно принятым стандартом HTML5. В этой связи актуальным является исследование возможностей использования названной специфики для создания веб-приложений на базе стандарта HTML5, расширяющих применение геоданных при решении социально-экономических задач за счет использования мобильных клиентских устройств с минимизацией последствий разрыва соединений в процессе обмена информацией между клиентом и сервером ГИС.

Степень разработанности темы исследования. Согласно приведенному в работе аналитическому обзору, близкими к теме диссертации являются две группы известных результатов, которые базируются на: 1) рекомендациях стандартов веб-представлений геоданных Open Geospatial Consortium; 2) результатах в области современных веб-технологий. При этом первые из них никак не ориентированы на обеспечение отказоустойчивости и возможности автономной работы, а вторые, в первую очередь, базирующиеся на рекомендациях стандарта HTML5, хотя и имеют соответствующие универсальные механизмы, но никак не учитывают специфику веб-представления геоданных. Результаты, на получение которых ориентирована работа, а именно: обеспечение отказоустойчивого доступа к геоданным с различных клиентских устройств с помощью веб-приложений с возможностью автономной работы и резервированием вводимых пользователем данных – среди известных отсутствуют.

Целью работы является исследование возможностей использования технологий стандарта HTML5 для создания веб-приложений, обеспечивающих мониторинг земной поверхности и работу пользователей с геоданными с различных клиентских устройств, включая мобильные, в условиях:

а) сбоев соединения с сервером ГИС (отказоустойчивость);

б) полного отсутствия соединения с сервером ГИС (автономность).

Научная новизна работы состоит в том, что в ней проведено исследование, которое учитывает специфику геоданных (объем, срок актуальности, пространственное разрешение, многослойность структуры) и использует хорошо согласующиеся с этой спецификой возможности стандарта создания веб-приложений HTML5 (механизмы сохранения данных и манипулирования ими на клиентском устройстве средствами только веб-приложения).

На основе результатов исследования предложено автономное отказоустойчивое веб-приложение (АВП). Его основная идея состоит в том, что в состав клиент-серверной системы вводится программно-информационный комплекс, обладающий следующим свойством: информация, попавшая в него с клиентского устройства или с сервера, сохраняется до тех пор, пока она необходима.

Для сохранения в АВП информации используются механизмы, предоставляемые стандартом HTML5. Управление локальным сохранением и актуализацией полученных с сервера данных на стороне клиента осуществляется на основе конфигурационного файла локального хранилища, предусмотренного этим же стандартом и называемого манифестом кэша.

Объектом исследований является система обеспечения доступа к данным дистанционного зондирования Земли с произвольных типов стационарных и мобильных клиентских устройств посредством сети Интернет/Интранет.

Областью исследований является разработка на базе стандарта HTML5 метода и основанных на нем алгоритмов функционирования клиент-серверного компонента системы мониторинга земной поверхности и обеспечения доступа к геопространственным данным.

Основные задачи, решаемые в работе:

1. анализ специфических особенностей геоданных, структуры веб-представлений геоданных и их сопоставление с возможностями, предоставляемыми стандартом HTML5;

2. разработка и обоснование метода функционирования клиентского приложения нового типа, ориентированного на расширение круга пользователей системы обеспечения доступа к геоданным, базирующегося на возможностях, предоставляемых стандартом HTML5 и обеспечивающего:

а) автономную работу веб-клиента с полученными с сервера данными;

б) автоматическое аварийное резервирование вводимых пользователем данных при потере соединения с сервером;

в) поддержку произвольных типов стационарных и мобильных клиентских устройств;

3. построение и обоснование математических моделей оценки экономии трафика и ускорения загрузки страниц в АВП по сравнению с традиционным веб-приложением;

4. разработка алгоритмов функционирования клиентских и серверных компонентов АВП;

5. реализация АВП на основе разработанных алгоритмов;

6. подтверждение заявленных преимуществ АВП на примерах решения задач: а) мониторинга лесного хозяйства; б) проведения геодезических работ.

Результаты, выносимые на защиту:

1. результаты анализа специфических особенностей геоданных, структуры веб-представлений геоданных и их сопоставление с возможностями, предоставляемыми стандартом HTML5;

2. метод функционирования клиентского приложения нового типа, названного АВП, который согласуется со спецификой веб-представлений геоданных и базируется на возможностях, предоставляемых стандартом HTML5, обеспечивая при этом: автономную работу веб-клиента с полученными с сервера данными; автоматическое аварийное резервирование вводимых пользователем данных при потере соединения с сервером; поддержку произвольных типов стационарных и мобильных клиентских устройств;

3. математические модели, разработанные для оценки количественных преимуществ использования веб-приложений на базе технологий стандарта HTML5 (АВП) для доступа к геоданным в зависимости от числа локально сохраненных элементов и их объема. Использование разработанных моделей показало преимущества АВП по сравнению с традиционным веб-приложением: экономию трафика – более 30%; уменьшение времени загрузки обновлений данных – более 86%; возможность моментального запуска.

4. архитектура АВП, основу которой составляют объединенные разработанными в диссертации алгоритмами элементы стандарта HTML5 (Local Storage, Application Cache), библиотека jQuery, а также концепция построения интерактивных веб-интерфейсов AJAX.

5. подтверждение заявленных преимуществ АВП на примерах решения задач: а) мониторинга лесного хозяйства; б) проведения геодезических работ.

Реализованное согласно предложенной архитектуре АВП использовалось для проведения натурных экспериментов и подтверждения достоверности разработанных моделей.

Методы исследования основаны на методах системного анализа прикладных объектов сбора, хранения и удаленной обработки информации, математического моделирования, а также

экспериментального определения характеристик пользовательских выборок данных, каналов связи и клиентских терминальных устройств.

Практическая значимость полученных в работе результатов заключается в создании нового типа веб-приложений, обеспечивающих отказоустойчивое взаимодействие пользователей различных клиентских платформ в компьютерных сетях при решении задач мониторинга земной поверхности и работы с геоданными, включая возможность полностью автономной работы с предварительно загруженными выборками данных. Это позволяет расширить круг терминальных устройств и обеспечить доступ конечных пользователей к данным там, где раньше это было невозможно. Одновременно обеспечивая экономию трафика и увеличение скорости загрузки страниц по сравнению с традиционным веб-приложением.

Диссертация является результатом исследований, проводимых на кафедре 609 «Прикладная информатика» Аэрокосмического факультета ФГБОУ ВО «Московского авиационного института (национального исследовательского университета)» в рамках научного проекта №14-08-01028а РФФИ; выполнена при финансовой поддержке Министерства образования и науки Российской Федерации в рамках базовой части государственного задания в сфере научной деятельности, проект № 834.

Апробация работы и публикации

Основные результаты доложены и обсуждены на всероссийских конференциях молодых ученых и студентов «Информационные технологии в авиационной и космической технике» (Москва, МАИ, 2008, 2009 гг.), VI всероссийской конференции студентов, аспирантов и молодых ученых «Технологии Microsoft в теории и практике программирования» (Москва, МАИ, 2009), 8-й, 9-й, 13-й международных конференциях «Авиация и Космонавтика» (Москва, МАИ, 2009, 2010, 2014 гг.), международных научно-практических конференциях «Развитие науки и образования в современном мире» (Москва, АР-Консалт, 2014), «Наука, образование, общество: тенденции и перспективы» (Москва, АР-Консалт, 2014), «Актуальные проблемы развития современной науки и образования» (Москва, АР-Консалт, 2015).

Список публикаций по теме диссертации содержит 15 наименований, из них 5 – в изданиях, рекомендованных ВАК.

Личный вклад автора. Диссертантом решена актуальная задача обеспечения широкого доступа к геопространственным данным за счет разработки нового метода обеспечения отказоустойчивого доступа к геоданным с различных клиентских устройств, включая мобильные, а также возможности автономной работы пользователей с полученными с сервера данными на их персональных устройствах при решении различных видов задач. В основе предложенного в диссертации решения лежит учет специфических особенностей веб-представлений геоданных и согласующихся с ними возможностей современных веб-

технологий, включая стандарт HTML5. Все представленные в работе результаты (метод функционирования АВП, математические модели сравнительных оценок АВП с традиционными веб-приложениями, алгоритмы функционирования, архитектура и программный комплекс АВП) получены автором лично.

Внедрение результатов работы

Результаты работы внедрены в деятельность организаций ОАО «Союзгипрозем», ООО «УК «Строительные-Технологии» в состав используемых данными организациями геоинформационных систем в виде метода функционирования, архитектуры, программной реализации АВП и методики модернизации существующего традиционного веб-приложения до АВП, а также внедрены в учебный процесс кафедры «Прикладная информатика» МАИ и на веб-ресурс Аэрокосмического факультета МАИ.

Достоверность результатов подтверждается корректным использованием методов системного анализа, математического моделирования, а также отсутствием существенных расхождений результатов проведенных расчетов с результатами проведенных натурных экспериментов.

Структура и объем работы

Диссертация состоит из введения, трех глав, заключения и списка литературных источников из 86 наименований. Работа изложена на 149 страницах машинописного текста, содержит 51 рисунок, 22 таблицы.

В первой главе рассмотрена структура системы обеспечения доступа к данным ДЗЗ и определено место АВП в ней. Отмечено, что в настоящее время «Концепцией развития российской космической системы дистанционного зондирования Земли на период до 2025 года» директивно определен ряд актуальных задач, связанных с необходимостью расширения использования данных ДЗЗ. Среди этих задач важное место занимает задача существенного расширения доступа пользователей к данным результатов космической деятельности через сеть Интернет с использованием различных терминальных устройств.

В этой же главе выполнен анализ особенностей и проблем ГИС с использованием данных ДЗЗ, потребностей использования и структуры веб-представлений геоданных, и сформулированы следующие требования к данной системе.

Выполнен анализ современных технологий, обеспечивающих доступ к таким ГИС через сеть Интернет, на соответствие сформулированным требованиям, в ходе которого установлено, что только предлагаемое в работе решение полностью соответствует этим требованиям. Показаны преимущества, которые обеспечит предлагаемое решение.

Известно, что на сегодняшний день проблемой являются потери информации за счет сбоев и помех в каналах передачи данных, особенно при использовании беспроводных каналов,

а также необходимость повторной загрузки больших объемов данных при использовании веб-приложений. Данные обстоятельства особо негативно сказываются при доступе к геоданным через Интернет, поскольку вероятность помех ограничивает длительность сеансов работы с данными, а использование веб-приложений исключает возможность автономного локального сохранения и использования индивидуальных наборов данных отдельными пользователями.

С принятием стандарта HTML5 названные ограничения стало возможным устранить с помощью реализации долговременных локальных хранилищ данных на стороне клиента, средствами HTML5. Данное обстоятельство легло в основу представленной диссертационной работы и вылилось в предложение нового метода функционирования веб-приложений для систем мониторинга земной поверхности и ГИС с использованием данных ДЗЗ, называемых в работе автономными веб-приложениями (АВП).

Глава содержит определение основных положений предложенного метода, базовые процессы функционирования веб-приложений согласно этому методу, а также концептуальную структуру реализующего эти процессы программного комплекса.

Вторая глава посвящена построению математической модели, содержащей зависимости основных показателей разработанной клиент-серверной системы с АВП (а именно: повышения отказоустойчивости, уменьшения количества запросов к серверу, расхода трафика, времени загрузки веб-страниц, суммарного времени ожидания ответа сервера, времени использования канала связи) от характеристик данных, постоянных локальных хранилищ клиентского устройства, канала передачи данных и последовательности событий.

Рассмотрены различные режимы локального кэширования на клиентском устройстве вводимых пользователем и получаемых с сервера данных. Приведены данные модельного сравнения АВП с традиционными веб-приложениями, показывающие преимущества АВП по сравнению с традиционными веб-приложениями.

Качественные преимущества АВП, которые традиционное веб-приложение не может обеспечить: возможность автономной работы пользователей с полученными с сервера данными и автоматическая актуализация локально сохраненных данных; возможность аварийного резервирования вводимых пользователем данных и автоматическая отправка сохраненных данных при возобновлении соединения с сервером;

Количественные преимущества АВП, которые обеспечиваются за счет долговременного сохранения получаемой с сервера информации на стороне клиента и поддающиеся численному сравнению: уменьшение расхода трафика (за счет загрузки части общего объема загружаемых данных из локального хранилища); уменьшение времени ожидания загрузки веб-страниц (за счет уменьшения загружаемого с сервера объема данных и снижения количества запросов к серверу).

Третья глава посвящена разработке архитектуры программного комплекса АВП и реализации его модулей.

При разработке и реализации АВП использовались основные конструктивы стандарта HTML5 и современные веб-технологии. Ими, в частности, являются: постоянное локальное хранилище данных Local Storage – для локального резервирования введенных пользователем данных для обеспечения надежной доставки вводимых пользователем данных на сервер вне зависимости от качества соединения; постоянное локальное хранилище данных Application Cache – для локального сохранения данных, получаемых с сервера, для обеспечения автономной работы АВП с уже загруженными данными при разрывах соединения с сервером; кроссбраузерная JavaScript библиотека jQuery – для обеспечения одинакового функционирования и унификации программного кода клиентской части разработанной системы для всех современных веб-браузеров различных терминальных устройств.

Для демонстрации возможностей интеграции предлагаемого решения с существующими традиционными веб-приложениями и их модернизации до АВП выбрана система WackoWiki.

Приведена методика тестирования работоспособности разработанного программного комплекса АВП при автоматическом аварийном резервировании вводимых пользователем данных и их восстановлении на сервер, установке, автономной работе и обновлении клиентской части АВП, управлении ее конфигурацией с сервера. Результаты проведенного тестирования позволяют сделать вывод о возможности безошибочного перехода от традиционного веб-приложения к АВП. Рассмотрены примеры использования АВП применительно к решению двух классов практических задач:

а) мониторинга земной поверхности в интересах лесного хозяйства;

б) проведения геодезических работ.

Показано, что при хорошей связи с сервером АВП обеспечивает количественные преимущества по сравнению с традиционным веб-приложением, при плохой – качественные и количественные, а при ее полном отсутствии – качественные.

ГЛАВА 1. ОПРЕДЕЛЕНИЕ АВТОНОМНОГО ОТКАЗОУСТОЙЧИВОГО ВЕБ-ПРИЛОЖЕНИЯ ДЛЯ ГЕОИНФОРМАЦИОННЫХ СИСТЕМ С ИСПОЛЬЗОВАНИЕМ ДАННЫХ ДИСТАНЦИОННОГО ЗОНДИРОВАНИЯ ЗЕМЛИ

1.1 Место АВП в структуре системы дистанционного зондирования Земли и предъявляемые к АВП требования

Представленные в данной диссертационной работе веб-приложения, названные автономными веб-приложениями, рассматриваются как компонент системы дистанционного зондирования Земли, предназначенный для обеспечения доступа пользователей к ГИС с использованием данных ДЗЗ. Исходя из этого, далее рассматривается место АВП в структуре системы ДЗЗ и предъявляемые к АВП требования, выполнение которых должно расширить круг потенциальных пользователей космической информации (КИ), что является одной из основных целей развития системы ДЗЗ.

Традиционно структура системы ДЗЗ включает космический сегмент в виде орбитальной группировки космических аппаратов (КА) ДЗЗ и наземный сегмент. Наземный сегмент в контексте данной диссертационной работы представляется в составе:

- комплекса приема, обработки и хранения космических данных;
- системы доступа к данным ДЗЗ.

Согласно «Концепции развития российской космической системы дистанционного зондирования Земли на период до 2025 года», наряду с развитием группировки КА ДЗЗ, наземный комплекс этой системы должен быть существенно модернизирован и сформирован в виде Единой территориально-распределенной информационной системы ДЗЗ (ЕТРИС ДЗЗ), обеспечивающей эффективное применение орбитальной группировки КА ДЗЗ [12]. При этом можно выделить два направления развития наземного комплекса, соответствующих названным выше его составляющим. Первое из них состоит в переходе от «разнородных и разобщенных центров, … многие из которых имеют слабое техническое оснащение», к созданию целостной системы развитых аналитических сервисов, в частности, в виде отраслевых центров обработки информации и подготовки данных по запросам конечных пользователей. Их основным назначением является прием, декодирование, аналитическая обработка массивов полученной информации для формирования целостных наборов непротиворечивых данных об участках поверхности Земли, а также их размещение в хранилищах данных отраслевых центров,

образующих Единый банк геоданных, который должен иметь собственные поисковые средства [2].

Второе направление развития наземного комплекса связано с созданием системы доступа пользователей КИ к хранилищам данных ДЗЗ. В рамках создания ЕТРИС ДЗЗ на неё возлагается задача достижения одной из основных целей системы ДЗЗ в целом, а именно: максимальное расширение круга потенциальных пользователей информацией, хранящейся в Едином банке геоданных ДЗЗ [12, 13, 14]. В представленной диссертационной работе предлагается осуществить такое расширение за счет включения в состав системы обеспечения доступа как её отдельных компонентов – ГИС на базе веб-приложений с расширенными функциями. Эти веб-приложения названы автономными веб-приложениями (АВП) и являются предметом рассмотрения настоящей работы.

Сразу определимся, что ГИС с использованием данных ДЗЗ имеет клиент-серверную организацию. Это связано с тем, что объем этих данных настолько велик [15], что их нельзя полностью передать на пользовательское устройство, целиком сохранить или обработать на нем. В связи с этим в рассматриваемой системе доступа выделяются две части: серверная и клиентская. Клиентская часть при этом определяется пользовательским устройством (терминалом). В этой структуре рассматриваемые АВП являются клиентской частью и резервируют долю ресурса серверной части, дополняя серверную функциональность и интегрируясь с ней.

АВП играет роль посредника между пользователями с их терминальными устройствами с базовым ПО и серверными средствами управления геоданными, образующими интерфейс к различным источникам геоданных. На рис. 1.1 показаны связи АВП с внешними системами.

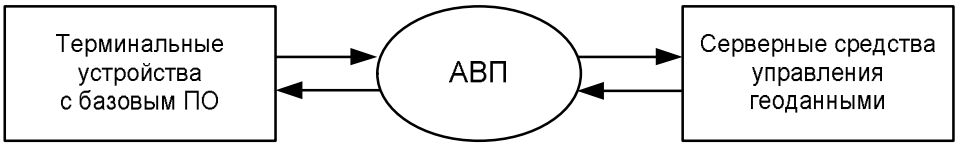


Рис. 1.1. Связи АВП с внешними системами

Именно на данный компонент возлагаются задачи, решение которых должно обеспечить расширение круга потенциальных пользователей геоданных. Это расширение достигается, в первую очередь, за счет снятия ограничений на применяемые пользователями терминальные устройства, на их географическое положение относительно мест расположения серверов ГИС, а также на используемые каналы связи. При этом особое внимание уделяется обеспечению работы пользователей в условиях, существенно затрудняющих коммуникации пользователей с хранилищами геоданных.

Рассмотрим сказанное подробнее и сформулируем требования к АВП, выполнение которых должно способствовать достижению названной выше цели – расширению круга потенциальных пользователей ГИС с использованием данных ДЗЗ.

Первое требование, предъявляемое к АВП как программному компоненту ГИС с использованием данных ДЗЗ, является возможность работы с различными видами программно-аппаратных платформ устройств, применяемых пользователями в качестве терминалов. В их числе могут находиться как стационарные устройства, в частности, персональные компьютеры (ПК), так и различного рода мобильные устройства, рост количества и разнообразия которых в настоящее время весьма существенен [67]. Уже в настоящее время владельцами хотя бы одного из таких устройств (смартфон, планшет), которое может использоваться в качестве терминала различных клиент-серверных информационных систем (ИС), является большинство населения. Поэтому можно говорить, что при удовлетворении рассматриваемого требования, то есть при наличии стандартного интерфейса между ГИС и различного вида мобильными и стационарными пользовательскими устройствами, каждый владелец хотя бы одного из этих устройств может рассматриваться как потенциальный пользователь ГИС. Данное требование к АВП далее будем называть **обеспечением кроссплатформенности.**

Ответом на требование кроссплатформенности является использование веб-приложений. Так как АВП выполняется в среде стандартных современных веб-браузеров (далее для краткости – браузеров), присутствующих на всех устройствах, то для перехода пользователя из состояния потенциального в состояние реального пользователя на его устройстве не требуется установка специализированного программного обеспечения (ПО).

Следующая группа требований связана с тем, что для максимизации числа пользователей геоданных необходимо использовать публичные, а с ростом числа мобильных устройств – беспроводные каналы связи и, в первую очередь, сеть Интернет. Это предъявляет особые требования к системе обеспечения доступа к данным, учитывая, что в загруженных публичных каналах связи невозможно обеспечить соединение постоянного качества между клиентом и сервером и неустранимо присутствует вероятность сбоев из-за влияния возможных помех, а также возможны периоды отсутствия соединения. Особенно, если преследуется цель расширения круга пользователей геоданных, то ГИС должна ориентироваться на неблагоприятные условия, в том числе экстремальные, когда связь может длительное время отсутствовать, работать с перебоями, или канал связи имеет малую пропускную способность.

Данные обстоятельства должны парироваться выполнением двух требований, которые далее определяются как **обеспечение отказоустойчивости** и **обеспечение автономной работы АВП.** Конкретизируем эти требования.

Выделим две основные черты, характерные для данных, хранящихся на серверах ГИС с использованием данных ДЗЗ. Первой из них является то, что объем запрашиваемых пользователем данных настолько велик, что требует длительного сеанса связи с серверной частью ГИС. Чтобы ускорить отображение загруженных данных и не дожидаться окончания загрузки всего набора, в системах с веб-интерфейсом общий объем загружаемых клиентом данных делится на более мелкие файлы, размер которых варьируется в пределах десятков-сотен Кбайт. Количество этих небольших файлов пропорционально суммарному объему информации, которую требуется загрузить. Например, чтобы загрузить 1 Гигабайт потребуется 10000 файлов по 100 Кбайт. При этом для беспроводной связи требуется значительное время ожидания соединения при передаче каждого составляющего элемента веб-страницы (далее для краткости – страницы). Учитывая, что страницы состоят из множества элементов, скорость их загрузки может составлять минуты, например, при спутниковом соединении. По статистике средняя страница в Интернете состоит из 96 элементов общим объемом 1977 Кбайт (по состоянию на 1 февраля 2015 года [52]), а объем и количество элементов страниц продолжают расти. Страницы веб-интерфейса ГИС с использованием данных ДЗЗ, рассматриваемого в данной работе, имеют существенно больший объем и количество составляющих элементов в зависимости от отображаемой выборки данных.

Беспроводные сети, в которых предполагается работа мобильного клиента клиент-серверной ИС – это сети, построенные по технологиям Wi-Fi (IEEE 802.11b/g/n), GSM (GPRS и EDGE на базе GSM-900, GSM-1800), LTE, спутниковая связь. Согласно методике проведения оценочных испытаний и норм на показатели качества услуг связи стандартов GSM/GPRS/EDGE/UMTS Департамента информационных технологий города Москвы доля неуспешных попыток установления мобильного соединения для передачи данных в норме составляет от 4% до 6% в местах покрытия сотовой сети, а норма покрытия сети составляет от 85% до 95% вне зданий и от 70% до 80% внутри зданий. Также в методике приведена норма разрывов установленного соединения не по инициативе абонента от 2% до 5% на примере голосовых соединений [24]. При этом во многих случаях качество сотовой связи ниже нормы (на примере голосовых соединений от 10% до 12% отказов при норме от 3% до 5% неуспешных вызовов для голосовых соединений) [10]. В этих условиях можно предположить, что чем больше количество элементов страницы и чем больший объем данных загружается, а, следовательно, чем более продолжительна загрузка по времени, тем более вероятно возникновение разрыва соединения во время этой загрузки.

Учитывая, что клиенты могут помимо загрузки данных с сервера уже сейчас использовать средства обратной связи и коммуникативную онлайн-платформу ГИС, а в будущем, вероятно, смогут отправлять в систему свои текстовые комментарии к различным

материалам ГИС (что уже встречается в некоторых геоинформационных веб-приложениях [17, 81]), требуется обеспечить не только сохранение полученных с сервера данных, но и резервирование вводимых пользователем данных для обеспечения их надежной доставки на сервер.

В этих условиях требование отказоустойчивости предлагается обеспечить за счет долговременного сохранения в локальном хранилище терминального устройства пользователя:

- некоторого набора данных, получаемых с сервера, который позволит уменьшить общий загружаемый объем данных за счет отказа от повторной загрузки данных и общее время загрузки, что снизит влияние сбоев соединения на работоспособность веб-приложения пропорционально объему локально сохраненных данных;
- всех данных, отправляемых пользователем на сервер, до подтверждения их получения сервером, что позволит полностью избежать потери введенных пользователем данных при разрыве соединения.

Другой характерной чертой данных ГИС с использованием данных ДЗЗ является то, что, наряду с возможным большим объемом запрашиваемых клиентом данных, период актуальности этих данных может быть достаточно продолжителен в зависимости от содержания, и загружать их повторно при каждом обращении нерационально. Учитывая данное обстоятельство, определим требование автономной работы АВП как необходимость кэширования геоданных на период их актуальности и загрузки только их обновлений в автоматическом режиме. Данное требование, помимо сокращения суммарной длительности сеансов связи клиентских устройств с серверной частью ГИС и, соответственно, уменьшения вероятности разрыва соединений, позволит конечным пользователям в условиях плохого соединения с сервером автономно работать с данными с продолжительным периодом актуальности и автоматически получать обновления при восстановлении соединения. Это позволит сократить трафик пользователя и нагрузку на сеть передачи данных.

Проиллюстрируем преимущества, получаемые при выполнении требования автономной работы АВП, в сравнении с традиционными Интернет-порталами на примере Геопортала Роскосмоса [23], являющегося ГИС с использованием данных ДЗЗ. Здесь можно наблюдать следующую специфику системы: при открытии главной страницы портала загружается 23,60 Мбайт данных (что может происходить достаточно долго при плохом соединении) в результате 83 запросов, отправляемых браузером клиента к веб-серверу.

Из общего указанного объема загружаемых данных подходят для долговременного кэширования на устройстве пользователя:

- файл описания каталога имеющихся в системе карт – 20,80 Мбайт (его для большей эффективности кэширования можно разделить на части: архивная – которая не меняется, и несколько текущих – которые меняются с разной периодичностью);
- главный элемент интерфейса портала, созданный на базе технологии Adobe Flash (исключает поддержку большинства мобильных устройств) – 1,80 Мбайт;
- служебные программы на языке JavaScript в файлах общим объемом до 250 Кбайт;
- служебные изображения – элементы интерфейса – общим объемом до 100 Кбайт.

Таким образом, примерно 22,95 Мбайт из загружаемых 23,60 Мбайт могут быть долговременно закэшированы локально, так как представляют собой интерфейс веб-приложения и каталог имеющихся карт. По количеству запросов долговременное кэширование позволит сэкономить примерно 26 из 83 запросов. Это означает уменьшение объема повторно загружаемых данных на более чем 97% и уменьшение количества запросов к серверу на более чем 31% при инициализации интерфейса Геопортала, что позволит сэкономить трафик при повторных обращениях и уменьшить время загрузки страниц Геопортала.

Для оценки возможности кэширования слоев данных ДЗЗ и других слоев карты рассмотрим подробнее устройство многослойной интерактивной карты с веб-интерфейсом.

Современные интерактивные карты с веб-интерфейсом помимо программных средств управления содержат комбинацию слоев растровых, векторных и текстовых данных, загруженных с привязкой к единой системе координат (рис. 1.2) [20, 31, 37]. Некоторые из этих данных могут сохранять актуальность в течение длительного времени в зависимости от задачи, для которой планируется их использование.

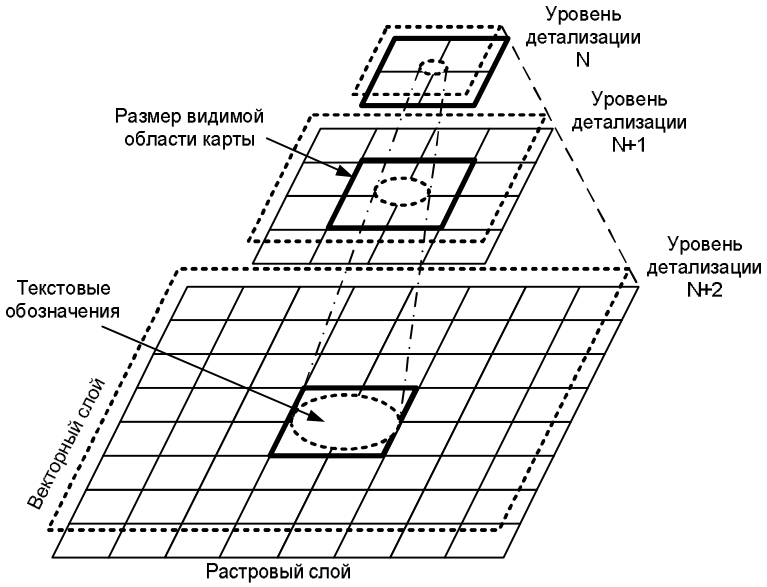


Рис.1.2. Состав элементов многослойной карты

Растровые слои карты состоят из матрицы точек – пикселей, передающих информацию своим цветом (рис. 1.3).

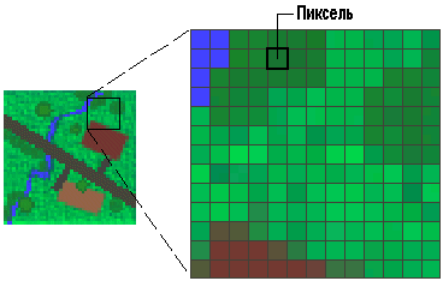


Рис. 1.3. Состав растрового слоя карты

Обычно растровыми слоями карты являются данные ДЗЗ, аэрофотосъемки или отсканированные карты с бумажных носителей [28]. Растровые слои выступают в качестве подложки для векторных слоев или в качестве самостоятельных источников информации о земной поверхности [11].

Растровые слои карты формируются из раздельно загружаемых элементов – тайлов (от англ. tile – плитка) размером 256x256 [57, 75] пикселей и объемом 5-200 Кбайт (наиболее распространенные в ГИС объемы 5-20 Кбайт) каждый для ускорения начала отображения карты пользователю.

Масштабирование растровых слоев при увеличении карты осуществляется увеличением количества элементов слоя в 4 раза для каждого последующего уровня детализации. В зависимости от уровня детализации растровый слой формируется из тайлов различного разрешения (таблица 1.1) [5, 70, 86].

Таблица 1.1. Разрешение тайлов растрового слоя карты в зависимости от уровня детализации

Уровень детализации	Размер тайла в градусах	Видимая область карты	Пространственное разрешение, м / пиксель	Примерный масштаб карты
n	$\frac{360}{2^n} \times \frac{170{,}1022}{2^n}$			
0	360 x 170,1022	Весь мир	156412	1:500 000 000
1	180 x 85,0511		78206	1:250 000 000
2	90 x 42,5256		39103	1:150 000 000
3	45 x 21,2628		19551	1:70 000 000
4	22,5 x 10,6314		9776	1:35 000 000
5	11,25 x 5,3157		4888	1:15 000 000
6	5,625 x 2,6579		2444	1:10 000 000
7	2,813 x 1,329		1222	1:4 000 000
8	1,406 x 0,6645		610,984	1:2 000 000
9	0,703 x 0,3323	Регион	305,492	1:1 000 000
10	0,352 x 0,1662		152,746	1:500 000
11	0,176 x 0,0831	Область	76,373	1:250 000
12	0,088 x 0,0416		38,187	1:150 000
13	0,044 x 0,0208	Деревня или город	19,093	1:70 000
14	0,022 x 0,0104		9,547	1:35 000
15	0,011 x 0,0052		4,773	1:15 000
16	0,005 x 0,0026	Небольшая дорога	2,387	1:8 000
17	0,003 x 0,0013		1,193	1:4 000
18	0,001 x 0,0007		0,596	1:2 000
19	0,0005 x 0,0004		0,298	1:1 000

Для обеспечения плавности перехода между уровнями детализации до загрузки составляющих элементов следующего уровня осуществляется пропорциональное растягивание (или сжатие при уменьшении карты) имеющихся элементов карты.

Растровые элементы карты обновляются редко, и тем реже, чем ниже уровень детализации.

Работа пользователя происходит на уровнях детализации с 14 до 19, поэтому для обеспечения автономной работы следует локально сохранить определенное количество тайлов растрового слоя карты, которое вместе с их суммарным объемом определяется площадью участка карты (таблица 1.2).

Таблица 1.2. Общее количество и суммарный объем данных сохраняемых тайлов растрового слоя в зависимости от площади участка в уровнях детализации с 14 по 19 при объеме одного тайла 20 Кбайт

Площадь участка, км^2	Общее количество тайлов, шт.	Суммарный объем данных, Кбайт
1	267	5340
2	500	10000
4	995	19900
9	2138	42760
16	3803	76060
25	5849	116980
36	8375	167500
49	11298	225960
100	23264	465280
225	51953	1039060
900	207140	4142800

Если растровая подложка, состоит не из набора тайлов, а представлена единым динамически генерируемым по размеру экрана изображением с заданными координатами и необходимым уровнем детализации (технология WMS [77, 82]), то для обеспечения работы АВП в клиентский интерфейс отображения карты вносится доработка, благодаря которой клиент запрашивает у сервера не произвольные области WMS, а изображения смежных областей с постоянными адресами, заданными типовой координатной сеткой, и отображает их аналогично тайлам [83]. Другим способом реализации работы АВП с WMS может быть использование промежуточного преобразователя из WMS в тайлы на стороне сервера [74, 80]. В обоих случаях становится возможным кэширование АВП полученной с сервера WMS информации.

Векторные слои карты состоят из точек, полилиний и полигонов, формирующих графические объекты векторного слоя, а также связанных с ними текстовых описаний или аналитической информации (рис. 1.4) [1, 5].

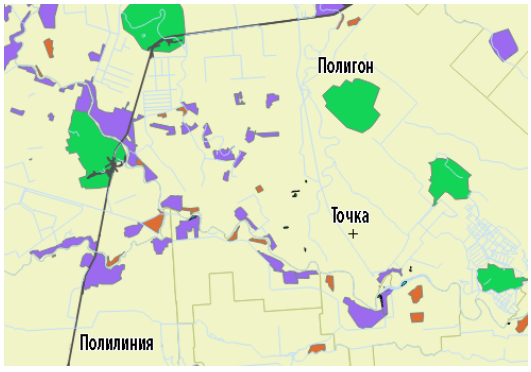


Рис. 1.4. Состав векторного слоя карты

На векторных слоях обычно отображаются геопространственные и тематические объекты с точно определяемыми границами, например: административно-территориальное деление, населенные пункты, дороги, городской и земельный кадастр, различные метки, данные о пожарах, метеоданные и другое [19, 22, 68].

Векторные слои отображаются на карте, начиная с определенного уровня детализации. При небольшом количестве элементов на слое загружаются сразу полностью. При большом количестве элементов слои загружаются частями, соответствующими отображаемому на экране участку карты и уровню ее детализации, что обеспечивает наилучшие восприятие карты пользователем и производительность терминального устройства.

Объем данных векторного слоя зависит от количества и характеристик отображаемых на нем элементов.

Масштабирование векторных слоев при увеличении карты осуществляется пропорциональным растягиванием (сжатием) и дозагрузкой (удалением) элементов, отображаемых при большей детализации. Это возможно благодаря специфике векторного изображения, а также многослойной структуре карты.

Векторные слои могут обновляться часто, так как их обновление может не зависеть от получаемых данных ДЗЗ и не требует большой серверной обработки. Фактическая частота обновления зависит от содержания отображаемой информации и периода ее актуальности, определяемого в соответствии с назначением.

Согласно «Концепции развития российской космической системы дистанционного зондирования Земли на период до 2025 года» определено 105 задач в различных сферах деятельности человека, решаемых с применением данных ДЗЗ.

В зависимости от сферы деятельности требуются различные виды данных ДЗЗ по пространственному и радиометрическому разрешению, спектральному диапазону и с различным сроком актуальности. Для оценки потребностей и возможностей долговременного кэширования данных ДЗЗ, применяемых в различных задачах, основными параметрами являются пространственное разрешение (позволяет определить, какие уровни детализации данных нужно кэшировать для решения конкретной задачи) и срок актуальности данных (позволяет определить возможности экономии трафика для конкретной задачи) (рис. 1.5).

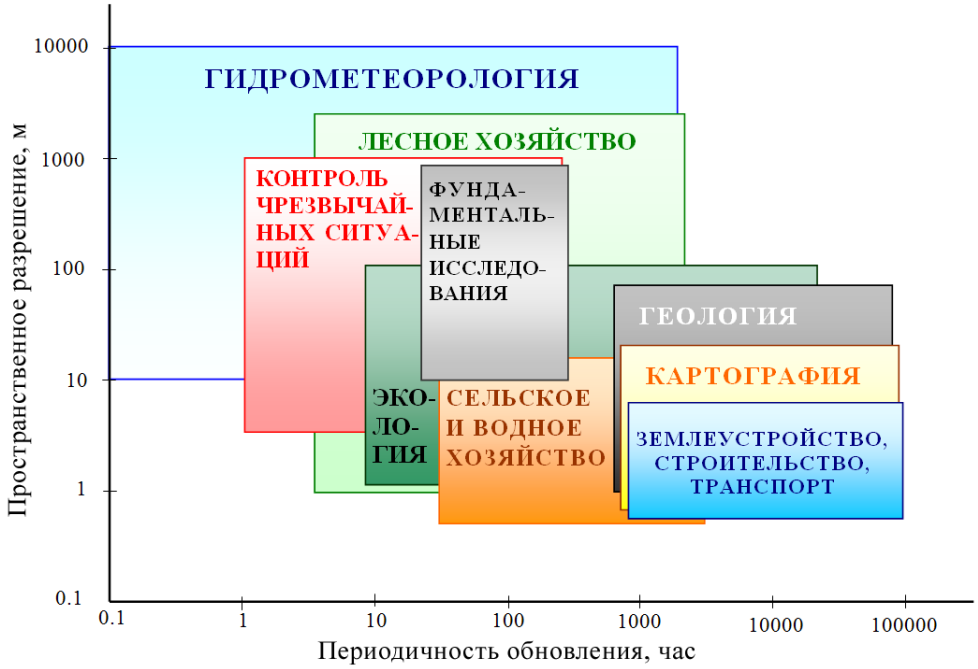


Рис. 1.5. Требования к периодичности обновления и пространственному разрешению данных ДЗЗ для различных областей применения

Обобщенный список задач включает в себя гидрометеорологию, картографию, контроль чрезвычайных ситуаций, фундаментальные исследования, охрану природы (экология), землеустройство, составление кадастров, инвентаризацию посевов, контроль хода производственных процессов в сельском, лесном, водном хозяйстве и других отраслях природопользования, прогноз урожая, выявление заболеваний и распространения насекомых-вредителей на лесных угодьях и сельскохозяйственных территориях, информационное обеспечение поиска полезных ископаемых, слежение за процессами урбанизации, подготовку строительства и прокладки транспортных магистралей и так далее [12].

Итак, предложенное к рассмотрению в настоящей диссертационной работе АВП определено как компонент ГИС с использованием данных ДЗЗ и может рассматриваться как часть системы дистанционного зондирования Земли, ориентировано на расширение круга потенциальных пользователей геоданных и для достижения этой цели должно удовлетворять следующим требованиям:

1. кроссплатформенности, которая состоит в унифицированной поддержке различных терминальных устройств, включая мобильные;
2. отказоустойчивости, которая реализуется за счет:
   2.1. сохранения введенных пользователем данных при разрыве соединения с сервером,
   2.2. восстановления сохраненных данных на сервер при возобновлении соединения с сервером;

3. автономности, которая реализуется за счет:
    3.1. возможности автономной работы с данными с продолжительным периодом актуальности в условиях отсутствия соединения с сервером,
    3.2. автоматического получения обновлений сохраненных данных при появлении соединения с сервером.

1.2 Влияние клиент-серверных технологий на эволюцию ГИС

В настоящее время потребители геоданных используют различные компоновки пакетов ПО для работы с этими данными в автономном режиме и в рамках клиент-серверного взаимодействия. Рассмотрим компоновки ПО для работы с геоданными, в том числе данными ДЗЗ, в порядке их возникновения в контексте соответствия сформулированным требованиям.

1.2.1 Отдельные приложения для ПК

Первыми появились автономные приложения для ПК для решения различных задач работы с геоданными, в том числе данными ДЗЗ, например, в области геодезии, топографии, систем инженерных коммуникаций, недропользования, землеустройства и других.

Примерами таких приложений для работы с геоданными являются ArcGIS, ENVI, ERDAS IMAGINE, Generic Mapping Tools, GeODin, Geomatica, Global Mapper, Google Earth, GRASS, gvSIG, NASA World Wind, PHOTOMOD, QGIS, SAS.Планета и другие.

Из-за особенностей реализации для автономной работы сетевые функции в таких приложениях могут касаться только передачи исходных данных и результатов работы или полностью переложены на пользователя, который получает выходной файл с результатом работы и может распоряжаться им по своему усмотрению.

Такие приложения:

- не являются кроссплатформенными в полной мере, а поддерживают одну или несколько операционных систем (ОС) для ПК;
- являются отказоустойчивыми за счет того, что работа осуществляется с локальными данными и отправка результатов куда-либо происходит только после их локального сохранения;
- являются полностью автономными и не зависят от сервера.

1.2.2 Клиент-серверное интегрированное приложение АРМ для ПК

С развитием возможностей сетевого взаимодействия возникла клиент-серверная организация приложения с клиентскими автоматизированными рабочими местами (АРМ) и серверным хранением и обработкой информации. Применяется для работы с различными

геоданными, в том числе данными ДЗЗ, для совместного анализа больших массивов данных и ввода результатов анализа данных в общие геоинформационные БД.

Примерами клиент-серверных интегрированных приложений АРМ являются ArcGIS for Server, GeODin Client/Server и другие.

Клиентская часть клиент-серверного интегрированного приложения АРМ для ПК представляет собой сетевое приложение, требующее подключения к серверу. Имеет минимум автономных функций или не имеет их вовсе и является лишь тонким клиентом. В редких случаях – частично автономное приложение с периодической синхронизацией с сервером. Серверная часть приложения содержит все данные и большую часть алгоритмов приложения. Такая организация позволяет работать с большими объемами данных, чем отдельные приложения для ПК, и задействовать производительность специализированного сервера для обработки.

Такие приложения:

- не являются кроссплатформенными в полной мере, так как клиентские приложения поддерживают одну или несколько операционных систем (ОС) для ПК;
- могут быть отказоустойчивыми за счет локального сохранения вводимых пользователем данных на стороне клиента;
- не являются автономными и зависят от сервера (в редких случаях частично автономны).

1.2.3 Традиционное веб-приложение

С повышением интенсивности труда, ростом пропускной способности публичных каналов связи и увеличением распространения мобильных устройств требуется обеспечить автоматизированные рабочие места вне стен предприятий или офисов полным набором информации, необходимой для работы.

Геоданные используются в различных сферах жизни, их хранилища становятся распределенными. В связи с этим развиваются традиционные веб-приложения, основным плюсом которых является возможность получать данные из различных БД и объединять их в едином веб-интерфейсе.

Примерами таких приложений для работы с геоданными являются Геопортал Роскосмоса, Космоснимки, Яндекс.Карты, ArcGIS Online, Google Maps, Wikimapia и другие.

Традиционное веб-приложение требует постоянного подключения к серверу для обмена данными и реакции на события интерфейса. Позволяет работать практически с любого устройства с установленным браузером. Интеграция с различными БД, а также управление их отображением и контроль доступа осуществляется серверной частью приложения.

Такие приложения:

- являются кроссплатформенными, требуют для работы только наличие браузера на терминальном устройстве;
- не являются отказоустойчивыми, так как не могут сохранить введенные пользователем данные;
- не являются автономными и полностью зависят от сервера.

При любом сбое подключения к серверу браузер сообщает пользователю о том, что сервер недоступен и страницу отобразить нельзя. Если сбой происходит в момент отправки данных заполненной веб-формы, то эти данные в случае с традиционным веб-приложением будут потеряны. Даже нажатие кнопки «Назад» браузера зачастую не приводит к возврату на страницу с уже заполненной формой в случае нарушения соединения с сервером.

Пользователю было необходимо всегда помнить, что веб-сайт в браузере – это только вид консоли сервера: обработка и хранение данных происходит на сервере (на локальном компьютере – только ввод/вывод), и все риски потери соединения ложатся на пользователя.

Потребность в резервировании вводимых пользователем данных возникла с появлением первых традиционных веб-приложений, позволивших отправлять данные на веб-сервер.

С развитием языка JavaScript и его поддержки браузерами появилась возможность асинхронной работы с сервером, что позволило осуществлять проверку доступности сервера перед отправкой данных.

Этот новый механизм позволял пользователю вручную сохранить введенные данные из формы при получении уведомления о недоступности сервера или заставлял ожидать, не закрывая браузера, восстановления соединения с сервером для повторной отправки данных заполненной формы. Однако этот механизм не мог обеспечить никакой защиты в случаях, когда соединение с сервером разрывалось после проверки доступности сервера уже в процессе передачи данных.

Без резервирования вводимых пользователем данных веб-приложения всегда уступали по надежности ввода данных тонким клиентам клиент-серверных ИС.

Что касается локального сохранения больших объемов данных средствами веб-приложения, то попытки, предпринимавшиеся ранее, не смогли обеспечить универсальную поддержку технологий всеми устройствами, так как проприетарные (являющееся частной собственностью авторов или правообладателей) [38] технологии не могут быть включены в стандарт для всех устройств. Среди этих технологий:

- Oracle (ранее Sun) Java – разрабатывалась для создания приложений и распространения их через Интернет. Требует наличия на каждом устройстве установленной среды выполнения Java. Среда выполнения Java поддерживается далеко не всеми ОС. В

настоящее время акцент в применении технологии перешел на приложения для ограниченного круга корпоративных потребителей;

- Adobe Flash – разрабатывалась для передачи и воспроизведения медиа-контента через Интернет в браузере пользователя. Не была предназначена для постоянного хранения больших объемов данных, а только для кратковременного кэширования. Требует наличия на каждом устройстве установленного Flash Player. Flash Player не поддерживается большинством мобильных ОС. В настоящее время технология используется при распространении медиа-контента;
- Microsoft Silverlight – конкурент Adobe Flash от компании Microsoft. Разрабатывалась для передачи и воспроизведения медиа-контента через Интернет в браузере пользователя. Не была предназначена для постоянного хранения больших объемов данных, а только для кратковременного кэширования. Требует наличия на каждом устройстве установленного плеера Silverlight. Плеер Silverlight не поддерживается большинством мобильных ОС и некоторыми ОС для ПК. В настоящее время технология используется при распространении медиа-контента компанией Microsoft;
- Google Gears – первая технология, которая разрабатывалась специально для организации долговременных локальных хранилищ больших объемов данных в браузерах пользователей. Требовала наличия на ПК пользователя установленного компонента Google Gears. Компонент Google Gears поддерживался не всеми браузерами и не всеми ОС для ПК, не поддерживался всеми мобильными ОС. Проект Google Gears закрыт в пользу поддержки компанией Google стандарта HTML5 [51].

В настоящее время дополнительные модули для браузеров от различных разработчиков не позволяют добиться всеобщего охвата потребителей с разными браузерами, операционными системами и устройствами.

При этом ни в одном из названных проектов проблема обеспечения отказоустойчивости или полностью автономной работы самих веб-приложений не рассматривалась. В самих браузерах не было предусмотрено постоянных хранилищ, и в среде браузера не могло выполняться сложное функционально насыщенное веб-приложение.

Разработчики пытались решать назревшие проблемы повышения доступности геоданных, в том числе данных ДЗЗ, с помощью традиционных веб-приложений, выводя данные в браузер. Но сделать что-то более существенное, чем просмотр данных, не получилось, так как возможности браузеров до последнего времени не предусматривали ничего, кроме отправки запросов к серверу, отображения текста, изображений, манипулирования отображением объектов на экране и подключения плагинов, которые не универсальны.

1.2.4 Мобильное приложение

Мобильные приложения для работы с геоданными стали появляться по мере распространения мобильных устройств в ответ на необходимость обеспечения геоданными и результатами их анализа пользователей вдали от стационарных рабочих мест и для обеспечения некоторого автономного функционала, недоступного в традиционных веб-приложениях.

Примерами мобильных приложений являются ArcGIS, Яндекс.Карты, Google Maps.

Как и единое интегрированное приложение АРМ для ПК, мобильное приложение для работы с геоданными представляет собой клиент-серверное приложение, требующее подключения к серверу, и может иметь ряд автономных функций.

Такие приложения:

- не являются кроссплатформенными в полной мере, а поддерживают одну или несколько программно-аппаратных платформ мобильных устройств;
- являются отказоустойчивыми, так как могут локально сохранить введенные пользователем данные;
- могут быть частично автономными от сервера.

В связи с тем, что платформы мобильных устройств разнообразны, а одно мобильное приложение не обеспечивает универсальную поддержку всех мобильных устройств, приходится разрабатывать много разного ПО, выполняющего одни и те же функции на разных платформах. Это существенно удорожает процесс разработки и ограничивает количество платформ, поддерживаемых программным решением, что в современных условиях не может удовлетворить потребителя.

Разработчики описанных компоновок используемых программных решений не могли охватить многих потребностей пользователей геоданных – им не хватало технологической базы. Приходилось ограничиваться организацией возможности подключения удаленных пользователей к файловым архивам и некоторым базам геоданных через Интернет.

Это не позволяло получать полноценный оперативный доступ к разнообразным выборкам данных без установки и настройки специального клиентского ПО. Поэтому многие возможности работы с ГИС с использованием данных ДЗЗ были доступны только со специализированного стационарного рабочего места и недоступны мобильным клиентам.

Традиционные веб-приложения, несмотря на все их недостатки, стали примером того, как геоданные могут отображаться пользователям в единой доступной среде, которая не требует установки и продолжительного обучения для начала использования ее клиентом. Использование традиционных веб-приложений дает пользователям готовый к работе инструмент, который не нужно устанавливать, настраивать и конфигурировать [47, 48].

В настоящее время, с ростом унификации браузеров в соответствии со стандартом HTML5 и появлением в них ряда новых функций, появилась возможность с помощью предлагаемого в данной работе нового класса веб-приложений расширить количество потенциальных пользователей ГИС с использованием данных ДЗЗ. С точки зрения затрат на модернизацию инфраструктуры для внедрения таких веб-приложений требуется лишь изменить программное обеспечение на веб-сервере, на котором ранее работало традиционное веб-приложение, и удостовериться, что все клиенты используют браузеры с поддержкой HTML5.

При рассмотрении различных компоновок ПО для работы с геоданными наблюдается ряд проблем. Этими проблемами являются: необходимость специализированного ПО и ограниченный функционал в веб-версии; невозможность использования произвольных мобильных устройств; необходимость загрузки большого объема данных для автономной работы при использовании специализированного ПО и полная невозможность автономной работы в веб-версии.

Перечисленные проблемы могут быть решены с использованием метода, предлагаемого в данной работе, что позволит: нарастить функционал веб-версии и сделать его доступным автономно в любых устройствах со стандартным браузером с поддержкой HTML5; обеспечить единообразную поддержку всех существующих и будущих мобильных устройств; создавать индивидуальные подборки необходимых данных для конкретного пользователя и обеспечивать их автономное использование на любом устройстве без подключения к серверу.

Возможность автономного использования ранее загруженных данных особенно важна при работе с большим объемом данных вдали от стационарного рабочего места, где связь может отсутствовать, работать с перебоями, или где канал связи имеет малую пропускную способность. Принципиальной является необходимость автоматического обновления сохраненной информации на клиентском устройстве в случае изменения на сервере ранее полученных клиентом данных.

Результаты проведенного анализа обобщены в виде таблицы 1.3, из которой видно, что среди доступных в настоящее время типов приложений для работы с геоданными ни одно из них не соответствует одновременно всем рассмотренным выше требованиям. В последней строке данной таблицы анонсировано предлагаемое решение. В следующем разделе работы дается его подробное описание в виде метода и соответствующих процессов функционирования клиент-серверных систем с использованием АВП.

Таблица 1.3. Сравнение возможностей различных типов приложений

Требование / Тип приложения	Кроссплатформенность	Отказоустойчивость		Автономность	
		Сохранение введенных данных при разрыве соединения	**Отправка данных на сервер при восстановлении соединения**	**Возможность автономной работы с полученными данными**	**Актуализация локально сохраненных данных при их изменении на сервере**
Отдельное приложение для ПК	Поддержка отдельных платформ ПК	Не зависит от наличия соединения	Ручное	Всегда работает автономно	Ручное
Клиент-серверное интегрированное приложение АРМ для ПК	Поддержка отдельных платформ ПК	Автоматическое	Автоматическое	Частичная, может не быть долговременного кэша	Автоматическое
Традиционное веб-приложение	Все устройства с браузером	Вероятен возврат к заполненной форме при нажатии кнопки «Назад» браузера (зависит от конкретного браузера), откуда пользователь должен вручную скопировать и сохранить данные	Ручное	Только с открытой веб-страницей	Автоматическое, но данные недоступны автономно после закрытия браузера
Традиционное веб-приложение с асинхронной передачей данных, без учёта возможного разрыва связи	Все устройства с браузером	Нет	Ручное	Только с открытой веб-страницей	Автоматическое, но данные недоступны автономно после закрытия браузера
Традиционное веб-приложение с асинхронной передачей данных, с учётом возможного разрыва связи	Все устройства с браузером	До получения ответа сервера форма не закрывается, что позволяет вручную скопировать и сохранить данные	Ручное или полуавтоматическое при оставлении пользователем открытого окна браузера до восстановления соединения	Только с открытой веб-страницей	Автоматическое, но данные недоступны автономно после закрытия браузера
Мобильное приложение	Поддержка отдельных платформ мобильных устройств	Автоматическое	Автоматическое	Частичная, может не быть долговременного кэша	Автоматическое
Предлагаемое решение	Все устройства с браузером стандарта HTML5	Автоматическое	Автоматическое	Есть	Автоматическое

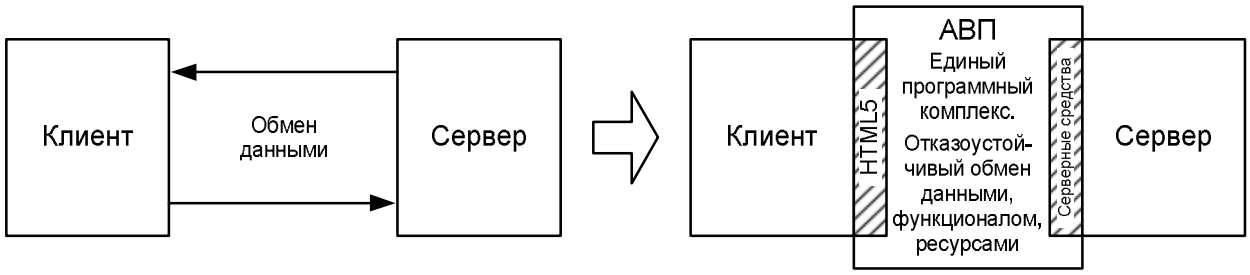
1.3 Основные положения и базовые процессы метода функционирования клиент-серверных систем с использованием АВП

1.3.1 Основные положения метода

Предлагаемый метод функционирования клиент-серверных систем с использованием АВП характеризуется следующими основными положениями.

1. В состав клиент-серверной системы вводится программно-информационный комплекс, названный выше АВП, который обладает следующим свойством: информация, попавшая в него с клиентского устройства или с сервера, сохраняется до тех пор, пока она необходима. Достигается это за счет того, что АВП состоит из двух частей. Одна из них располагается на сервере, а вторая – на клиентском устройстве. При этом полученная с сервера и хранимая на клиентском устройстве информация автоматически актуализируется, а информация, передаваемая с клиентского устройства, хранится на нем до получения подтверждения о её получении сервером и только после этого уничтожается.

Такого рода комплекс, включенный в состав традиционного веб-приложения, занимая место посредника (рис. 1.6) между клиентским устройством и сервером,



а) **б)**

Рис. 1.6. Клиент-серверное взаимодействие
а) на основе традиционного веб-приложения; б) на основе АВП

позволяет обеспечить:

- **отказоустойчивость передачи информации** за счет её буферизации в памяти АВП. В случае потери соединения во время сеанса передачи информации между клиентом и сервером информация сохраняет целостность и не теряется, сеанс может быть продолжен при появлении соединения. Сеанс считается завершенным только после получения подтверждения, что вся запрошенная информация получена;
- **автономный режим работы** пользователя с загруженной с сервера на его устройство информацией с возможностью манипулирования ею, используя заимствованную от

сервера необходимую функциональность. Обеспечивается за счет того, что АВП является самостоятельным программно-информационным комплексом, его функциональность управляется сервером и может выходить за рамки процессов передачи данных. Это позволяет говорить о возможности значительного расширения функциональности АВП. В предлагаемой работе основное внимание уделяется разработке функциональности, обеспечивающей выполнение сформулированных выше требований.

Сказанное определяет принципиальное отличие предлагаемого метода от традиционных веб-приложений, где совместная работа серверного и клиентского компонентов полностью зависит от работоспособности канала передачи данных (например, сети Интернет). При потере соединения с сервером традиционное веб-приложение, как правило, становится полностью неработоспособным. При этом никакие страницы, включая просмотренные ранее, не могут быть отображены, а отправлявшиеся на сервер в момент разрыва соединения введенные пользователем данные с большой вероятностью теряются безвозвратно.

2. Для реализации клиентской части АВП на клиентском устройстве выделяется память для долговременного сохранения данных, получаемых с сервера и вводимых пользователем (рис. 1.7). Для поддержания информации на клиентском устройстве в актуальном состоянии эта информация синхронизируется с сервером при наличии/восстановлении соединения.

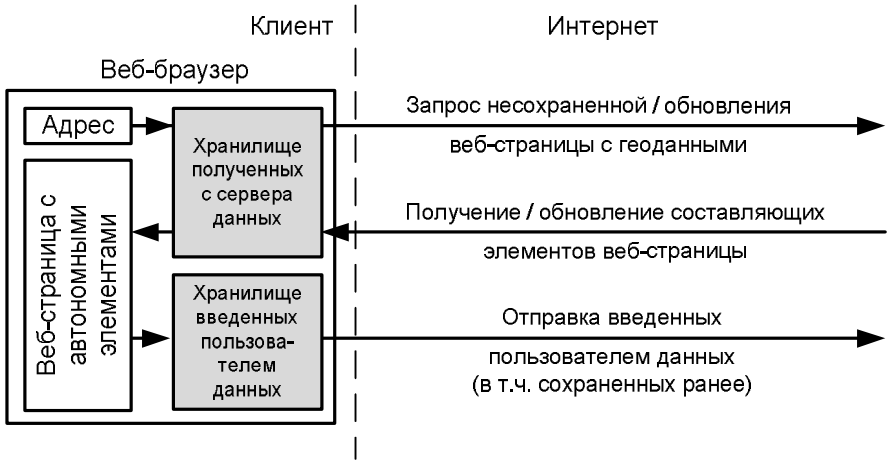


Рис. 1.7. Выделение памяти на клиентском устройстве для сохранения данных обмена между клиентом и сервером АВП

В традиционном веб-приложении, в отличие от АВП, отсутствует возможность долговременного сохранения полученных с сервера и введенных пользователем данных, что и обусловило один единственный режим работы – при наличии подключении к серверу. При потере соединения с сервером традиционное веб-приложение становится полностью

неработоспособным, и браузер выдает пользователю сообщение об ошибке: «Невозможно отобразить страницу».

3. Для сохранения в АВП информации, пока она необходима, используются механизмы, предоставляемые стандартом HTML5.

С появлением данного стандарта его механизмы встроены в современные браузеры для ПК и мобильных устройств. В частности, что важно для предлагаемого метода, стандартом предусмотрено наличие в браузерах постоянных локальных хранилищ: Local Storage [79] для введенных пользователем данных и Application Cache [56] для полученной с сервера информации и исходного кода программ. Использование в АВП других технологий локального сохранения данных средствами веб-браузера возможно без изменения архитектуры и метода функционирования АВП и может внедряться по мере появления и развития таких технологий.

Выбор технологий локального хранения, используемых в данной работе, обусловлен тем, что в настоящее время эти технологии стандарта HTML5 поддерживаются практически всеми браузерами для ПК (Chrome 8+, Firefox 3.5+, Safari 4+, Opera 10.6+, Internet Explorer 10+) и мобильных устройств (iOS Safari 3.2+, Android Browser 2.1+, Opera Mobile 11+, Internet Explorer Mobile 10+) [34, 35]. Это позволяет сделать вывод о возможности установки АВП на любое клиентское устройство, в том числе мобильное. Соответственно, **включение АВП в состав ГИС с использованием данных ДЗЗ обеспечит отказоустойчивую кроссплатформенность данной системы**, которая требуется для доступа к базе геоданных с различных клиентских устройств и, в первую очередь, мобильных.

Важно, что существующие решения на основе традиционного веб-приложения могут быть модернизированы до АВП добавлением разработанных клиентских и серверных компонентов, что обеспечивает преемственность развития имеющихся программных продуктов без необходимости их полной замены. Кроме того, предлагаемый в работе метод предусматривает возможность автоматической загрузки с сервера и развертывания на клиентских устройствах клиентской части АВП без необходимости каких-либо действий со стороны пользователя.

4. Управление локальным сохранением полученных с сервера данных на стороне клиента осуществляется на основе манифеста кэша (МК). МК, предусмотренный стандартом HTML5, является файлом настроек локального хранилища данных Application Cache. Содержит список файлов (элементов страниц) для сохранения на стороне клиента (рис. 1.8). Располагается на сервере АВП, загружается и сохраняется браузерами с поддержкой стандарта HTML5. Обеспечивает управление долговременным сохранением получаемых с сервера данных и клиентской части АВП.

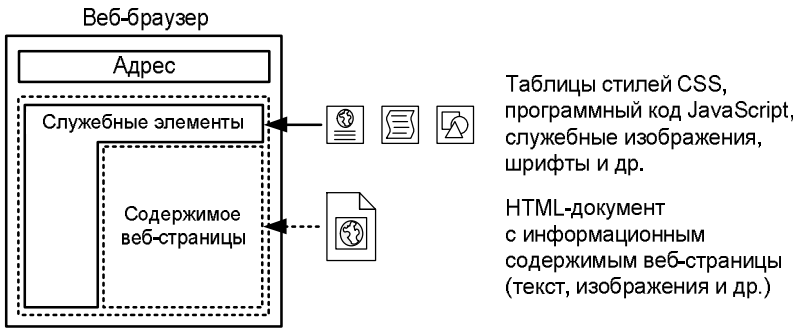


Рис. 1.8. Файловый состав элементов веб-страницы

МК запрашивается клиентом с сервера при каждом запросе страницы АВП. Содержит сведения о том, какие файлы нужно загрузить с сервера и сохранить в хранилище Application Cache, какие файлы нельзя сохранять, какие из сохраненных файлов использовать в качестве заменителей незагруженных несохраненных элементов страницы в случае потери соединения с сервером, а также дату и время момента актуализации данного списка. Запросы МК клиентом с сервера позволяют серверу управлять инициализацией клиентской части АВП и актуализацией локального хранилища полученных с сервера данных.

5. Исключение повторной передачи статической информации. Принципиальным моментом, определяющим функционирование АВП, является то, что в системах с веб-интерфейсом загружаемые клиентом данные разделены на множество небольших (единицы-десятки-сотни Кбайт) относительно их общего объема (единицы-десятки Мбайт) файлов, которые затем формируют отображаемую страницу на стороне клиента (как уже было сказано, в Интернете по статистике на 1 февраля 2015 одна страница состоит в среднем из 96 файлов общим объемом 1977 Кбайт [52]). Это делается, чтобы как можно раньше начать отображение страницы пользователю, не дожидаясь полной загрузки данных. При этом многие повторно передаваемые файлы являются элементами различных или всех страниц веб-приложения. Наличие большого числа повторяющихся элементов (примерно 46% [52]) на разных страницах одного веб-приложения позволяет при их долговременном сохранении на стороне клиента не запрашивать их повторно, чем уменьшить общее количество запросов к серверу, сократить время открытия страниц и объем трафика. Набор элементов, составляющий страницу, включает в себя: HTML-документ, генерируемый приложением на основе данных из БД, содержащий ссылки на остальные элементы; статические и динамические файлы, формирующие информационное содержимое страницы и ее стилевое оформление; программные компоненты, выполняющиеся на стороне клиента. Значительная часть этих элементов не меняется в течение долгого времени (имеют продолжительный период актуальности) и поэтому могут быть сохранены на стороне клиента до их изменения (по статистике могут быть сохранены те же 46% элементов страниц [52]).

В эти 46% входят: каскадные таблицы стилей (CSS – англ. Cascading Style Sheets), загружаемые шрифты и служебные изображения, отвечающие за формирование стилевого оформления любого веб-приложения (остаются неизменными для всех страниц этого веб-приложения и меняются только при его переделке для внедрения нового стилевого оформления); программные компоненты на языке JavaScript, выполняющиеся на стороне клиента (могут изредка меняться для добавления новых функций) – их состав на разных страницах одного веб-приложения может несколько отличаться в зависимости от содержимого конкретной страницы (например, визуальный редактор в форме ввода данных или инструменты управления картой, средства управления меню, валидаторы вводимых данных и прочие).

Зная период актуальности, можно долговременно сохранить с помощью АВП на стороне клиента и часть информационного содержимого. На примере Геопортала Роскосмоса таким сохраняемым содержимым может быть: неизменяемая часть каталога геоданных, содержащая перечень архивов снимков за прошедшие периоды; редко изменяемые элементы многослойной карты, такие как растровая базовая подложка низкого и среднего разрешения, векторные слои географических и тематических обозначений. Текстовые страницы могут быть сохранены полностью.

Сказанное отличает АВП от традиционных веб-приложений, где существующий сеансовый кэш сохраняет часть данных (элементов страниц) только в пределах текущего сеанса. Данное обстоятельство требует повторной передачи редко актуализируемых элементов страниц и, соответственно, не позволяет экономить трафик. Заметим, что это же обстоятельство не позволяет решить задачи обеспечения отказоустойчивости и автономной работы в рамках традиционных веб-приложений.

1.3.2 Базовые процессы метода

Основу метода функционирования клиент-серверных систем с использованием АВП составляют следующие рассматриваемые ниже базовые процессы:

- **отправка данных на сервер;**
- **получение данных от сервера;**
- **актуализация локального хранилища данных.**

При этом для обеспечения названных требований к АВП отправка данных на сервер делится на два принципиально отличающихся случая:

- отправку запросов на получение данных с сервера;
- отправку введенных пользователем данных на сервер.

Первое их отличие состоит в том, что запросы на получение данных имеют заданную структуру, а отправляемые данные могут содержать любую, в том числе, неструктурированную

информацию (например, комментарии, сообщения форумов, данные полей различных форм и другое). Второе их отличие состоит в том, как происходит работа с локальными хранилищами. Так, отправка запросов на получение данных включает в себя проверку наличия запрашиваемых данных, сохраненных в Application Cache, и запрос передается на сервер только при отсутствии данных на клиентском устройстве. Отправка же введенных пользователем данных включает в себя предварительное сохранение введенных данных в локальном хранилище Local Storage перед отправкой на сервер. Любой вид отправки данных на сервер может происходить в АВП как с перезагрузкой, так и без перезагрузки страницы.

Получение данных от сервера тоже делится на два случая по способу получения данных:

- с полной перезагрузкой страницы при получении запрошенных элементов (синхронная загрузка). Так загружаются страницы АВП;
- без перезагрузки страницы, но с изменением отдельных ее элементов при фоновом получении запрошенных элементов (асинхронная загрузка). Так загружаются служебные сообщения о состоянии сервера и подтверждения доставки фоновой отправки данных.

Здесь же заметим, что заявленная выше автономная работа с синхронизацией данных между клиентом и сервером включает в себя:

- непосредственно автономную работу с ранее сохраненными данными (режимы которой рассмотрены в главе 2) и использование загруженных с сервера программных средств для работы с этими данными;
- обновление ранее сохраненных данных на клиентском устройстве во время появления соединения с сервером при изменении их на сервере, что осуществляется с помощью разработанных программных средств и определяется как актуализация локального хранилища данных.

Рассмотрим базовые процессы подробнее.

Отправка запросов на получение данных с сервера. При вводе пользователем адреса страницы и при переходе по ссылкам на страницах АВП браузер направляет на сервер запросы только для получения элементов страниц, отсутствующих в локальном хранилище Application Cache, обновления устаревших сохраненных и для получения МК. Запросы для получения элементов страниц, уже присутствующих и актуальных в Application Cache, перенаправляются в хранилище Application Cache. Это позволяет уменьшить количество запросов к серверу.

При вводе адреса страницы пользователем браузер стандарта HTML5 проверяет наличие в хранилище Application Cache локально сохраненных данных и программного кода АВП для этого адреса. Если сохраненные данные для веденного адреса есть в хранилище, браузер запрашивает с сервера МК, сразу начинает отображение сохраненных данных и посылает запросы на сервер для получения несохраненных элементов отображаемой страницы, а также

для загрузки новых и проверки обновления уже имеющихся элементов нового МК, если он отличается от старого.

Отправка введенных пользователем данных на сервер. Перед отправкой на сервер введенные пользователем данные сначала сохраняются в хранилище Local Storage, затем отправляются на сервер, а после получения подтверждения доставки данных на сервер удаляются из хранилища.

В случае возникновения сбоя при отправке введенных данных на сервер они не пропадут, так как останутся сохраненными в Local Storage. При восстановлении соединения с сервером данные из Local Storage будут автоматически отправлены на сервер клиентской частью АВП, а после получения подтверждения их доставки будут удалены из хранилища.

Итак, в основе предлагаемого метода лежит буферизация данных в отведенной на клиентском устройстве памяти для размещения АВП, а его сутью является управление этими данными как в автономном режиме (англ. offline – без подключения к серверу) на стороне клиента, так и в режиме обмена информацией между клиентской и серверной частями. С учетом базирования АВП на стандарте HTML5 функции буфера/хранилища данных, как уже говорилось, выполняют объекты этого стандарта: Local Storage – для введенных пользователем данных; Application Cache – для полученной с сервера информации и исходного кода программ для насыщения клиентского устройства функциями, обеспечивающими манипулирование полученной информацией.

Получение данных от сервера при синхронной загрузке. При синхронной загрузке с сервера загружаются только элементы страниц, не сохраненные локально, МК и устаревшие сохраненные элементы страниц. Получение данных от сервера АВП бывает двух видов: начальная загрузка АВП (инициализация), когда происходит загрузка с сервера всех элементов страницы по указанному адресу, МК и перечисленного в нем начального набора данных АВП в локальное хранилище Application Cache; последующая загрузка инициализированным АВП с сервера при обращении к любой странице МК, несохраненных элементов этой страницы и обновленных элементов страниц, сохраненных в Application Cache. При работе АВП HTML-файлы посещаемых страниц, относящихся к АВП, автоматически добавляются в локальное хранилище, даже не будучи явно указанными в МК, что делает возможным их автономное использование.

При отсутствии или сбое соединения с сервером полностью инициализированное АВП переходит в автономный режим и использует ранее сохраненные в Application Cache загруженные с сервера данные. Для незагруженных элементов страниц используются ранее сохраненные перечисленные в МК файлы-заменители, служащие для информирования пользователя об отсутствии на странице недоступных в автономном режиме элементов.

В случае возникновения сбоя соединения до полной инициализации АВП возможны несколько вариантов ситуаций на разных этапах инициализации. А именно:

- браузер не успел загрузить даже первый HTML-файл страницы АВП. В таком случае инициализация АВП еще даже не могла начаться. Пользователю придется дождаться восстановления соединения и повторить попытку;
- браузер успел загрузить HTML-файл, но не успел загрузить связанный с ним МК. В таком случае загрузка элементов для сохранения еще не началась, но HTML-файл уже сохранен в локальном хранилище. Пользователю придется дождаться восстановления соединения и повторить попытку для загрузки МК со списком файлов для сохранения и самих этих файлов;
- браузер успел загрузить HTML-файл и связанный с ним список файлов для сохранения, но не успел загрузить другие элементы страницы и/или перечисленные в списке файлов для сохранения. В таком случае HTML-файл и часть загруженных элементов успели сохраниться в локальном хранилище, остальные элементы из списка файлов для локального сохранения будут загружены при следующем обращении по адресу любой страницы этого АВП. Если реализация браузера не позволяет ему запомнить, какие элементы для сохранения не были загружены, в предлагаемом решении присутствует программный актуализатор локального кэша, который запускает процедуру принудительного обновления отсутствующих или устаревших файлов, посылая клиенту новый список файлов для сохранения с более поздней отметкой даты и времени актуальности.

Получение данных от сервера при асинхронной загрузке. При асинхронной загрузке получаемые с сервера данные не кэшируются и сразу отображаются пользователю средствами клиентской части АВП в составе просматриваемой страницы. Отсутствие кэширования таких данных является необходимым, так как подтверждения доставки данных и уведомления о состоянии сервера меняются и должны всегда быть актуальными.

Актуализация локального хранилища данных. При каждом обращении к странице АВП происходит загрузка с сервера МК (рис. 1.9). Если очередной загруженный с сервера МК отличается от предыдущего датой и временем актуализации или набором файлов-элементов страниц, перечисленных в разделах МК, начинается процедура актуализации локального хранилища данных. Эта процедура включает в себя: удаление из локального хранилища Application Cache ранее сохраненных файлов, отсутствующих в списке сохраняемых файлов в новом МК; загрузку с сервера в Application Cache новых файлов, которые появились в новом МК; проверку наличия обновления на сервере для каждого из ранее сохраненных файлов, перечисленных в новом МК и загрузку только тех из них, которые были изменены с момента их

предыдущей загрузки. Проверка обновления конкретного сохраненного файла заключается в отправке запроса на сервер с датой и временем его сохранения. Если на сервере дата и время модификации файла с учетом часового пояса клиента и сервера раньше даты его сохранения на стороне клиента, то сервер выдает клиенту сообщение «файл не изменен». Если дата и время модификации файла на сервере позже даты и времени загрузки этого файла клиентом, то сервер в ответ на запрос посылает клиенту обновленный файл. При необходимости могут быть добавлены дополнительные этапы проверки данных при синхронизации клиента и сервера в зависимости от требований конкретной конфигурации ИС.

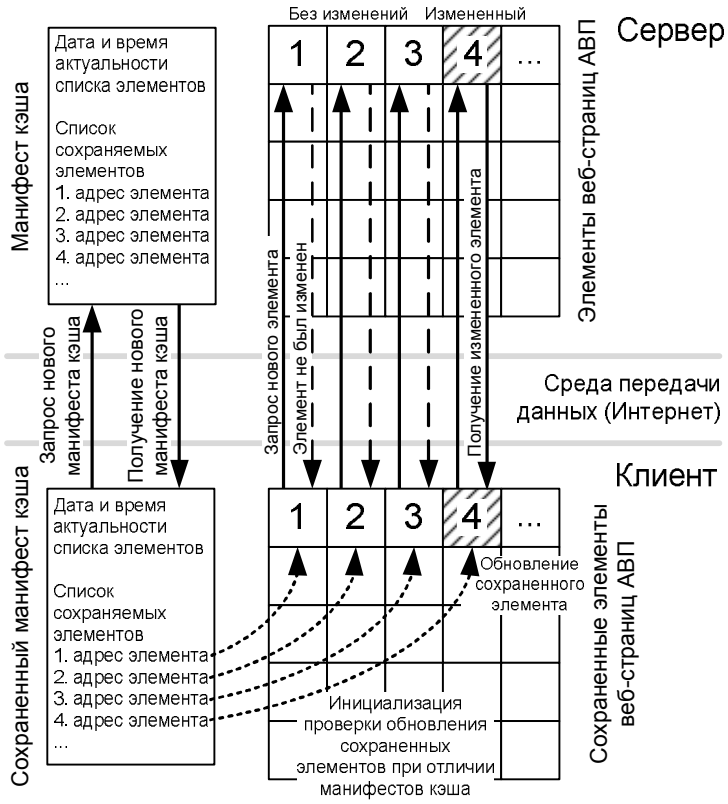


Рис. 1.9. Управление локальным сохранением полученных с сервера данных с помощью манифеста кэша

В случае возникновения сбоя соединения в процессе актуализации локального хранилища данных до того, как сохраненные обновляемые файлы АВП успели обновиться полностью, возможны несколько вариантов ситуаций на разных этапах обновления. А именно:

- браузер не успел скачать новый МК до возникновения сбоя. В таком случае при следующем обращении к любой странице этого АВП после восстановления соединения произойдет загрузка нового МК, на его основе будет установлено наличие обновлений и проведена загрузка измененных и новых файлов для сохранения;
- браузер успел скачать новый МК, но не все новые или измененные файлы, либо не успел проверить наличие обновления части файлов до возникновения сбоя. В таком случае

часть новых или обновленных файлов успели сохраниться в локальном хранилище, остальные файлы из списка файлов для локального сохранения нового МК будут загружены или проверены при следующем обращении по адресу любой страницы АВП. Если реализация браузера не позволяет ему запомнить, какие элементы для сохранения не были загружены или проверены, в предлагаемом решении присутствует актуализатор локального кэша, который запускает процедуру принудительного обновления отсутствующих или устаревших файлов, посылая клиенту новый список файлов для сохранения с более поздней отметкой даты времени актуальности.

1.4 Основные компоненты АВП

Локальные хранилища и МК, предоставляемые стандартом HTML5, сами по себе не обеспечивают выполнение функций, которые выше были возложены на АВП, но позволяют на их основе создать предлагаемый в настоящей работе программный комплекс АВП, который обеспечивает их выполнение.

Разработанный программный комплекс осуществляет процедуры, выполняющие базовые процессы. В рамках клиент-серверной организации осуществление процедур выполняется набором компонентов: разработанными клиентскими и серверными модулями и конфигурационным файлом локального хранилища полученных с сервера данных – МК.

Взаимосвязь процедур и процессов АВП показана в таблице 3.1. Архитектура и взаимодействие разработанных программных модулей АВП показаны на рис. 3.5.

Программная реализация АВП подробно рассмотрена в главе 3.

Ниже представлена принципиальная схема функционирования АВП (рис. 1.10).

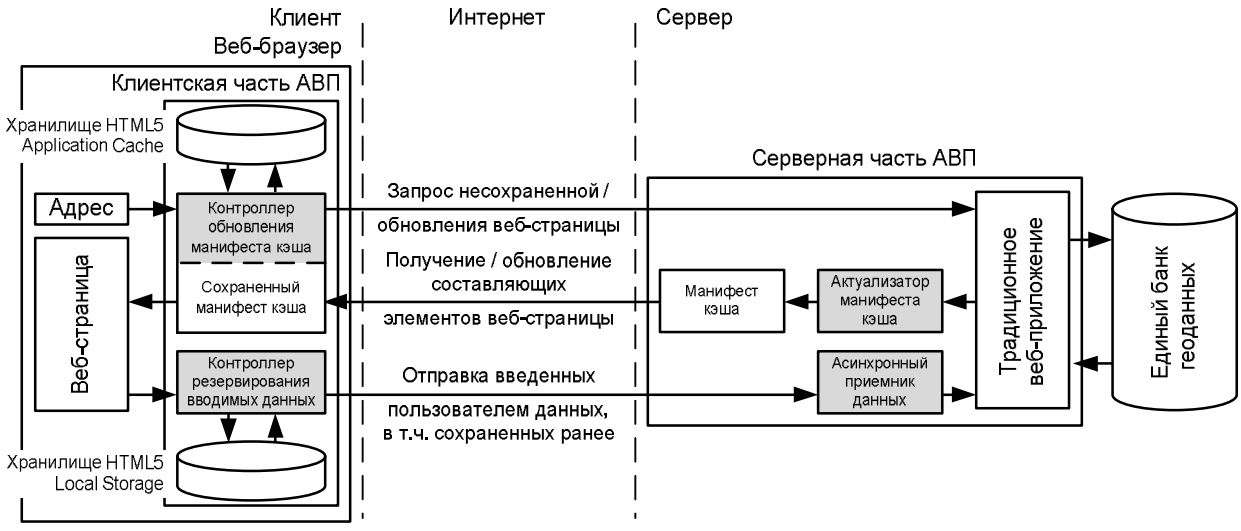


Рис. 1.10. Принципиальная схема АВП

На приведенной схеме помимо уже названных локальных хранилищ данных на стороне клиента присутствуют разработанные программные средства для управления этими данными, как в части сохранения, так и актуализации. Данные программные средства включают в себя клиентские и серверные модули, реализующие процедуры обеспечения отказоустойчивости и автономного режима работы клиент-серверной системы.

Обеспечение отказоустойчивого ввода данных пользователем, включающего в себя автоматическое резервирование и восстановление на сервер вводимых пользователем данных в АВП, осуществляется процедурами отправки на сервер с локальным сохранением вводимых данных и восстановления на сервер локально сохраненных данных. Данные процедуры реализуются следующими модулями:

- **клиентский контроллер резервирования вводимых пользователем данных** осуществляет:
  - сохранение вводимых пользователем данных в Local Storage с учетом пользовательского сеанса с авторизацией и контекста ввода данных;
  - отправку их на сервер (включая как штатную отправку при наличии соединения клиента с сервером, так и отправку ранее сохраненных данных при восстановлении соединения после сбоя);
  - получение подтверждения с сервера о доставке введенных данных;
  - удаление успешно доставленных на сервер данных из Local Storage;
- **серверный асинхронный приемник данных** осуществляет:
  - получение введенных пользователем данных, отправленных на сервер клиентским контроллером резервирования вводимых пользователем данных (включая как штатное получение данных при наличии соединения с сервером в рамках текущего пользовательского сеанса с авторизацией и известным контекстом ввода данных, так и получение ранее сохраненных данных при восстановлении соединения после сбоя с авторизацией пользователя и восстановлением контекста ввода данных);
  - запрос у актуализатора МК генерации нового МК, чтобы распространить изменения данных для всех клиентов, если введенные пользователем данные, отправленные на сервер, влияют на изменение закэшированных ресурсов АВП.

Вместе эти модули реализуют следующий алгоритм.

1. Пользователь вводит данные в форму на странице и нажимает кнопку «Отправить».

2. Клиентский контроллер резервирования вводимых пользователем данных сохраняет в хранилище Local Storage введенные пользователем данные, контекст ввода (указание, в какую форму и на какой странице они были введены) и состояние авторизации пользователя без пароля (в целях обеспечения безопасности АВП потребует повторной авторизации

пользователя при восстановлении после сбоя соединения), после чего отправляет введенные данные на сервер. В случае возникновения сбоя соединения пользователю выводится информационное сообщение о том, что введенные данные не были доставлены на сервер, сохранены локально и будут отправлены автоматически при восстановлении соединения. При сбое соединения в действие вступает клиентский модуль проверки доступности сервера, который отслеживает момент восстановлении соединения и инициирует повторную отправку сохраненных данных.

3. Серверный асинхронный приемник данных получает данные и передает их существующим средствам сохранения введенных пользователем данных традиционного веб-приложения, а после их сохранения посылает подтверждение клиентскому контроллеру резервирования.

4. Получив подтверждение успешной доставки введенных данных на сервер, клиентский контроллер резервирования удаляет соответствующую запись из хранилища. Отображаемая пользователю страница может измениться с учетом введенных данных. Если на сервер были успешно отправлены несколько сохраненных записей введенных данных из Local Storage, то пользователю будет выведено сообщение об этом.

Обеспечение автономной работы с данными, сокращение времени загрузки страниц ГИС и синхронизация локального хранилища сохраненных данных с сервером для поддержания актуальности данных осуществляется процедурой обновления данных в локальном хранилище Application Cache. Данная процедура реализуется следующими модулями:

- **манифест кэша (МК)** – управляет долговременным сохранением полученных с сервера данных на стороне клиента и отображением страниц в автономном режиме при помощи списка файлов для сохранения и правил для работы с несохраненными элементами страниц. Генерируется на сервере программно или вручную администратором сервера в зависимости от набора начальных данных АВП и решаемых задач. Передается клиенту при каждом запросе страниц АВП, чтобы инициировать загрузку и поддерживать актуальность набора данных АВП в Application Cache;
- **серверный актуализатор МК** – обновляет на сервере МК при изменении данных на сервере для автоматического обновления клиентской части АВП;
- **клиентский контроллер обновления МК –** проверяет состояние локального хранилища Application Cache, информирует пользователя о наличии обновления через модуль вывода информационных сообщений, вызывает отображение обновленных данных. При обнаружении неполной инициализации Application Cache или по команде пользователя запрашивает у серверного актуализатора МК новый МК для запуска обновления.

  Вместе эти модули реализуют следующий алгоритм.

1. При первом обращении браузера по адресу любой страницы АВП на клиентское устройство вместе с запрошенной страницей с сервера загружается МК и набор данных АВП для сохранения в хранилище Application Cache.

2. При последующих обращениях к любым страницам данного АВП с сервера загружаются только несохраненные или обновленные элементы страниц, а остальные элементы получаются из локального хранилища Application Cache.

3. При возникновении сбоев соединения или при отсутствии соединения с сервером АВП переходит в автономный режим и работает с ранее загруженными с сервера данными, сохраненными в Application Cache.

В процедурах АВП также участвуют вспомогательные модули, не показанные на схеме:

- **клиентский модуль вывода информационных сообщений пользователю** – информирует пользователя о текущем состоянии локальных хранилищ Local Storage и Application Cache, о возникновении сбоев соединения с сервером и действии других компонентов АВП;
- **клиентский модуль проверки доступности сервера** – проверяет, не восстановилось ли соединение после сбоя, периодически обращаясь к ответчику доступности сервера, чтобы дать сигнал другим модулям для выполнения предусмотренных в таком случае автоматических процедур;
- **серверный модуль – ответчик доступности сервера –** дает клиенту информацию о восстановлении соединения с сервером и о состоянии сервера (например, информацию о состоянии авторизации пользователя на сервере).

Итак, на основе изложенного можно выделить следующие две группы преимуществ АВП по сравнению с традиционным веб-приложением:

1. качественные (что традиционное веб-приложение не может обеспечить):
   - возможность автономной работы пользователей с полученными с сервера данными и автоматическая актуализация локально сохраненных данных;
   - возможность аварийного резервирования вводимых пользователем данных и автоматическая отправка сохраненных данных при возобновлении соединения с сервером.

   Подтверждение названных преимуществ представлено в третьей главе диссертации в виде результатов тестирования разработанного программного комплекса, доказывающих реализуемость названных возможностей АВП.

2. количественные, которые АВП получает за счет долговременного сохранения получаемой с сервера информации на стороне клиента и поддающиеся численному сравнению. К их числу относятся следующие характеристики:

- уменьшение количества запросов к серверу;
- уменьшение расхода трафика;
- уменьшение времени загрузки страниц;
- уменьшение суммарного времени ожидания ответа сервера;
- уменьшение времени использования канала связи.

Для подтверждения данных преимуществ в следующей главе диссертации разработаны аналитические модели, содержащие зависимости основных характеристик функционирования АВП от характеристик постоянных локальных кэшей клиентского устройства, а также для расчета тех же характеристик традиционных веб-приложений (рис. 1.11).

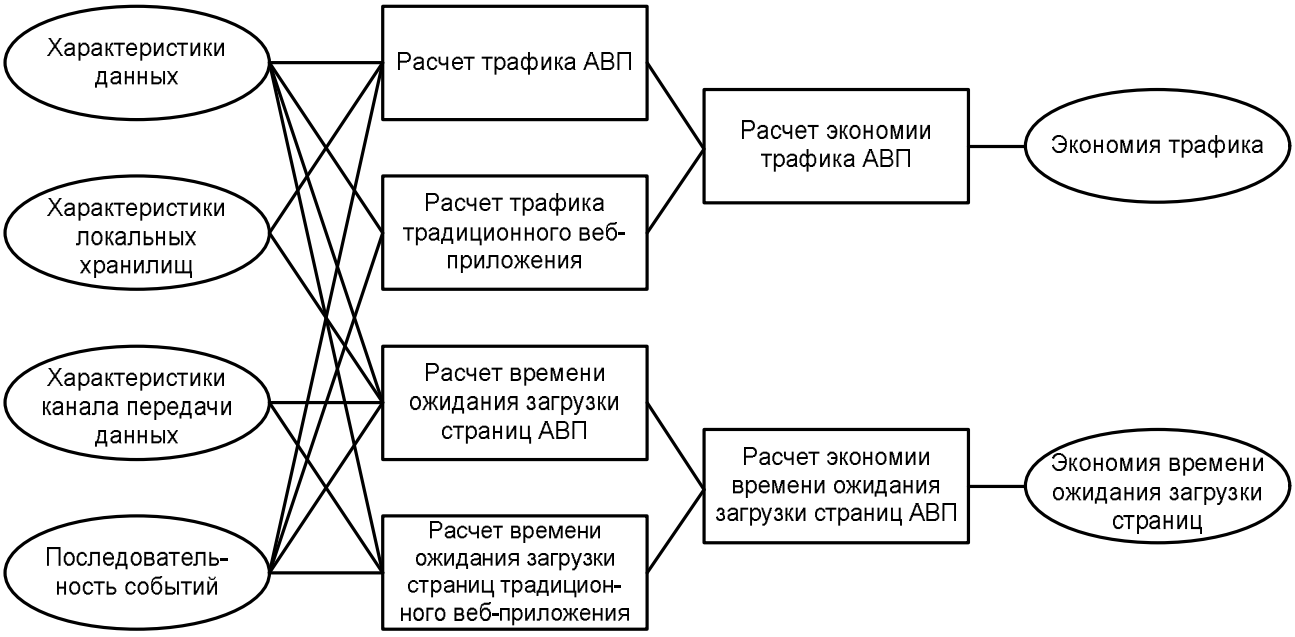


Рис. 1.11. Схема модели АВП и традиционного веб-приложения

Здесь же выполнено количественное сравнение названных характеристик АВП и традиционных веб-приложений. Рассмотрены конкретные примеры преимуществ АВП при работе с ГИС с использованием данных ДЗЗ.

ГЛАВА 2. МОДЕЛИ И АНАЛИЗ КОЛИЧЕСТВЕННЫХ ОЦЕНОК ХАРАКТЕРИСТИК АВП И ТРАДИЦИОННЫХ ВЕБ-ПРИЛОЖЕНИЙ

Согласно изложенному выше, в основе предложенного метода функционирования клиент-серверных систем с АВП лежит возможность создания хранилищ информации на клиентском устройстве, используя конструктивы стандарта HTML5. С целью получения количественных оценок преимуществ, которые предоставляет названная возможность, а также определения условий, максимизирующих эти преимущества, в настоящей главе построены соответствующие модели. На их основе и выполнен сопоставительный анализ АВП и традиционных веб-приложений, а также анализ влияния характеристик хранилищ на эффективность клиент-серверной системы в целом. За критерии эффективности приняты названные выше: количество запросов к серверу, объем загружаемого трафика, суммарное время ожидания ответов сервера, время использования каналов передачи данных и время загрузки страниц [6]. При выполнении количественного анализа для общего случая использовались статистические данные по структуре и взаимодействию с пользователем сайтов в сети Интернет [52]. Оценка работы АВП с геоданными осуществлена на основе решения практических задач.

2.1 Модели взаимодействия клиентов с сервером ГИС

Рассмотрим модели взаимодействия множества элементов системы «мобильный клиент – сервер ИС» при использовании традиционных веб-приложений и АВП для определения возможностей по повышению за счет использования АВП надежности функционирования данной системы и улучшения ряда параметров системы. Построенные модели справедливы для всех имеющихся мобильных и стационарных клиентов ГИС и охватывают такие параметры функционирования рассматриваемой системы как отказоустойчивость, количество запросов к серверу, объем загружаемого трафика, суммарное время ожидания ответа сервера, время использования каналов связи и время загрузки страниц. Для обеспечения функционирования АВП предложено несколько режимов работы локальных хранилищ вводимых пользователем данных и данных, полученных с сервера.

В описании моделей приняты следующие обозначения:

i – индекс последовательности получения или отправки клиентом сообщений;

j – индекс принимаемого или отправляемого файла или сообщения в рамках последовательности с индексом i;

j' – индекс принимаемого файла или сообщения в рамках последовательности получения подтверждения доставки на сервер введенных пользователем данных файла или сообщения с индексом j;

n – индекс сеанса обмена входящими и исходящими сообщениями клиента с сервером ИС, где $n \in N$, N – множество всех сеансов обмена сообщениями между клиентом и сервером. Сеансом будем считать произвольную мультипоследовательность (мультимножество последовательностей) получения клиентом данных с сервера и передачи клиентом данных на сервер, объединенных общей авторизацией клиента на сервере или происходящих в рамках одной серверной сессии без авторизации. Элементами такой мультипоследовательности могут быть запросы, файлы и сообщения;

m – индекс файла или страницы на сервере, которые могут быть автономно сохранены в постоянном кэше приложения АВП, где $m \in M$, M – множество всех файлов и страниц на сервере.

Введены операторы:

$\mathrm{size} : X \to Y$, где X – файл, Y – размер данного файла в килобайтах (Кбайт);

$\mathrm{len} : W \to V$, где W – конечная последовательность или мультипоследовательность, V – число элементов данной последовательности или мультипоследовательности.

Модель сетевого взаимодействия веб-клиентов и сервера ИС содержит следующие элементы:

S – сервер ИС;

K – мобильный клиент;

C – канал связи между клиентом и сервером $C = \{(K, S), V_C, t_C\}$ (именно здесь возникают сбои);

V_C – пропускная способность канала связи (скорость передачи данных) C. Будем считать, что при отсутствии соединения между клиентом и сервером $V_C = 0$, а при недопустимом снижении скорости, когда соединение есть, но работать все равно невозможно, $V_C \to 0$;

$Q_{ij}^{получ}$ – запрос на получение блока данных $P_{ij}^{получ}$, отправляемый клиентом K серверу S;

$Q^{ман}$ – запрос на получение файла манифеста кэша $M^{кэш}$, отправляемый клиентом K серверу S;

t_C – время отклика сервера – время между отправкой по каналу C запросов $Q_{ij}^{получ}$ или $Q^{ман}$ клиентом K и получением первого байта ответа сервера S, а также интервал между отправкой по каналу C последнего байта сообщения $P_{ij}^{отпр}$ на сервер и получением первого байта подтверждения от сервера. Время t_C является характеристикой канала C. Так как запросы $Q_{ij}^{получ}$ клиента к серверу на получение данных малы по размеру (не могут превышать максимального размера в 2 Кбайт [32, 61, 62, 69] из-за ограничений браузеров, а на практике они значительно меньше 500 байт), будем считать время доставки любого запроса клиентом к серверу и получения первого байта ответа сервера постоянным для канала связи C и равным t_C, что соответствует проведенным наблюдениям. Одним из факторов, характеризующих недопустимое качество соединения в канале C, является превышение времени между отправкой запроса и получением ответа установленного браузером клиента таймаута $t_C > t_C^{\max}$. Обычно такое происходит в случаях, когда из-за помех или сбоев в канале $t_C \to \infty$. Таймаут в браузерах может достигать $t_C^{\max} = 5$ минут и более [53] в зависимости от модели браузера и его номера версии, а также изменяться в настройках самим пользователем. На практике в мобильных устройствах из-за учета возможных нарушений соединения, когда связь не возобновится и ждать нет смысла, $t_C^{\max}$ уменьшен производителем до десятков секунд. Будем считать, что сервер ИС моментально отдает первый байт ответа после получения запроса, то есть t_C характеризует только канал, а не быстродействие сервера. Также будем считать, что в канале C время в пути запроса от клиента к серверу составляет $\frac{t_C}{2}$, обратный путь первого байта ответа от сервера к клиенту тоже составляет $\frac{t_C}{2}$;

$U = \{u_i\}$ – конечная мультипоследовательность передачи клиентом K на сервер S введенных пользователем данных;

$u_i = \{P_{ij}^{отпр}\}$ – каждая отдельная последовательность передачи клиентом K на сервер S введенных пользователем данных;

$P_{ij}^{отпр}$ – каждый отдельный неделимый блок данных, передаваемых на сервер в рамках данной последовательности передачи введенных пользователем данных u_i, $P_{ij}^{отпр} \in \{u_i \mid u_i \in U\}$, клиента K на сервер S;

$t_{ij}^{отпр}$ – время отправки на сервер данных $P_{ij}^{отпр}$, $t_{ij}^{отпр} = \dfrac{\text{size}(P_{ij}^{отпр})}{V_C}$;

$D = \{d_i\}$ – конечная мультипоследовательность получения данных клиентом K с сервера S;

$d_i = \{P_{ij}^{получ}\}$ – каждая отдельная последовательность получения данных клиентом K с сервера;

$P_{ij}^{получ}$ – каждый отдельный неделимый блок данных, получаемых с сервера в рамках данной последовательности получения данных d_i, $P_{ij}^{получ} \in \{d_i \mid d_i \in D\}$, клиента K с сервера S;

$t_{ij}^{получ}$ – время получения с сервера данных $P_{ij}^{получ}$, $t_{ij}^{получ} = \dfrac{\text{size}(P_{ij}^{получ})}{V_C}$;

G_n – каждый отдельный сеанс соединения клиента K с сервером S, включающий в себя последовательности получения и отправки данных, $G_n \supset \{\{u_i\} \in U \cup \{d_i\} \in D\}$.

Рассмотрим сначала модель взаимодействия традиционного мобильного веб-клиента с сервером клиент-серверной ИС (рис. 2.1).

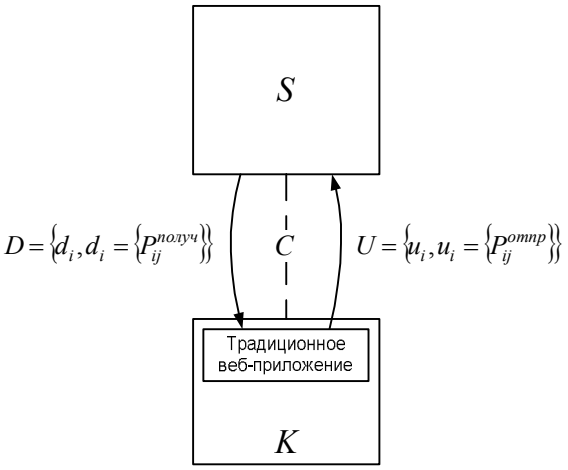


Рис. 2.1. Модель взаимодействия традиционного мобильного веб-клиента с сервером ИС

Каждый веб-клиент характеризуется общими параметрами, и его модель выглядит следующим образом:

$$K = \{E, A, R, C^{сеанс}, k^{парал}\},$$

где $E = \{E^{текущ}, E^{ист}, E^{шифр}, E^{кэш}\}$ – набор характеристик, относящихся к отображаемой странице:

$E^{текущ}$ – текущая страница ИС. Указание текущей страницы необходимо, поскольку отображение страниц ИС является основополагающей функцией веб-клиента ИС даже в случае,

если отображаемая страница содержит не данные для просмотра, а форму для ввода данных пользователем для отправки на сервер ИС. В терминах модели сетевого взаимодействия между клиентом и сервером $E^{текущ} = d_i$;

$E^{ист}$ – история посещенных страниц в данном сеансе G_n (начиная с первой) для обеспечения возможности обратного перехода между посещенными страницами с частичным использованием обычного сеансового кэша $C^{сеанс}$;

$E^{шифр}$ – указание на использование шифрования в ходе сеанса G_n. Необходимо для обеспечения защиты передачи данных между клиентом и сервером ИС по протоколам SSL/TLS,

$E^{кэш}$ – указание на возможность и правила кэширования данных $P_{ij}^{получ}$ в рамках текущего сеанса G_n. Необходимо для разрешения загрузки файлов и сообщений $P_{ij}^{получ}$ в обычный сеансовый кэш $C^{сеанс}$, его очистки или обновления содержимого.

A – множество действий пользователя (вход на страницу, авторизация в ИС, выход из ИС, ввод данных, переход по ссылкам и др.). Необходимо для получения команд осуществления перехода между состояниями системы.

R – статус результата действия пользователя. Требуется для вывода пользователю служебных сообщений в случае, если ожидаемый результат действия пользователя не получен.

$C^{сеанс}$ – сеансовый кэш. Используется для ускорения загрузки страниц ИС в рамках конкретного сеанса G_n благодаря локальному сохранению на устройстве пользователя части данных $P_{ij}^{получ}$, статических файлов. Эффективен при кэшировании изображений. Сеансовый кэш обладает существенными недостатками:

- требует наличия активного подключения к серверу,
- проверяет наличие обновлений всех хранящихся данных каждый сеанс G_n, при этом требует синхронизации времени между клиентом и сервером, что может быть недоступно,
- не позволяет клиенту отображать сохраненную информацию при отсутствии соединения с сервером,
- управляется HTTP-заголовками, но не все браузеры трактуют их одинаково, поэтому полагаться на такое управление нельзя,
- не кэширует данные, полученные по протоколу HTTPS,
- при закрытии браузера может быть автоматически удален.

$k^{парал}$ – количество одновременных соединений браузера клиента с сервером. Браузер клиента открывает параллельные соединения, начиная со второго загружаемого элемента в каждой последовательности d_i, так как в первом элементе загружается HTML-содержимое со ссылками на остальные элементы, которые должны быть загружены. В существующих в настоящее время браузерах $2 \le k^{парал} \le 8$ [71, 72, 73]. Будем считать, что $k^{парал}$ является постоянной характеристикой браузера и не меняется в каждой последовательности d_i. Будем также считать, что браузер клиента K открывает новые соединения с сервером группами по $k^{парал}$ штук после получения всех данных в ранее открытых параллельных соединениях, до тех пор, пока число оставшихся незагруженных файлов или сообщений $k_i^{незагр}$ в последовательности d_i такое, что $k^{парал} \le k_i^{незагр}$.

При $k^{парал} > k_i^{незагр}$ браузер открывает оставшиеся $k_i^{незагр}$ соединений. Такой порядок работы браузера показан на рис. 2.2.

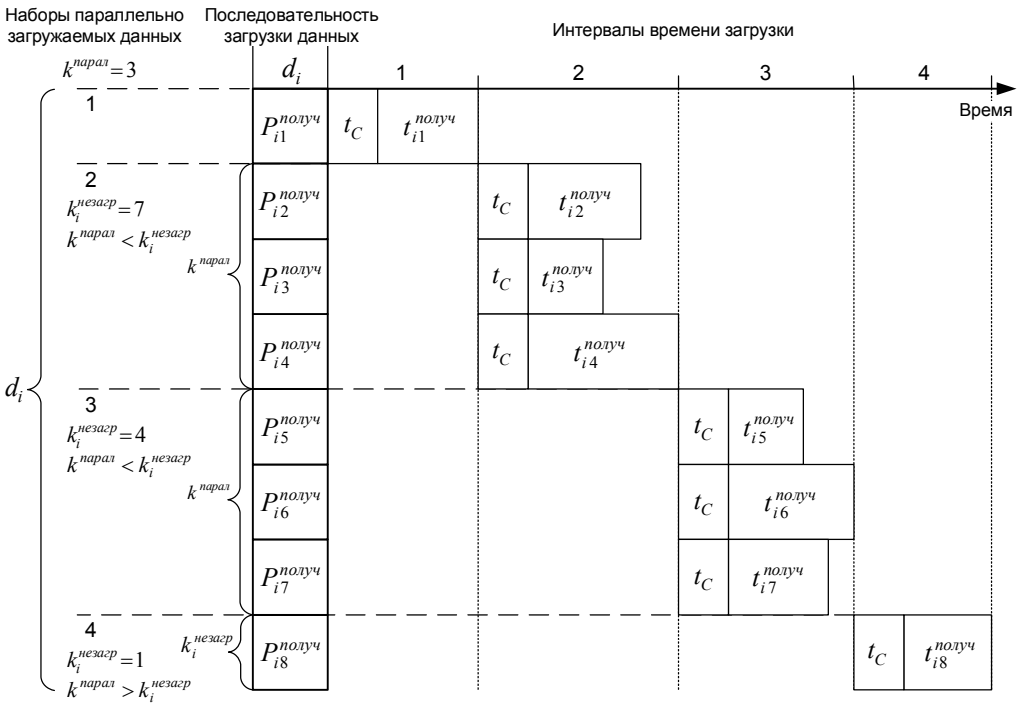


Рис. 2.2. Разделение последовательности d_i на итерации в зависимости от $k^{парал}$ и количества незагруженных файлов или сообщений $k_i^{незагр}$

Данные действия браузера направлены на снижение продолжительности времени получения клиентом информации с сервера и должны быть учтены при его расчете.

В отличие от модели традиционного веб-клиента предлагаемая модель веб-клиента АВП

$$K = \left\{E, A, R, C^{сеанс}, k^{парал}, C^{пост.дан}, M^{кэш}, C^{пост.прил}\right\}$$

включает в себя дополнительные библиотеки для обеспечения штатной автономной работы и резервирования вводимых пользователем данных и поэтому имеет дополнительные параметры.

$C^{пост.дан}$ – кэш для постоянного локального хранения введенных пользователем данных $P_{ij}^{отпр}$, которые не удалось отправить на сервер S, $C^{пост.дан} = \left\{P_{ij}^{отпр}\right\}$. Этот кэш необходим для организации отказоустойчивого резервирования вводимых пользователем данных с целью предотвращения их потери при сбоях соединения с сервером S и последующей автоматической отправки на сервер.

$M^{кэш}$ – манифест кэша. Управляет постоянным хранением на стороне клиента K предварительно заданного списка L файлов и страниц (сообщений) данных $P_m^{сохр}, L = \left\{P_m^{сохр}\right\}$, получаемых с сервера S, актуальных по состоянию на определенный момент времени на сервере $T^{акт.серв}$ (может быть индивидуальным для каждого клиента). $M^{кэш}$ содержит список кэшируемых файлов $P_m^{сохр}$, отметку времени $T^{акт.серв}$ на сервере и другую информацию для управления кэшированием. Манифест кэша необходим для управления $C^{пост.прил}$ с целью поддержания в актуальном состоянии данных, содержащихся локально на устройстве пользователя и недопущения их рассинхронизации с данными на сервере S. На стороне сервера существует возможность использования отдельных манифестов кэша для управления постоянным хранилищем данных приложения $C^{пост.прил}$ для каждого клиента K, а также для различных состояний одного и того же клиента. $M^{кэш}$ не является одним из $P_{ij}^{получ}$ и, соответственно, не является частью последовательности d_i. $M^{кэш}$ загружается АВП дополнительно к каждой последовательности d_i.

t_M – время получения с сервера файла манифеста кэша $M^{кэш}$, $t_M = \dfrac{\text{size}(M^{кэш})}{V_C}$.

$C^{пост.прил}$ – кэш для постоянного локального хранения данных $P_{ij}^{получ}$, полученных с сервера S, актуальных на момент времени $T^{акт.кэш}$. Этот кэш необходим для организации автономной работы веб-клиента с ранее загруженными с сервера данными при сбоях или в отсутствии соединения с сервером S. Управляется манифестом кэша $M^{кэш}$, $C^{пост.прил} = \left\{P_{ij}^{получ}, T^{акт.кэш} \mid P_{ij}^{получ} = P_m^{сохр}, T^{акт.кэш} = T^{акт.серв}\right\}$, где $T^{акт.кэш}$ – момент

времени на сервере, который был указан в предыдущем загруженном манифесте кэша $M^{кэш}$, $T^{акт.серв}$ – момент времени на сервере, который указан в новом загруженном манифесте кэша $M^{кэш}$. Равенство $T^{акт.кэш} = T^{акт.серв}$ является условием актуальности данных, сохраненных в $C^{пост.прил}$, в противном случае начинается проверка обновления и загрузка с сервера измененных $P_m^{сохр}$, перечисленных в новом $M^{кэш}$.

Благодаря использованию $C^{пост.прил}$ часть файлов данных $d_i^{сохр} \subset C^{пост.прил}$ начинает подаваться клиенту в рамках последовательностей получения данных $d_i^{сохр} \subset d_i$ не с сервера, а из $C^{пост.прил}$. Для разных последовательностей d_i часть данных $d_i^{сохр}$, подаваемых из кэша $C^{пост.прил}$, может являться произвольным подмножеством кэша $C^{пост.прил}$. Равенство $d_i^{сохр} = d_i$ означает, что все данные последовательности d_i доступны автономно.

В результате появления двух дополнительных библиотек на стороне веб-клиента для резервирования вводимых пользователем данных и для обеспечения возможности автономной работы модель его взаимодействия с сервером меняется, вследствие чего радикально возрастает его отказоустойчивость по сравнению с традиционным веб-клиентом.

Рассмотрим взаимодействие предлагаемого автономного отказоустойчивого мобильного веб-клиента с сервером клиент-серверной ИС (рис. 2.3). Здесь $D = \left\{ d_i, d_i = \left\{ P_{ij}^{получ} \right\} \middle| P_{ij}^{получ} \notin C^{пост.прил}, T^{акт.кэш} = T^{акт.серв} \right\}$. При условии актуальности содержимого постоянного кэша приложения веб-клиента с сервера загружаются только те данные, которые не находятся в $C^{пост.прил}$.

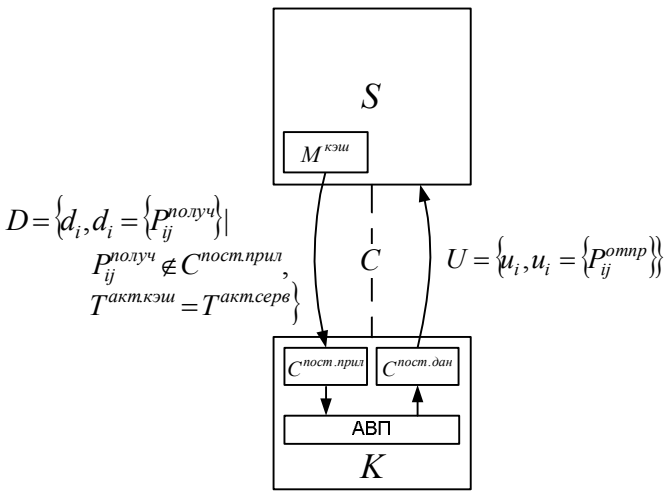


Рис. 2.3. Модель взаимодействия автономного отказоустойчивого мобильного веб-клиента с сервером ИС

Модель на рис. 2.3 учитывает влияние ненадежности каналов передачи данных C между клиентом K и сервером S на отправляемые веб-клиентами на сервер данными U и получаемыми ими с сервера данными D, предусматривая возможность резервирования вводимых пользователем данных в $C^{пост.дан}$ и автономной работы каждого мобильного клиента K с данными, полученными с сервера и хранящимися в $C^{пост.прил}$.

Предлагаемое решение позволяет:

- полностью исключить потери вводимых пользователем данных U за счет полного резервирования передаваемых данных в $C^{пост.дан} = \left\{P_{ij}^{отпр}\right\}$ (рис. 2.4) и удаления из кэша отправленных на сервер данных только после подтверждения их получения сервером;
- за счет предварительной загрузки в локальный кэш получаемых данных основной необходимой информации и долговременного кэширования получаемых с сервера данных в $C^{пост.прил} = \left\{P_{ij}^{получ}, T^{акт.кэш} \mid P_{ij}^{получ} = P_m^{сохр}, T^{акт.кэш} = T^{акт.серв}\right\}$ (рис. 2.5, 2.6):
  - уменьшить количество запросов к серверу;
  - снизить расход сетевого трафика клиентом;
  - снизить загрузку каналов операторов связи;
  - ускорить загрузку получаемых страниц с сервера;
  - уменьшить влияние на удобство работы низкой скорости соединения с сервером;
  - уменьшить расход энергии батареи мобильным клиентом [43, 65];
  - уменьшить негативное влияние возможных потерь получаемых с сервера данных D;
- обеспечить возможность полностью автономной работы для некоторых особо важных наборов данных за счет полного предварительного локального кэширования в $C^{пост.прил} = \left\{d_i, T^{акт.кэш} \mid d_i = \left\{P_m^{сохр}\right\}, T^{акт.кэш} = T^{акт.серв}\right\}$ (рис. 2.7).

Для обеспечения предлагаемых функций АВП постоянные кэши $C^{пост.дан}$ и $C^{пост.прил}$ имеют несколько режимов работы.

1. Резервирование пользовательского ввода. Для обеспечения резервирования вводимых пользователем данных все $\left\{P_{ij}^{отпр}\right\}$ сохраняются в $C^{пост.дан}$ перед отправкой на сервер и удаляются оттуда после получения подтверждения сервера о получении. В отсутствие

соединения с сервером в кэше вводимых пользователем данных хранятся все неотправленные $\left\{P_{ij}^{отпр}\right\}$ (рис. 2.4).

Рис. 2.4. Резервирование вводимых пользователем данных $P_{ij}^{отпр}$ в $C^{пост.дан}$

2. Минимальный начальный кэш. Режим работы $C^{пост.прил}$ для экономии трафика и ускорения загрузки АВП при минимальном начальном объеме данных на стороне клиента. Для экономии трафика и ускорения загрузки страниц ИС в $C^{пост.прил}$ предварительно при первом запуске загружаются $\left\{P_{ij}^{получ}\right\}$, которые используются во всех последовательностях d_i, $C^{пост.прил} \subset d_i$. В ходе работы с ИС $C^{пост.прил}$ может пополняться дополнительными $P_{ij}^{получ}$, что управляется $M^{кэш}$ с сервера (рис. 2.5);

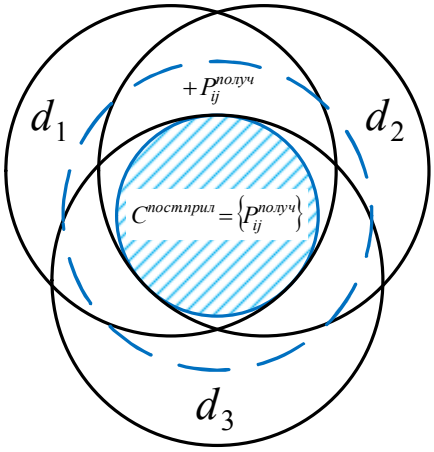


Рис. 2.5. Режим работы АВП с минимальным начальным объемом локально сохраненных данных $C^{пост.прил} \subset d_i$

3. Универсальный набор данных. Режим работы $C^{пост.прил}$ с большим универсальным набором автономных элементов для экономии трафика и ускорения загрузки разных страниц АВП, $C^{пост.прил} \cup d_i$. Данный режим предназначен для одновременного обеспечения функционирования нескольких клиентских приложений в рамках одной системы. В ходе

работы с ИС $C^{пост.прил}$ может пополняться дополнительными $P_{ij}^{получ}$, что управляется $M^{кэш}$ с сервера (рис. 2.6);

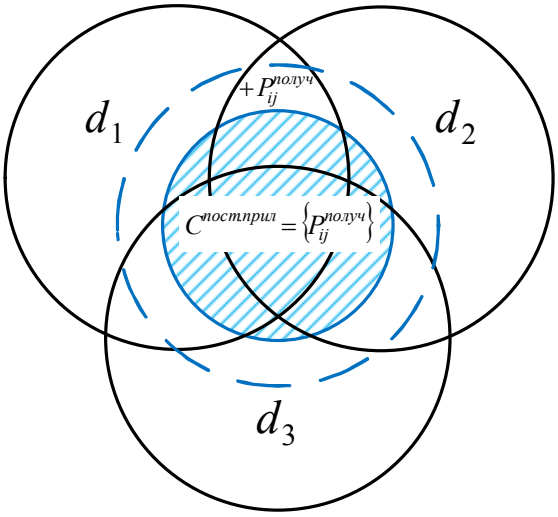


Рис. 2.6. Режим работы АВП с универсальным набором локально сохраненных данных $C^{пост.прил} \cup d_i$

При работе в режимах 2 и 3 после каждой последовательности получения данных с сервера кэш $C^{пост.прил}$ пополняется HTML-файлами и сообщениями, полученными с сервера в данной последовательности, и поэтому начинает соответствовать все большему числу запросов клиента к серверу. Варианты данного функционирования управляются $M^{кэш}$ с сервера ИС.

4. Автономный режим. Режим работы $C^{пост.прил}$ для обеспечения функционирования АВП без подключения к серверу. Для работы АВП без подключения к серверу предварительно при первом запуске в $C^{пост.прил}$ загружаются все необходимые данные тех последовательностей $\{d_i\}$, которые должны быть доступны автономно (рис. 2.7).

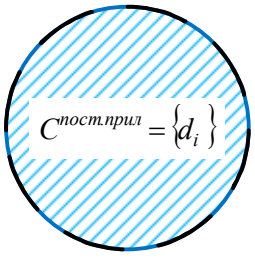


Рис. 2.7. Автономный режим работы АВП с хранением данных автономных последовательностей $\{d_i\}$ в $C^{пост.прил}$

Для работы перечисленных режимов функционирования кэшей $C^{пост.дан}$ и $C^{пост.прил}$ постоянные хранилища должны иметь определенный минимальный размер доступного пространства.

Для работы АВП в режиме 1 – резервирование пользовательского ввода – $\text{size}(C^{пост.дан}) \geq 1$ Мбайт, так как данные, вводимые пользователями вручную, не превышают объема в 1 Мбайт (для наглядности, в 1 Мбайт помещается более 500000 знаков в кодировке UTF-8, где на одну букву используется 2 байта). Данные, которые превышают объем в 1 Мбайт, как правило, копируются из файла, уже сохраненного на устройстве пользователя, а значит, их дополнительное резервирование не требуется.

Для работы в режимах 2 – минимальный начальный кэш, и 3 – универсальный набор данных – $\text{size}(C^{пост.прил}) \geq \sum_{P_m^{сохр} \in L} \text{size}(P_m^{сохр})$, чтобы туда могли поместиться дополнительные $P_{ij}^{получ}$, не указанные напрямую в $M^{кэш}$, но относящиеся к данному АВП, которые загружаются клиентом при переходе пользователя между страницами. Обычно, занимаемый объем минимального начального кэша для режима 2 для одного веб-приложения составляет 500-3500 Кбайт и увеличивается по мере использования веб-приложения, пока есть доступное место или не сохранено все, что могло быть сохранено. Если пользователь должен работать с несколькими разными веб-приложениями, то объем необходимого универсального набора данных для режима 3 в худшем случае равен произведению начального объема кэша для режима 2 на количество независимых приложений.

Для работы АВП в режиме 4 – автономный режим – $\text{size}(C^{пост.прил}) = \sum_{P_m^{сохр} \in L} \text{size}(P_m^{сохр})$, так как все данные, предлагаемые манифестом кэша, должны быть размещены в $C^{пост.прил}$, чтобы быть доступными автономно. Для полноценной автономной работы с мобильным устройством на период отсутствия подключения к серверу клиенту может потребоваться до 1 Гбайта данных (например, в случае с автономным использованием данных ДЗЗ или нескольких интерактивных руководств с обилием графической информации).

На практике большинство современных мобильных устройств располагает свободным пространством, исчисляемым гигабайтами, и объем их памяти продолжает расти. В работе предполагается, что объем доступной памяти значительно больше необходимого для территориально локализованных задач. Например, в объем памяти 4 Гигабайта войдет 900 км^2 растровых данных ДЗЗ уровней детализации с 14 по 19 включительно (суммарно) и программный код АВП (таблица 1.2).

С течением времени АВП смогут загружать все большие объемы данных на устройства пользователей. Однако наличие свободного места не означает его бесконтрольное использование. Пользователь всегда может посмотреть, сколько места занимает АВП на его устройстве, и может удалить данные АВП самостоятельно в любой момент, если ему понадобится свободное пространство. Перед инициализацией АВП проверяет наличие необходимого свободного места на устройстве пользователя и выдает сообщение о результатах проверки в случае, если места недостаточно.

Состав и объем данных, подлежащих автономному сохранению на стороне клиента, определяется администратором ИС путем редактирования файлов манифеста кэша на сервере различными доступными способами в зависимости от известных:

- типовых элементов страниц, необходимых клиенту для всех страниц ИС (АВП, режим 2);
- типовых элементов страниц, необходимых клиенту для различных часто используемых страниц ИС (АВП, режим 3);
- заданных групповых ролей пользователей ИС (АВП, режимы 3 и 4);
- потребностей различных пользователей ИС в автономном доступе к страницам ИС (АВП, режим 4)

или самими пользователями в зависимости от прав доступа и текущих потребностей (АВП, режим 4) путем внесения желаемых страниц и связанных элементов в индивидуальные манифесты кэша средствами ИС с помощью переключателя автономной доступности страницы или целого кластера страниц ИС, размещенного на конкретной странице.

Список файлов, подлежащих автономному сохранению на стороне клиента, отражается в файле манифеста кэша, индивидуальном для конкретного пользователя или общим для определенной группы пользователей.

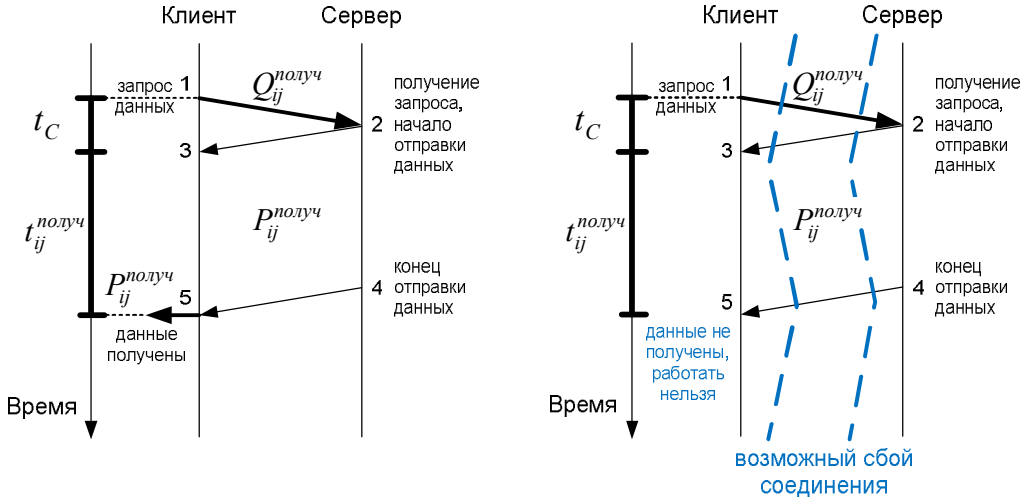
Таким образом, в зависимости от назначения АВП имеет 4 режима функционирования кэшей $C^{пост.дан}$ и $C^{пост.прил}$ с разным объемом сохраняемых локально данных для обеспечения отказоустойчивости, снижения количества запросов к серверу, экономии трафика, уменьшения суммарного времени ожидания ответа сервера, времени использования каналов связи и времени загрузки страниц.

Для оценки возможностей по улучшению названных параметров далее проводится аналитическое исследование моделей, сравниваются основные показатели функционирования традиционного веб-приложения и АВП.

2.2 Традиционное веб-приложение

Проанализируем на основе представленной модели традиционного веб-приложения количество его запросов к серверу, объем трафика, суммарное время ожидания ответа сервера, время использования каналов связи и время загрузки страниц.

Работа клиентской части традиционного веб-приложения состоит из загрузки данных с сервера (рис. 2.8) и отправки введенных пользователем данных на сервер (рис. 2.9).



а) **б)**

Рис. 2.8. Работа традиционного веб-приложения при загрузке данных с сервера: а) в штатном режиме, б) в случае сбоя соединения

Загрузка данных с сервера в штатном режиме показана на рис. 2.8а и осуществляется в следующем порядке:

1. Клиент посылает запрос $Q_{ij}^{получ}$ к серверу.
2. Сервер получает запрос от клиента и начинает отправлять в ответ данные $P_{ij}^{получ}$.
3. Через интервал времени t_C первый байт отправленных сервером данных $P_{ij}^{получ}$ достигает клиента.
4. Сервер заканчивает отправку данных $P_{ij}^{получ}$ клиенту.
5. Через интервал времени $t_{ij}^{получ}$ данные $P_{ij}^{получ}$ полностью получены клиентом.

Так как любая страница состоит из множества элементов (таблицы стилей CSS, программы на языке JavaScript, различные изображения и др.), то до ее отображения пользователю все эти элементы последовательно загружаются. Это является итерационным процессом. Процедура повторяется, пока все элементы страницы не будут загружены.

В случае сбоя соединения с сервером клиент не получает либо уже первого байта данных $P_{ij}^{получ}$ в точке 3, либо полные данные в точке 5 (рис. 2.8б). В этом случае дальнейшая работа с традиционным веб-приложением невозможна.

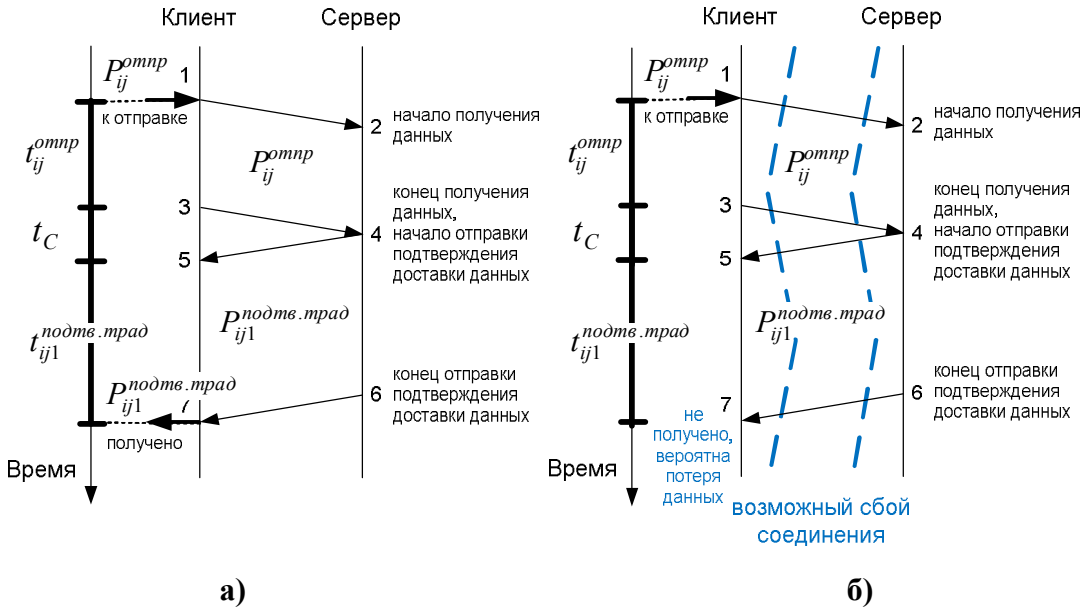


Рис. 2.9. Работа традиционного веб-приложения при отправке данных на сервер: а) в штатном режиме, б) в случае сбоя соединения

Отправка введенных пользователем данных на сервер в штатном режиме показана на рис. 2.9а и осуществляется в следующем порядке:

1. Клиент посылает введенные данные $P_{ij}^{отпр}$ на сервер.
2. Сервер получает первый байт данных $P_{ij}^{отпр}$.
3. Через интервал времени $t_{ij}^{отпр}$ клиент полностью отправил данные $P_{ij}^{отпр}$ на сервер.
4. Сервер заканчивает получение данных $P_{ij}^{отпр}$ от клиента и начинает отправку данных $P_{ij1}^{подтв.трад}$ клиенту для подтверждения получения от него данных.
5. Через интервал времени t_C клиент получает первый байт данных $P_{ij1}^{подтв.трад}$.
6. Сервер заканчивает отправку данных $P_{ij1}^{подтв.трад}$ клиенту.
7. Через интервал времени $t_{ij1}^{подтв.трад}$ данные $P_{ij1}^{подтв.трад}$ полностью получены клиентом.

В случае сбоя соединения с сервером клиент не получает либо уже первого байта данных подтверждения $P_{ij1}^{подтв.трад}$ в точке 5, либо все подтверждение в точке 7 (рис. 2.9б). В этом случае клиент не может сделать никаких предположений о результатах доставки отправленных данных на сервер и, в большинстве случаев, не может восстановить у себя отправленные данные. Если данные $P_{ij}^{отпр}$ не успели дойти до сервера до возникновения сбоя соединения, то они будут потеряны.

2.2.1 Сетевой трафик традиционного веб-приложения

Сетевой трафик $D^{трад}$ традиционного веб-приложения при загрузке данных с севера выражается формулой

$$D^{трад} = \sum_{i=1}^{\text{len}(D)} \sum_{j=1}^{\text{len}(d_i)} \text{size}(P_{ij}^{получ}), \quad (2.2.1)$$

где $\text{len}(D)$ – количество последовательностей d_i получения клиентом данных с сервера, $\text{len}(d_i)$ – количество страниц, файлов и сообщений, получаемых клиентом с сервера в последовательности d_i, $\text{size}(P_{ij}^{получ})$ – размер в Кбайтах j-й страницы, файла или сообщения в последовательности d_i.

Сетевой трафик при отправке данных на сервер рассматривается в совокупности с подтверждением о доставке отправленных данных, получаемым с сервера, поскольку в противном случае было бы непонятно, какие сообщения считать доставленными, а какие – нет.

У традиционного веб-приложения трафик при отправке данных на сервер $U^{трад}$ выражается формулой

$$U^{трад} = \sum_{i=1}^{\text{len}(U)} \sum_{j=1}^{\text{len}(u_i)} \text{size}(P_{ij}^{отпр}), \quad (2.2.2)$$

где $\text{len}(U)$ – количество последовательностей u_i отправки клиентом данных на сервер, $\text{len}(u_i)$ – количество сообщений, отправляемых клиентом на сервер в последовательности u_i, $\text{size}(P_{ij}^{отпр})$ – размер в Кбайтах j-го сообщения в последовательности u_i.

Обычно $\text{len}(u_i) = 1$, так как при заполнении одной текстовой формы может формироваться только одно сообщение на сервер, а отправка следующей текстовой формы является уже другой последовательностью u_i, поскольку необходимо заново загрузить с сервера страницу с веб-формой. Даже в случае, если на сервер отправляется, например,

заполненная текстовая форма и несколько прикрепленных к ней файлов, то этот набор отправляется в виде единого комбинированного сообщения типа “multipart/form-data” [44]. В любом случае для каждой последовательности u_i существует одна последовательность загрузки подтверждения $d_{ij}^{подтв}$ доставки единственного $P_{ij}^{отпр}$. Модификация традиционного веб-приложения с асинхронной веб-формой может посылать на сервер несколько сообщений $P_{ij}^{отпр}$ в одной последовательности u_i, при этом количество загружаемых элементов подтверждений доставки каждого отправленного сообщения $P_{ij}^{отпр}$ обычно невелико.

Сетевой трафик $D^{подтв.трад}$, затрачиваемый на загрузку ответа сервера, подтверждающего доставку отправленных клиентом на сервер данных, выражается формулой

$$D^{подтв.трад} = \sum_{i=1}^{\text{len}(U)} \sum_{j=1}^{\text{len}(u_i)} \sum_{j'=1}^{\text{len}(d_{ij}^{подтв})} \text{size}(P_{ijj'}^{подтв.трад}), \quad (2.2.3)$$

где $\text{len}(d_{ij}^{подтв})$ – количество страниц, файлов и сообщений, загруженных клиентом с сервера для подтверждения доставки отправленных на сервер данных в последовательности u_i,

$\text{size}(P_{jj'}^{подтв.трад})$ – размер в Кбайтах j'-й загружаемой с сервера страницы, файла или сообщения, последовательности элементов, подтверждающих доставку сообщения $P_{ij}^{отпр}$, отправленного пользователем на сервер.

2.2.2 Время загрузки страниц, количество запросов и время ожидания ответа сервера традиционного веб-приложения

Время загрузки $t_D^{трад}$ страниц, файлов и сообщений традиционного веб-приложения с сервера состоит из времени загрузки элементов страниц, файлов и сообщений $\frac{D^{трад}}{V_C}$ и времени запросов к серверу, равному произведению времени t_C ожидания ответа сервера на количество запросов к серверу $\left(\text{len}(D) + \sum_{i=1}^{\text{len}(D)} \left\lceil \frac{\text{len}(d_i) - 1}{k^{парал}} \right\rceil \right)$

$$t_D^{трад} = \frac{D^{трад}}{V_C} + t_C \left(\text{len}(D) + \sum_{i=1}^{\text{len}(D)} \left\lceil \frac{\text{len}(d_i) - 1}{k^{парал}} \right\rceil \right), \quad (2.2.4)$$

где $\text{len}(D)$ – количество последовательностей d_i получения клиентом данных с сервера, $\text{len}(d_i)$ – количество страниц, файлов и сообщений, получаемых клиентом с сервера в последовательности d_i, $k^{парал}$ – количество одновременных соединений браузера клиента с сервером, определяющее, сколько файлов может загружаться одновременно, $\lceil \cdot \rceil$ – операция округления до ближайшего большего целого.

В соответствии с порядком работы браузера, показанным на рис. 2.2, в формуле (2.2.4) и далее для расчета количества итераций параллельного обращения клиента к серверу используется округление слагаемых до большего целого.

Время отправки $t_U^{трад}$ введенных данных в традиционном веб-приложении на сервер

$$t_U^{трад} = \frac{U^{трад}}{V_C}. \qquad (2.2.5)$$

Время загрузки $t_D^{подтв.трад}$ подтверждений доставки данных с сервера традиционным веб-приложением состоит из времени загрузки элементов страниц подтверждений $\frac{D^{подтв.трад}}{V_C}$ и времени запросов к серверу, равному произведению времени t_C ожидания ответа сервера на количество запросов к серверу $\left(\sum_{i=1}^{\text{len}(U)} \text{len}(u_i) + \sum_{i=1}^{\text{len}(U)} \sum_{j=1}^{\text{len}(u_i)} \left\lceil \frac{\text{len}(d_{ij}^{подтв}) - 1}{k^{парал}} \right\rceil \right)$

$$t_D^{подтв.трад} = \frac{D^{подтв.трад}}{V_C} + t_C \left(\sum_{i=1}^{\text{len}(U)} \text{len}(u_i) + \sum_{i=1}^{\text{len}(U)} \sum_{j=1}^{\text{len}(u_i)} \left\lceil \frac{\text{len}(d_{ij}^{подтв}) - 1}{k^{парал}} \right\rceil \right), \qquad (2.2.6)$$

где $\text{len}(U)$ – количество последовательностей u_i отправки клиентом данных на сервер, $\text{len}(u_i)$ – количество сообщений, отправляемых клиентом на сервер в последовательности u_i (так как загрузка подтверждений доставки происходит после отправки данных на сервер), $\text{len}(d_{ij}^{подтв})$ – количество страниц, файлов и сообщений, загруженных клиентом с сервера для подтверждения доставки отправленных на сервер данных в последовательности u_i.

Отличие от обычной загрузки страниц состоит в том, что первым запросом является не запрос страницы по ее адресу, а отправка введенных пользователем данных по адресу отправки результатов заполнения веб-формы.

Несмотря на большое сходство загрузки подтверждения отправки данных традиционного веб-приложения с обычной загрузкой страниц их нельзя объединить, так как иногда требуется сначала загрузить набор связанных элементов $\left\{ P_{ijj'}^{подтв.трад} \right\}$, чтобы

удостовериться, что сервер получил введенное сообщение, и только потом загрузить следующую страницу $\left\{P_{ij}^{получ}\right\}$ с данными ИС.

2.2.3 Время использования канала связи традиционным веб-приложением

Время использования канала связи рассчитывается для определения минимальной пропускной способности V_C канала связи, необходимой для функционирования данного приложения.

Если в приложениях с непрерывной передачей данных можно говорить о постоянной загрузке канала связяи в виде среднего сетевого трафика входящего или исходящего направлений в единицу времени, то применительно к веб-приложениям любого типа из-за дискретного характера передачи данных между клиентом и сервером можно говорить об общем времени использования канала клиентом при получении или отправке данных без учета времени ожидания клиентом ответов сервера.

Время использования канала $C_D^{трад}$ входящего направления для канала связи C между клиентом и сервером традиционного веб-приложения при загрузке данных с сервера определяется как

$$C_D^{трад} = \frac{D^{трад}}{V_C}. \tag{2.2.7}$$

Время использования канала $C_U^{трад}$ исходящего направления для канала связи C между клиентом и сервером традиционного веб-приложения при отправке данных на сервер определяется как

$$C_U^{трад} = \frac{U^{трад}}{V_C}. \tag{2.2.8}$$

Время использования канала $C_D^{подтв.трад}$ входящего направления для канала связи C между клиентом и сервером традиционного веб-приложения при получении подтверждения с сервера о загрузке данных определяется как

$$C_D^{подтв.трад} = \frac{D^{подтв.трад}}{V_C}. \tag{2.2.9}$$

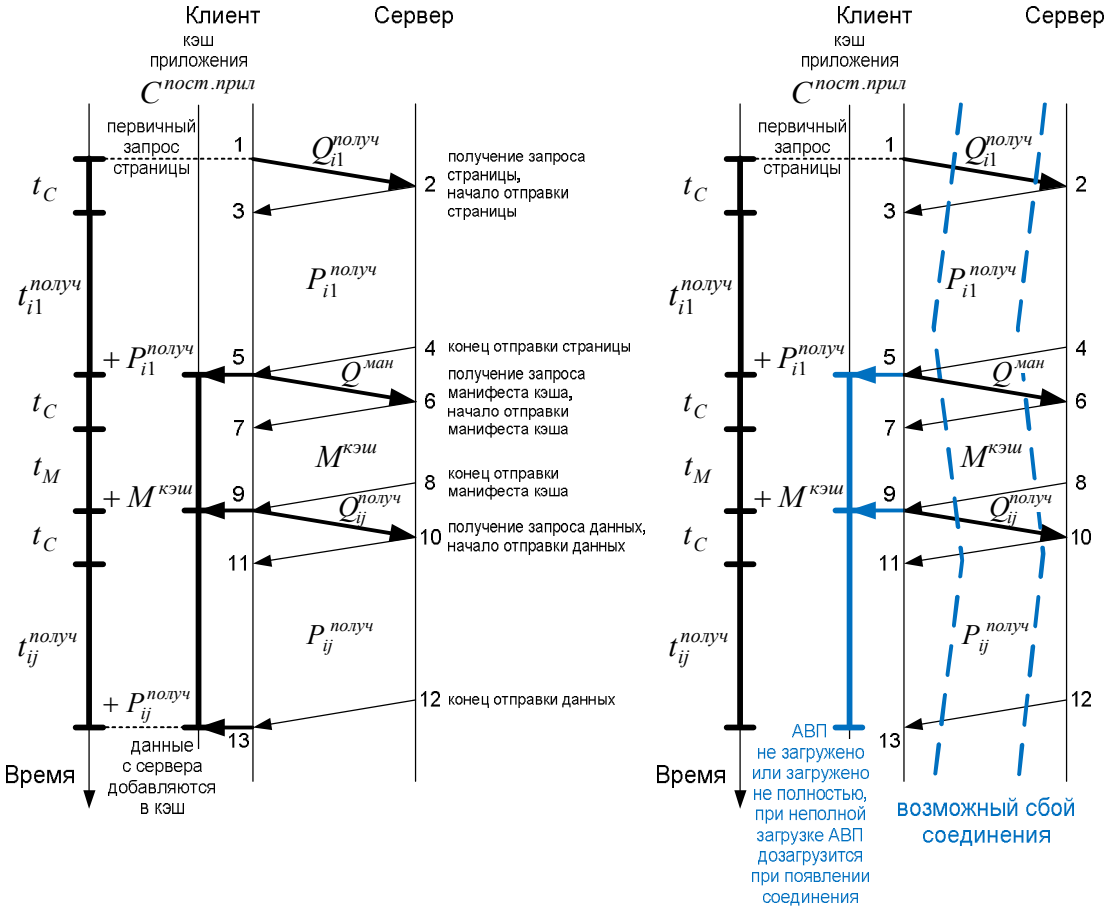
Таким образом, на основе представленной модели традиционного веб-приложения установлены зависимости между количеством его запросов к серверу, объемом трафика, суммарным временем ожидания ответа сервера, временем использования каналов связи и временем загрузки страниц.

2.3 *Автономное отказоустойчивое веб-приложение*

Проанализируем на основе представленной модели АВП количество его запросов к серверу, объем трафика, суммарное время ожидания ответа сервера, время использования каналов связи и время загрузки страниц.

Благодаря наличию $C^{пост.дан}$ и $C^{пост.прил}$ АВП может резервировать все вводимые пользователем данные и обеспечивать несколько режимов работы с данными, получаемыми клиентом с сервера ИС.

Работа клиентской части АВП состоит из первоначальной загрузки данных с сервера для инициализации кэша $C^{пост.прил}$ (рис. 2.10), загрузки новых данных с сервера при работе кэша $C^{пост.прил}$ в режимах 2, 3 и 4 (рис. 2.11), отправки введенных пользователем данных на сервер с использованием кэша $C^{пост.дан}$ для резервирования введенных данных в режиме 1 (рис. 2.12).



а) **б)**

Рис. 2.10. Работа АВП при первоначальной загрузке данных с сервера для инициализации кэша $C^{пост.прил}$: а) в штатном режиме, б) в случае сбоя соединения

Первоначальная загрузка данных с сервера для инициализации кэша $C^{пост.прил}$ в штатном режиме показана на рис. 2.10а и осуществляется в следующем порядке:

1. Клиент посылает запрос $Q_{i1}^{получ}$ к серверу на получение первой страницы АВП.
2. Сервер получает запрос от клиента и начинает отправлять в ответ HTML-страницу $P_{i1}^{получ}$.
3. Через интервал времени t_C первый байт отправленных сервером данных $P_{i1}^{получ}$ достигает клиента.
4. Сервер заканчивает отправку данных $P_{i1}^{получ}$ клиенту.
5. Через интервал времени $t_{i1}^{получ}$ данные $P_{i1}^{получ}$ полностью получены клиентом. Клиент находит ссылку на манифест кэша $M^{кэш}$ в теле страницы $P_{i1}^{получ}$ и посылает серверу запрос $Q^{ман}$ на загрузку манифеста кэша, а также сохраняет данные $P_{i1}^{получ}$ в $C^{пост.прил}$.
6. Сервер получает запрос $Q^{ман}$ и начинает отправлять клиенту файл $M^{кэш}$.
7. Через интервал времени t_C первый байт отправленного сервером манифеста кэша $M^{кэш}$ достигает клиента.
8. Сервер заканчивает отправку манифеста кэша $M^{кэш}$ клиенту.
9. Через интервал времени t_M манифест кэша $M^{кэш}$ полностью получен клиентом. Клиент сохраняет манифест кэша в $C^{пост.прил}$ и посылает запрос $Q_{ij}^{получ}$ к серверу на получение первого файла из списка перечисленных в манифесте кэша.
10. Сервер получает запрос от клиента и начинает отправлять в ответ данные $P_{ij}^{получ}$.
11. Через интервал времени t_C первый байт отправленных сервером данных $P_{ij}^{получ}$ достигает клиента.
12. Сервер заканчивает отправку данных $P_{ij}^{получ}$ клиенту.
13. Через интервал времени $t_{ij}^{получ}$ данные $P_{ij}^{получ}$ полностью получены клиентом и клиент сохраняет их в $C^{пост.прил}$.

Запросы $Q^{ман}$ манифеста кэша и $Q_{ij}^{получ}$ файлов с сервера в точках 5 и 9 выполняются параллельно группами по $k^{парал}$ штук. На схемах на рис. 2.10 и 2.11 изображено взаимодействие клиента с сервером при $k^{парал}=1$.

Процедура отправки запросов $Q_{ij}^{получ}$, получения и сохранения в $C^{пост.прил}$ данных $P_{ij}^{получ}$ повторяется, пока все данные, перечисленные в $M^{кэш}$, не будут загружены.

Если сбой соединения с сервером (рис. 2.10б) произошел до того, как клиент успел получить файл манифеста кэша $M^{кэш}$ (точка 9), то дальнейшая работа с АВП требует повторной инициализации.

Если до возникновения сбоя клиент успел получить файл $M^{кэш}$, то АВП будет автоматически пытаться загрузить все перечисленные в нем файлы, пока не загрузит их.

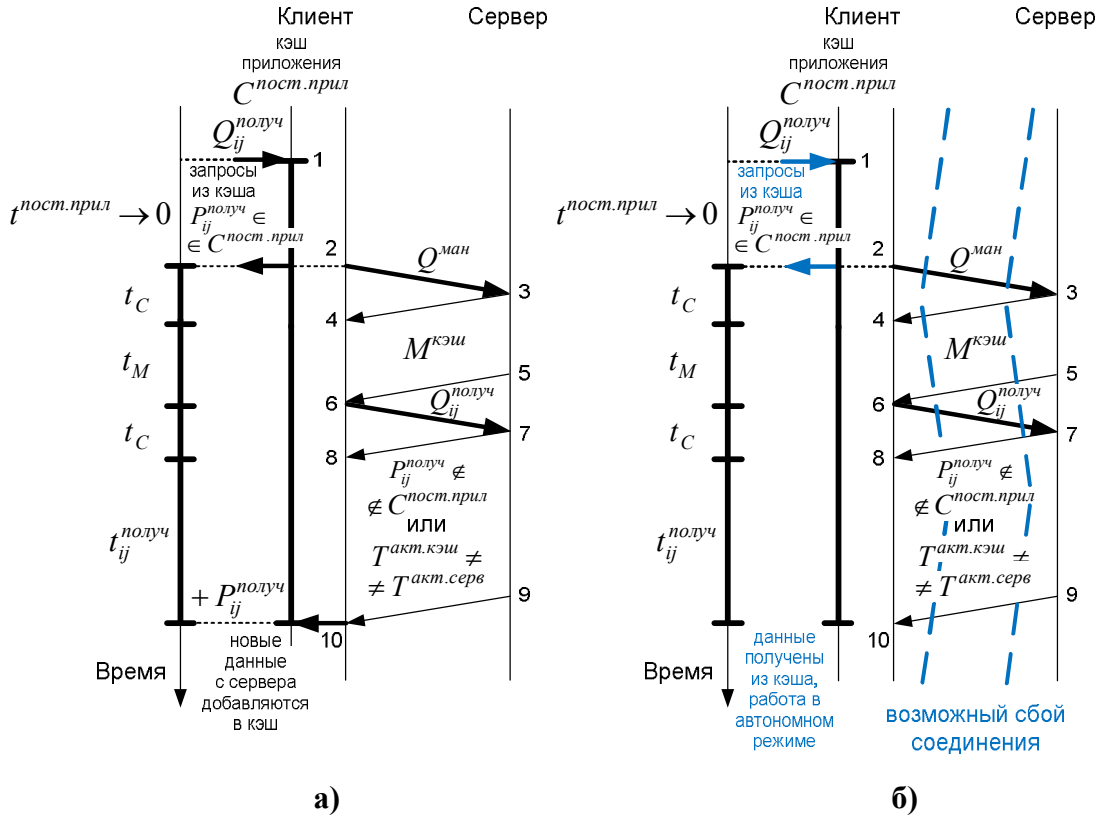


Рис. 2.11. Работа АВП при загрузке новых данных с сервера при работе кэша $C^{пост.прил}$ в режимах 2, 3, 4: а) в штатном режиме, б) в случае сбоя соединения

Загрузка новых данных с сервера при работе кэша $C^{пост.прил}$ в режимах 2, 3 и 4 при наличии соединения с сервером показана на рис. 2.11а и осуществляется в следующем порядке:

1. В режимах 2 и 3 часть запросов, а в режиме 4 все запросы $Q_{ij}^{получ}$ клиента на получение страниц АВП и их элементов под управлением манифеста кэша получают ответы из кэша $C^{пост.прил}$.
2. Кэш $C^{пост.прил}$ позволяет браузеру практически мгновенно $t^{пост.прил} \to 0$ начать отображение данных $P_{ij}^{получ}$, сохраненных локально, без подключения к серверу. Для проверки актуальности данных в $C^{пост.прил}$ клиент отправляет на сервер запрос $Q^{ман}$ на загрузку файла манифеста кэша $M^{кэш}$.
3. Сервер получает запрос $Q^{ман}$ и начинает отправлять клиенту файл $M^{кэш}$.
4. Через интервал времени t_C первый байт отправленного сервером манифеста кэша $M^{кэш}$ достигает клиента.
5. Сервер заканчивает отправку манифеста кэша $M^{кэш}$ клиенту.
6. Через интервал времени t_M манифест кэша $M^{кэш}$ полностью получен клиентом. Клиент сохраняет манифест кэша в $C^{пост.прил}$ и посылает запрос $Q_{ij}^{получ}$ к серверу на получение очередного файла $P_{ij}^{получ}$, которого нет в $C^{пост.прил}$ или на обновление очередного файла из $C^{пост.прил}$, если отметка времени в сохраненном локально и полученном с сервера файлах манифеста кэша не совпадают, $T^{акт.кэш} \neq T^{акт.серв}$.
7. Сервер получает запрос от клиента и начинает отправлять в ответ данные $P_{ij}^{получ}$.
8. Через интервал времени t_C первый байт отправленных сервером данных $P_{ij}^{получ}$ достигает клиента.
9. Сервер заканчивает отправку данных $P_{ij}^{получ}$ клиенту.
10. Через интервал времени $t_{ij}^{получ}$ данные $P_{ij}^{получ}$ полностью получены клиентом и клиент сохраняет их в $C^{пост.прил}$.

Процедура отправки запросов $Q_{ij}^{получ}$, получения и сохранения в $C^{пост.прил}$ данных $P_{ij}^{получ}$ повторяется, пока все данные, перечисленные в $M^{кэш}$ не будут загружены или обновлены.

Если происходит сбой соединения (рис. 2.11б), то АВП продолжает работу с локально сохраненными в $C^{пост.прил}$ данными.

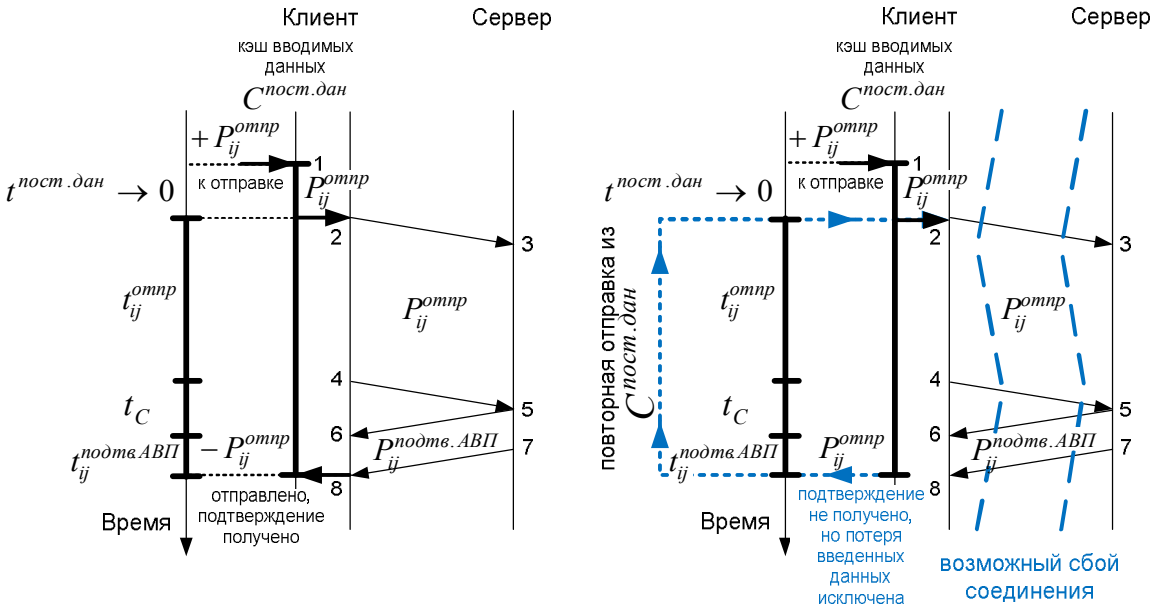


Рис. 2.12. Работа АВП при введенных пользователем данных на сервер при работе кэша $C^{пост.дан}$ в режиме 1: а) в штатном режиме, б) в случае сбоя соединения

Отправка введенных пользователем данных на сервер с использованием кэша $C^{пост.дан}$ для резервирования введенных данных в режиме 1 при наличии соединения с сервером показана на рис. 2.12а и осуществляется в следующем порядке:

1. Клиент сохраняет введенные пользователем данные $P_{ij}^{отпр}$ в локальный кэш $C^{пост.дан}$. Это происходит практически мгновенно, так как $t^{пост.дан} \to 0$.
2. Клиент начинает отправку введенных пользователем данных $P_{ij}^{отпр}$ на сервер.
3. Сервер получает первый байт данных $P_{ij}^{отпр}$.
4. Через интервал времени $t_{ij}^{отпр}$ клиент полностью отправил данные $P_{ij}^{отпр}$ на сервер.
5. Сервер заканчивает получение данных $P_{ij}^{отпр}$ от клиента и начинает отправку клиенту подтверждения $P_{ij}^{подтв.АВП}$ получения введенных данных.
6. Через интервал времени t_C клиент получает первый байт подтверждения $P_{ij}^{подтв.АВП}$.

7. Сервер заканчивает отправку данных $P_{ij}^{подтв.АВП}$ клиенту.
8. Через интервал времени $t_{ij}^{подтв.АВП}$ данные подтверждения $P_{ij}^{подтв.АВП}$ полностью получены клиентом. Клиент удаляет успешно доставленные на сервер данные $P_{ij}^{отпр}$ из кэша $C^{пост.дан}$.

В случае сбоя соединения с сервером клиент не получает подтверждения от сервера в точках 6 или 8 (рис. 2.12б). В этом случае клиент будет автоматически пытаться отправить данные $P_{ij}^{отпр}$ из кэша $C^{пост.дан}$ через определенные промежутки времени и при восстановлении соединения отправит их. Сохраненные данные будут удалены из кэша $C^{пост.дан}$ только после их успешной доставки на сервер и получения клиентом подтверждения.

При работе с АВП можно свободно закрывать браузер и даже выключать устройство – данные в $C^{пост.дан}$ и $C^{пост.прил}$ не пропадут.

Режимы 2, 3 и 4 работы с данными, получаемыми клиентом с сервера, описываются общими формулами, и отличие между ними заключается в разных значениях коэффициентов, учитывающих попадание запросов в кэш данных веб-приложения как по числу запросов, так и по количеству переданных байт. Этими коэффициентами являются:

1. H (англ. hit rate или hit ratio) – доля запросов, обслуживаемых из кэша (по количеству запросов) – среднее значение по всем последовательностям получения данных $d_i \in D$ отношений H_i количества запросов клиента K в последовательности d_i, ответ на которые был получен из кэша данных веб-приложения $C^{пост.прил}$, к общему количеству запросов клиента K в этой последовательности.

$$H = \frac{1}{\text{len}(D)} \sum_{i=1}^{\text{len}(D)} H_i , \tag{2.3.1}$$

где

$$H_i = \frac{m_i}{k_i}, \tag{2.3.2}$$

где m_i – количество запросов (страниц, файлов и сообщений) в рамках последовательности d_i, ответ на которые был получен из кэша $C^{пост.прил}$,

k_i – общее количество запросов (страниц, файлов и сообщений) в рамках последовательности d_i.

Подставляя (2.3.2) в (2.3.1) получим

$$H = \frac{1}{\text{len}(D)} \sum_{i=1}^{\text{len}(D)} \frac{m_i}{k_i}. \quad (2.3.3)$$

Если $\forall i \;\; m_i = k_i$, то $H = 1$, что означает нахождение ответов на все запросы в кэше $C^{пост.прил}$.

2. B (англ. byte hit rate или byte hit ratio) – доля данных, подаваемых из кэша (по объему данных) – среднее значение по всем последовательностям получения данных $d_i \in D$ отношений B_i суммарного количества байт данных элементов последовательности d_i, которые были получены клиентом K из кэша данных веб-приложения $C^{пост.прил}$ к общему числу байт данных, полученных клиентом K в этой последовательности.

$$B = \frac{1}{\text{len}(D)} \sum_{i=1}^{\text{len}(D)} B_i, \quad (2.3.4)$$

где

$$B_i = \frac{\sum_{m=1}^{\text{len}(d_i^{сохр})} \text{size}(P_{im}^{сохр})}{\sum_{j=1}^{\text{len}(d_i)} \text{size}(P_{ij}^{получ})}, \quad (2.3.5)$$

$\text{len}(d_i^{сохр})$ – количество файлов или сообщений последовательности получения данных d_i, подаваемых клиенту из кэша $C^{пост.прил}$,

$\text{size}(P_{im}^{сохр})$ – размер в Кбайтах файла или сообщения последовательности получения данных d_i, подаваемого клиенту из кэша $C^{пост.прил}$, при условии, что $P_{im}^{сохр} = P_{ij}^{получ}$.

Подставляя (2.3.5) в (2.3.4), получим

$$B = \frac{1}{\text{len}(D)} \sum_{i=1}^{\text{len}(D)} \frac{\sum_{m=1}^{\text{len}(d_i^{сохр})} \text{size}(P_{im}^{сохр})}{\sum_{j=1}^{\text{len}(d_i)} \text{size}(P_{ij}^{получ})}. \quad (2.3.6)$$

Если $\forall i \;\; d_i = d_i^{сохр}$, то $B = 1$, что означает нахождение всех запрашиваемых страниц, файлов и сообщений в кэше $C^{пост.прил}$.

Средние значения H и B используются потому, что в ходе эксплуатации АВП кэш $C^{пост.прил}$ пополняется новыми $P_{ij}^{получ}$ и значения коэффициентов H и B увеличиваются.

Коэффициенты H и B могут принимать различные значения в зависимости от типа информационного наполнения АВП.

Для систем обеспечения доступа к данным ДЗЗ и информационной поддержки различных видов деятельности основная польза состоит в автономном управляемом манифестом кэша $M^{кэш}$ кэшировании в $C^{пост.прил}$ и поддержании в актуальном состоянии страниц из ИС, что позволяет достичь полной автономности (режим 4) с $H=1$, $B=1$ и работать без необходимости постоянного подключения к серверу.

Для среднестатистической страницы в Интернете при использовании АВП $H=0{,}46$ [52]

Таким образом, все без исключения виды АВП получают выгоду от режимов повышения отказоустойчивости и увеличения скорости загрузки с резервированием вводимых пользователем данных, минимальным набором данных начального кэша для простых приложений и с универсальным набором данных для больших приложений с разнообразным функционалом.

2.3.1 Сетевой трафик АВП

Сетевой трафик АВП $D^{АВП}$ при загрузке данных с сервера определяется по формуле

$$D^{АВП}=\sum_{i=1}^{\text{len}(D)}\left(\text{size}(M^{кэш})+(1-B)\sum_{j=1}^{\text{len}(d_i)}\text{size}(P_{ij}^{получ})\right). \quad (2.3.7)$$

В отличие от традиционного веб-приложения АВП не требуется повторная загрузка данных после их первоначального кэширования в $C^{пост.прил}$, а доля объема закэшированных данных составляет B.

В случае полностью автономного режима работы АВП в кэше $C^{пост.прил}$ в каждой отдельной последовательности d_i загрузки j страниц, файлов и сообщений с сервера присутствуют все странницы, файлы и сообщения, перечисленные в ранее загруженном $M^{кэш}$.

Если $B=1$, запрашиваемый с сервера $\text{size}(M^{кэш})=0$ (недоступен), то $D^{АВП}=0$, что соответствует полностью автономной работе веб-приложения без загрузки данных с сервера.

$M^{кэш}$ загружается только при наличии соединения с сервером для проверки актуальности имеющейся локальной копии данных. Невозможность его загрузки при отсутствии соединения с сервером говорит приложению, что оно работает в автономном режиме. Недоступность $M^{кэш}$ с сервера не влияет на работоспособность АВП в автономном режиме, так как приложение имеет локальную копию $M^{кэш}$.

Сетевой трафик АВП $U^{АВП}$ при загрузке данных на сервер определяется как

$$U^{АВП} = \sum_{i=1}^{\text{len}(U)} \sum_{j=1}^{\text{len}(u_i)} \text{size}(P_{ij}^{отпр}). \quad (2.3.8)$$

Сетевой трафик $D^{подтв.АВП}$, затрачиваемый АВП на загрузку ответа сервера, подтверждающего наличие соединения с сервером и доставку отправленных клиентом на сервер данных, определяется как

$$D^{подтв.АВП} = \sum_{i=1}^{\text{len}(U)} \sum_{j=1}^{\text{len}(u_i)} \text{size}(P_{ij}^{подтв.АВП}), \quad (2.3.9)$$

где $\text{size}(P_{ij}^{подтв.АВП})$ – размер в Кбайтах сообщения об успешном получении введенных данных, отправляемого сервером клиенту в ответ на получение каждого сообщения $P_{ij}^{отпр}$. В сообщениях $P_{ij}^{подтв.АВП}$ может содержаться различная информация в зависимости от отправленных данных.

Сообщения $P_{ij}^{подтв.АВП}$ не кэшируются, так как отражают текущее состояние сервера и результата передачи данных.

2.3.2 Время загрузки страниц, количество запросов и время ожидания ответа сервера АВП

Время загрузки $t_D^{АВП}$ страниц, файлов и сообщений АВП с сервера состоит из времени загрузки манифеста кэша, элементов страниц, файлов и сообщений $\frac{D^{АВП}}{V_C}$ и времени запросов к серверу, равному произведению времени t_C ожидания ответа сервера на количество запросов к серверу $\left((1-H)\,\text{len}(D) + \sum_{i=1}^{\text{len}(D)} \left\lceil \frac{1+(1-H)(\text{len}(d_i)-1)}{k^{парал}} \right\rceil\right)$. При загрузке АВП первым перед остальными элементами страницы без учета $k^{парал}$ загружается главный HTML-файл каждой страницы. Как и остальные элементы страницы, он может загружаться из $C^{пост.прил}$, если был закэширован. Для проверки обновлений закэшированных файлов АВП всегда старается загрузить манифест кэша $M^{кэш}$ с сервера, может это делать параллельно с загрузкой других некэшированных файлов с учетом $k^{парал}$ и использует локально сохраненную версию $M^{кэш}$, только если сервер недоступен.

$$t_D^{АВП} = \frac{D^{АВП}}{V_C} + t_C\left((1-H)\,\mathrm{len}(D) + \sum_{i=1}^{\mathrm{len}(D)}\left\lceil \frac{1+(1-H)(\mathrm{len}(d_i)-1)}{k^{парал}} \right\rceil\right). \quad (2.3.10)$$

Если $H = 1$, автоматически $B = 1$, так как кэшируемые элементы сохраняются в кэше $C^{пост.прил}$ только полностью, и тогда $t_D^{АВП} = (t_M + t_C)\,\mathrm{len}(D)$, что соответствует фоновой проверке манифеста кэша и автономной работе АВП с моментальным открытием страниц из $C^{пост.прил}$.

Время отправки $t_U^{АВП}$ данных АВП на сервер

$$t_U^{АВП} = \frac{U^{АВП}}{V_C}. \quad (2.3.11)$$

Время загрузки $t_D^{подтв.АВП}$ подтверждений доставки данных с сервера АВП состоит из времени загрузки подтверждений получения сервером отправленных сообщений $\frac{D^{подтв.АВП}}{V_C}$ и времени ожидания подтверждения сервера после отправки введенных данных, равному произведению времени t_C ожидания ответа сервера на количество отправленных на сервер сообщений $\sum_{i=1}^{\mathrm{len}(U)}\mathrm{len}(u_i)$, так как в АВП на одно сообщение требуется только один файл подтверждения

$$t_D^{подтв.АВП} = \frac{D^{подтв.АВП}}{V_C} + t_C \sum_{i=1}^{\mathrm{len}(U)}\mathrm{len}(u_i). \quad (2.3.12)$$

2.3.3 Время использования канала связи АВП

Время использования канала $C_D^{АВП}$ входящего направления для канала связи C между АВП и сервером при загрузке данных с сервера определяется как

$$C_D^{АВП} = \frac{D^{АВП}}{V_C}. \quad (2.3.13)$$

В случае, если все запрашиваемые клиентом файлы последовательности получения данных d_i находятся в $C^{пост.прил}$, то есть $\forall P_{ij}^{получ}\ \exists P_{im}^{сохр}$ такие, что $P_{ij}^{получ} = P_{im}^{сохр}$, $B = 1$, а запрашиваемый с сервера манифест кэша недоступен – $\mathrm{size}(M^{кэш}) = 0$, то $C_D^{АВП} = 0$, что соответствует полностью автономной работе веб-приложения без использования каналов связи.

Время использования канала $C_U^{АВП}$ исходящего направления для канала связи C между АВП и сервером при отправке данных на сервер определяется как

$$C_U^{АВП} = \frac{U^{АВП}}{V_C} \ . \tag{2.3.14}$$

Время использования канала $C_D^{подтв.АВП}$ входящего направления для канала связи C между АВП и сервером при получении подтверждения с сервера о загрузке данных определяется как

$$C_D^{подтв.АВП} = \frac{D^{подтв.АВП}}{V_C} \ . \tag{2.3.15}$$

Таким образом, на основе представленной модели АВП установлены зависимости между количеством его запросов к серверу, объемом трафика, суммарным временем ожидания ответа сервера, временем использования каналов связи и временем загрузки страниц.

2.4 Сравнение традиционного веб-приложения и АВП

На основе установленных по моделям зависимостей проанализируем преимущества АВП по сравнению с традиционным веб-приложением по критериям уменьшения количества запросов к серверу, экономии трафика, уменьшения суммарного времени ожидания ответа сервера, времени использования каналов связи и времени загрузки страниц.

2.4.1 Экономия трафика в АВП

Экономия трафика ΔD при загрузке данных с севера при работе с АВП по сравнению с традиционным веб-приложением достигается за счет загрузки определенной доли данных от общего объема загружаемых данных из локального хранилища $C^{пост.прил}$ и определяется по формуле

$$\Delta D = D^{трад} - D^{АВП} =$$

$$= \sum_{i=1}^{\text{len}(D)} \sum_{j=1}^{\text{len}(d_i)} \text{size}(P_{ij}^{получ}) - \sum_{i=1}^{\text{len}(D)} \left(\text{size}(M^{кэш}) + (1-B) \sum_{j=1}^{\text{len}(d_i)} \text{size}(P_{ij}^{получ}) \right) =$$

$$= \sum_{i=1}^{\text{len}(D)} \left(B \sum_{j=1}^{\text{len}(d_i)} \text{size}(P_{ij}^{получ}) - \text{size}(M^{кэш}) \right), \tag{2.4.1}$$

при этом $\Delta D > 0$, так как при правильной организации работы кэша АВП размер закэшированных файлов всегда больше размера файла манифеста кэша $M^{кэш}$. Из

формулы (2.4.1) видно, что чем больше B – доля объема данных, подаваемых из кэша $C^{пост.прил}$, тем больше экономия трафика при загрузке данных с сервера.

Экономия трафика $\Delta D^{подтв}$, затрачиваемого на загрузку ответа сервера, подтверждающего доставку отправленных клиентом на сервер данных при работе с АВП по сравнению с традиционным веб-приложением достигается за счет уменьшения количества загружаемых файлов подтверждения доставки и снижения их объема, и определяется по формуле

$$\Delta D^{подтв} = D^{подтв.трад} - D^{подтв.АВП} =$$

$$= \sum_{i=1}^{\text{len}(U)} \sum_{j=1}^{\text{len}(u_i)} \sum_{j'=1}^{\text{len}(d_{ij}^{подтв})} \text{size}(P_{ijj'}^{подтв.трад}) - \sum_{i=1}^{\text{len}(U)} \sum_{j=1}^{\text{len}(u_i)} \text{size}(P_{ij}^{подтв.АВП}) =$$

$$= \sum_{i=1}^{\text{len}(U)} \sum_{j=1}^{\text{len}(u_i)} \left(\left(\sum_{j'=1}^{\text{len}(d_{ij}^{подтв})} \text{size}(P_{ijj'}^{подтв.трад}) \right) - \text{size}(P_{ij}^{подтв.АВП}) \right), \quad (2.4.2)$$

при этом $\Delta D^{подтв} > 0$, так как

$$\sum_{j'=1}^{\text{len}(d_{ij}^{подтв})} \text{size}(P_{ijj'}^{подтв.трад}) > \text{size}(P_{ij}^{подтв.АВП}) \quad (2.4.3)$$

и даже $\text{size}(P_{ij1}^{подтв.трад}) > \text{size}(P_{ij}^{подтв.АВП})$, поскольку $P_{ij1}^{подтв.трад}$ является HTML-файлом или HTML-сообщением (в случае асинхронного подтверждения), содержащим информацию и HTML-разметку для формирования интерфейса, в то время как $P_{ij}^{подтв.АВП}$ содержит только информацию, а все неизменяемые компоненты интерфейса уже находятся в $C^{пост.прил}$ на момент отправки пользователем каких-либо введенных данных на сервер.

Сами отправляемые на сервер введенные пользователем данные одинаковы в обоих случаях, как при работе с традиционным веб-приложением, так и при работе с АВП.

2.4.2 Уменьшение времени загрузки страниц, снижение количества запросов к серверу и уменьшение времени ожидания ответа сервера АВП

Уменьшение времени загрузки Δt_D страниц клиентом с сервера ИС при работе с АВП по сравнению с традиционным веб-приложением достигается за счет уменьшения загружаемого объема данных и снижения количества запросов к серверу (что, в свою очередь, влияет на уменьшение времени ожидания ответа сервера), и определяется по формуле

$$\Delta t_D = t_D^{трад} - t_D^{АВП} =$$

$$= \frac{D^{трад}}{V_C} + t_C\left(\text{len}(D) + \sum_{i=1}^{\text{len}(D)}\left\lceil \frac{\text{len}(d_i)-1}{k^{парал}} \right\rceil\right) -$$

$$- \frac{D^{АВП}}{V_C} - t_C\left((1-H)\,\text{len}(D) + \sum_{i=1}^{\text{len}(D)}\left\lceil \frac{1+(1-H)(\text{len}(d_i)-1)}{k^{парал}} \right\rceil\right) =$$

$$= \frac{\Delta D}{V_C} + t_C\left(H\,\text{len}(D) + \sum_{i=1}^{\text{len}(D)}\left\lceil \frac{H(\text{len}(d_i)-1)-1}{k^{парал}} \right\rceil\right), \qquad (2.4.4)$$

при этом $\Delta t_D > 0$, так как в среднем для веб-сайтов в Интернете при использовании АВП $H = 0{,}46$ и $\text{len}(d_i) = 96$ [52].

Уменьшение времени загрузки $\Delta t_D^{подтв}$ подтверждения с сервера о получении введенных данных при работе с АВП по сравнению с традиционным веб-приложением достигается за счет уменьшения загружаемого объема подтверждений и снижения количества запросов к серверу, и определяется по формуле

$$\Delta t_D^{подтв} = t_D^{подтв.трад} - t_D^{подтв.АВП} =$$

$$= \frac{D^{подтв.трад}}{V_C} + t_C\left(\sum_{i=1}^{\text{len}(U)}\text{len}(u_i) + \sum_{i=1}^{\text{len}(U)}\sum_{j=1}^{\text{len}(u_i)}\left\lceil \frac{\text{len}(d_{ij}^{подтв})-1}{k^{парал}} \right\rceil\right) -$$

$$- \frac{D^{подтв.АВП}}{V_C} - t_C \sum_{i=1}^{\text{len}(U)}\text{len}(u_i) =$$

$$= \frac{\Delta D^{подтв}}{V_C} + t_C \sum_{i=1}^{\text{len}(U)}\sum_{j=1}^{\text{len}(u_i)}\left\lceil \frac{\text{len}(d_{ij}^{подтв})-1}{k^{парал}} \right\rceil, \qquad (2.4.5)$$

при этом $\Delta t_D^{подтв} > 0$, так как $\text{len}(d_{ij}^{подтв}) \geq 1$, поскольку каждому отправленному сообщению соответствует последовательность загрузки данных подтверждения $d_{ij}^{подтв}$, состоящая не менее чем из одного элемента.

2.4.3 Уменьшение времени использования канала связи в АВП

Уменьшение времени использования ΔC_D канала связи между клиентом и сервером при загрузке данных с сервера АВП по сравнению с традиционным веб-приложением достигается за счет уменьшения объема загружаемых данных и составляет

$$\Delta C_D = \frac{D^{трад}}{V_C} - \frac{D^{АВП}}{V_C} = \frac{\Delta D}{V_C}, \qquad (2.4.6)$$

при этом $\Delta C_D > 0$, так как $\Delta D > 0$.

Уменьшение времени использования $\Delta C_D^{подтв}$ канала связи между клиентом и сервером при получении подтверждения с сервера о загрузке данных АВП по сравнению с традиционным веб-приложением достигается за счет уменьшения объема загружаемых подтверждений и составляет

$$\Delta C_D^{подтв} = \frac{D^{подтв.трад}}{V_C} - \frac{D^{подтв.АВП}}{V_C} = \frac{\Delta D^{подтв}}{V_C}, \quad (2.4.7)$$

при этом $\Delta C_D^{подтв} > 0$, так как $\Delta D^{подтв} > 0$.

Таким образом, в результате анализа моделей традиционного веб-приложения и АВП установлено, что АВП, помимо наличия функции резервирования вводимых пользователем данных и возможности полностью автономной работы с данными, полученными с сервера ИС, также превосходят традиционные веб-приложения по критериям уменьшения количества запросов к серверу, экономии трафика, уменьшения суммарного времени ожидания ответа сервера, времени использования каналов связи и времени загрузки страниц.

2.5 Исследование моделей работы традиционного веб-приложения и АВП

Покажем, за счет чего предлагаемое решение позволяет достичь таких результатов. Рассмотрим схему переходов между состояниями традиционного веб-приложения для выявления переходов, которые могут стать невозможными под влиянием разрывов соединения или сильного снижения скорости передачи данных, что означает нарушение работы приложения.

На рис. 2.13 показаны переходы между состояниями традиционного веб-приложения и участки возможных разрывов соединения. Разрывы соединения с сервером полностью блокируют загрузку с сервера страниц и связанных с ними файлов, а также отправку пользователем на сервер введенных данных из загруженных в момент наличия соединения с сервером веб-форм.

В рассматриваемой схеме не предусмотрено никаких резервных состояний и служебных переходов, позволяющих оградить данные, с которыми работает пользователь, от влияния помех и разрывов соединения с сервером.

Именно поэтому использование традиционных веб-приложений в клиент-серверных ИС для мобильных клиентов не получило до сих пор большого распространения, хотя количество разнородных мобильных платформ растет, а поддерживать отдельные нативные клиентские приложения одновременно для всех уже существующих и появляющихся новых мобильных платформ невозможно по техническим и экономическим причинам.

До настоящего времени для разнородных мобильных устройств не существовало единого приемлемого способа для создания универсальных клиентских приложений для клиент-серверных ИС.

В данной работе предлагается такой способ на основе АВП.

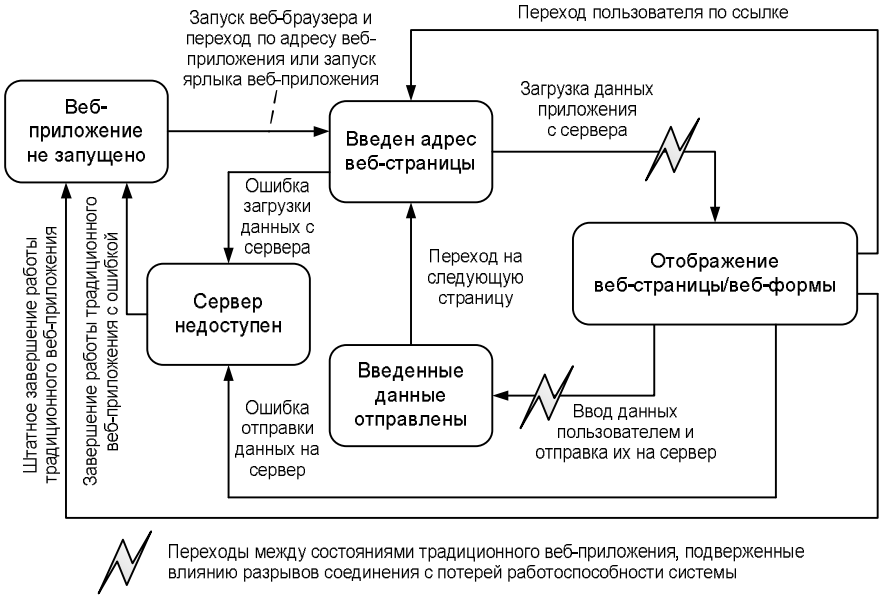


Рис. 2.13. Схема основных переходов между состояниями традиционного веб-приложения и участки возможных разрывов соединения

После модернизации существующего веб-приложения до АВП посредством добавления в существующее веб-приложение разработанных универсальных кроссплатформенных библиотек для реализации функций резервирования вводимых пользователем данных и обеспечения возможности автономной работы схема переходов между состояниями полученного приложения преобразуется к виду, показанному на рис. 2.14.

При данной конфигурации веб-приложения в нем возникают промежуточные состояния, дополнительные переходы и фоновые процессы, которые обуславливают:

- полное исключение потерь вводимых пользователем данных;
- минимизацию негативного влияния возможных потерь получаемых с сервера данных;
- возможность полностью автономной работы с особо важными данными;
- снижение расхода сетевого трафика;
- снижение загрузки каналов связи;
- сокращение времени загрузки страниц.

Таким образом, значительно снижается влияние помех и разрывов соединения клиента с сервером на работу пользователя с данными, отправляемыми им и получаемыми с сервера ИС.

На рис. 2.14 показаны дополнительные состояния и переходы, обеспечивающие АВП возможность не терять работоспособность при разрывах соединения с сервером ИС или при ухудшении качества связи, а лишь приостанавливать работу с незагруженными данными.

Сочетание процессов переднего плана, фоновых процессов и настроек кэширования, управляемых с сервера, в АВП позволяют пользователю работать с ним как с нативным приложением, часто даже не замечая наличия или отсутствия подключения к серверу ИС.

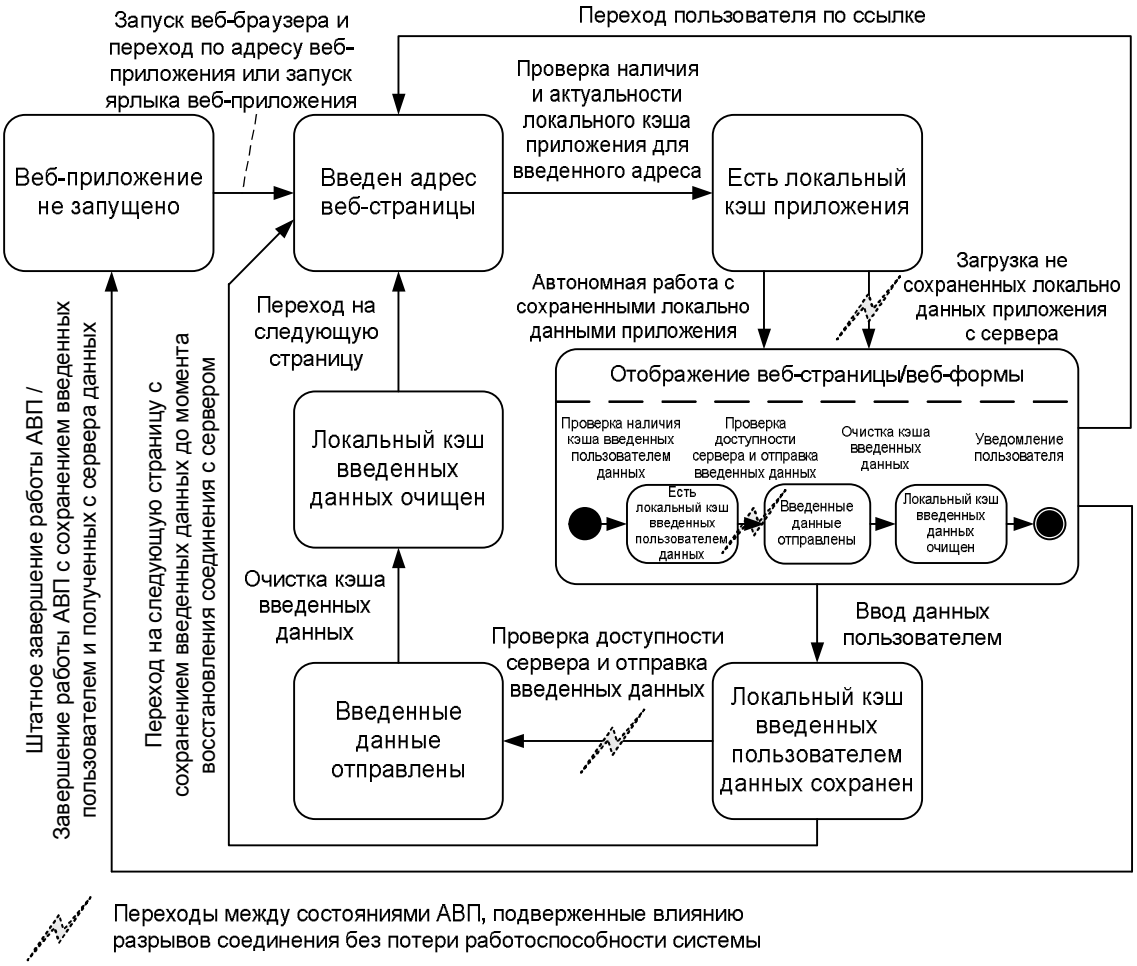


Рис. 2.14. Схема основных переходов между состояниями АВП

Конфигурация АВП, показанная на рис. 2.14, позволяет заменить ими даже нативные клиентские приложения для клиент-серверных систем.

Это обеспечит унифицированную поддержку всех существующих и будущих стационарных и мобильных платформ благодаря использованию стандарта HTML5, принятого и поддерживаемого всеми современными браузерами. Использование постоянного локального хранилища для получаемых с сервера данных позволяет не только работать в автономном режиме при сбоях соединения с сервером, экономить трафик, но и повышать скорость взаимодействия веб-клиента на базе АВП с сервером ИС. Это подтверждается результатами проведенных расчетов и испытаний.

При проведении расчетов учитывалось, что АВП будет работать в мобильных устройствах с различными видами беспроводных каналов связи C от быстрых частных Wi-Fi с большой пропускной способностью V_C и небольшим временем отклика сервера t_C, до беспроводных спутниковых каналов часто с меньшей пропускной способностью V_C и существенно большим временем отклика сервера t_C. Во всех графиках принято $t_C = 30$ мс, что соответствует беспроводной связи по Wi-Fi 802.11g.

2.5.1 Уменьшение трафика, количества запросов к серверу и времени загрузки страниц при загрузке подтверждений доставки введенных данных АВП

Минимальные достаточные размеры сообщений об успешном получении введенных данных, отправляемых сервером клиенту, составляют $P_{ij1}^{подтв.трад} = 5$ Кбайт для традиционного веб-приложения и $P_{ij}^{подтв.АВП} = 1$ Кбайт – для АВП. Это связано с тем, что традиционное веб-приложение должно загрузить для подтверждения полную HTML-страницу, содержащую текст подтверждения и HTML-разметку (2.2.3), а АВП загружает только сам текст подтверждения (2.3.9) с помощью AJAX и для отображения подтверждений использует программный код из $C^{пост.прил}$, генерирующий типовые сообщения на стороне клиента. Таким образом, минимальная экономия трафика при использовании АВП по сравнению с традиционным веб-приложением при отправке введенных пользователем данных на сервер составляет 4 Кбайта на каждое отправленное на сервер сообщение (2.4.2). В реальных условиях минимальное подтверждение получения сервером веденных данных $P_{ij1}^{подтв.трад}$ может быть значительно больше 5 Кбайт для традиционного веб-приложения, что зависит от объема HTML-разметки и наличия дополнительных блоков данных или элементов на странице подтверждения, которые загружаются в виде отдельных файлов.

Для АВП размер единственного $P_{ij}^{подтв.АВП}$ для одного $P_{ij}^{отпр}$ превышает 1 Кбайт лишь в редких случаях, когда сервер сообщает отладочную информацию. Такие случаи для обычных пользователей исключены.

Учитывая объем трафика подтверждения доставки данных на сервер для рассматриваемых приложений, оценим экономию суммарного времени $t_{ij}^{отпр} + t_{ij}^{подтв}$ в секундах отправки введенных пользователем данных $t_{ij}^{отпр}$ и получения минимального

достаточного подтверждения доставки от сервера $t_{ij}^{подтв}$ для различных объемов отправляемых сообщений $P_{ij}^{отпр}$. Хотя объем передаваемых данных $P_{ij}^{отпр}$, а значит и $t_{ij}^{отпр}$, одинаков в традиционном веб-приложении (2.2.2) и в АВП (2.3.8), рассматривается суммарное время $t_{ij}^{отпр} + t_{ij}^{подтв}$ для того, чтобы продемонстрировать значимость снижения времени загрузки подтверждения сервера (2.4.5) при малых размерах отправляемых данных $P_{ij}^{отпр}$ (в большинстве случаев ручного ввода данных) и низкой пропускной способности канала V_C (при использовании каналов мобильной связи).

Таблица 2.1 и график на рис. 2.15 построены для одной итерации отправки введенных пользователем данных и получения минимального достаточного подтверждения доставки от сервера для случаев использования традиционного веб-приложения (схема на рис. 2.9а), (2.2.5), (2.2.6) и АВП (схема на рис. 2.12а), (2.3.11), (2.3.12) на различных каналах связи. Построенные таблица и график отражают зависимость суммарного времени $t_{ij}^{отпр} + t_{ij}^{подтв}$ в секундах от пропускной способности канала V_C при фиксированном значении времени отклика сервера t_C и размере отправляемых данных $P_{ij}^{отпр}$.

Таблица 2.1. Время $t_{ij}^{отпр} + t_{ij}^{подтв}$ передачи с подтверждением введенных данных на сервер традиционным веб-приложением и АВП с резервированием, с

Тип веб-прило-жения	V_C, Кбит/с / Раз-мер $P_{ij}^{отпр}$, Кбайт	64	128	256	512	1024	2048	4096	10240	20480
Трад.	1	0,780	0,405	0,218	0,124	0,077	0,053	0,042	0,035	0,032
	10	1,905	0,968	0,499	0,264	0,147	0,089	0,059	0,042	0,036
	100	13,155	6,593	3,311	1,671	0,850	0,440	0,235	0,112	0,071
	1024	128,655	64,343	32,186	16,108	8,069	4,050	2,040	0,834	0,432
	10240	1280,655	640,343	320,186	160,108	80,069	40,050	20,040	8,034	4,032
АВП	1	0,280	0,155	0,093	0,061	0,046	0,038	0,034	0,032	0,031
	10	1,405	0,718	0,374	0,202	0,116	0,073	0,051	0,039	0,034
	100	12,655	6,343	3,186	1,608	0,819	0,425	0,227	0,109	0,069
	1024	128,155	64,093	32,061	16,046	8,038	4,034	2,032	0,831	0,430
	10240	1280,155	640,093	320,061	160,046	80,038	40,034	20,032	8,031	4,030

По данным таблицы 2.1 построен график (рис. 2.15) зависимости суммарного времени передачи данных на сервер и получения минимального необходимого подтверждения доставки данных от пропускной способности V_C канала связи при $t_C = 30$ мс для значений передаваемых данных 1 Кбайт, 10 Кбайт и 100 Кбайт (выделено в таблице) для традиционных веб-приложений и АВП.

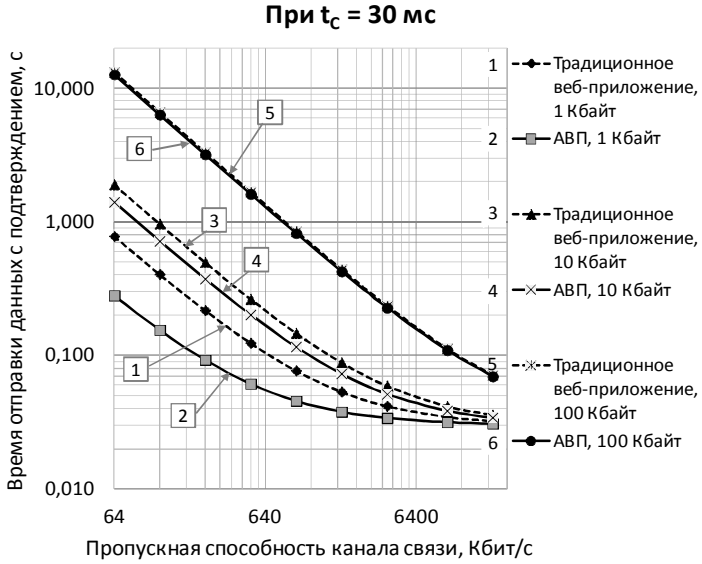


Рис. 2.15. Время передачи с подтверждением введенных данных на сервер традиционным веб-приложением и АВП с резервированием

Проведенные расчеты показывают, что при всех значениях пропускной способности V_C каналов связи от 64 Кбит/с до 20480 Кбит/с и времени отклика t_C в диапазоне от 30 мс (беспроводная связь по Wi-Fi 802.11g) до 800 мс (спутниковая связь) суммарные затраты времени на отправку данных на сервер и получение подтверждения при использовании АВП меньше, чем при использовании традиционных веб-приложений. Разница становится особенно заметной на загруженных каналах связи с малой пропускной способностью.

В реальных условиях численные значения эффективности АВП будут отличаться от расчетных, так как время отклика сервера t_C зависит не только от типа абонентского канала связи, но и от протяженности магистрального канала до сервера (которая индивидуальна для каждого клиента и размещения сервера), вносящей свою задержку. Однако это не меняет соотношения в пользу выигрыша во времени при работе АВП.

В случае, если в рамках сеанса связи между клиентом и сервером ИС осуществляется отправка нескольких сообщений введенных данных, суммарный выигрыш во времени от использования АВП увеличивается пропорционально количеству отправляемых сообщений.

При пропускной способности каналов связи более 20480 Кбит/с разница во времени передачи данных на сервер между традиционным веб-приложением и АВП практически исчезает.

На любых каналах связи сохраняется и является определяющим преимущество АВП перед традиционным веб-приложением в виде локального резервирования вводимых пользователем данных в $C^{пост.дан}$ и исключения вероятности их потери при нарушении соединения с сервером (работа в режиме 1).

2.5.2 Уменьшение трафика, количества запросов к серверу и времени загрузки при загрузке страниц АВП

Преимущества АВП перед традиционными веб-приложениями раскрываются в полной мере при плохом (режимы 2 и 3) и полностью отсутствующем (режим 4) соединении с сервером. Также режимы 2 и 3 позволяют существенно экономить трафик при оплате за потребленный объем.

Чтобы понять, как работают режимы 2 и 3 с частичным автономным использованием данных из локального кэша $C^{пост.прил}$, рассмотрим состав и структуру загружаемых страниц.

Каждая страница состоит из HTML-файла, который загружается первым. HTML-файл содержит ссылки на связанные элементы, загружаемые автоматически сразу после HTML-файла, такие как таблицы стилей CSS, программный код на языке JavaScript, различные изображения, шрифты, медиа-данные и другие файлы, без которых отображение всего этого набора пользователю в виде единой страницы невозможно.

По данным глобальной Интернет-статистики HTTP Archive по состоянию на 1 февраля 2015 года среднестатистический набор данных, загружаемый клиентом для просмотра одной веб-старницы в Интернете, имеет объем 1977 Кбайт и загружается за 96 запросов [52].

Из этого числа (рис. 2.16, 2.17):

- HTML-содержимое составляет 59 Кбайт и загружается примерно за 10 запросов,
- таблицы стилей CSS составляют 60 Кбайт и загружаются примерно за 7 запросов,
- программный код на языке JavaScript составляет 301 Кбайт и загружается примерно за 18 запросов,
- изображения составляют 1260 Кбайт и загружаются за 53 запроса,
- шрифты составляют 84 Кбайта и загружаются за 2 запроса,
- остальной загружаемый объем 213 Кбайт и 6 запросов приходятся на другие файлы.

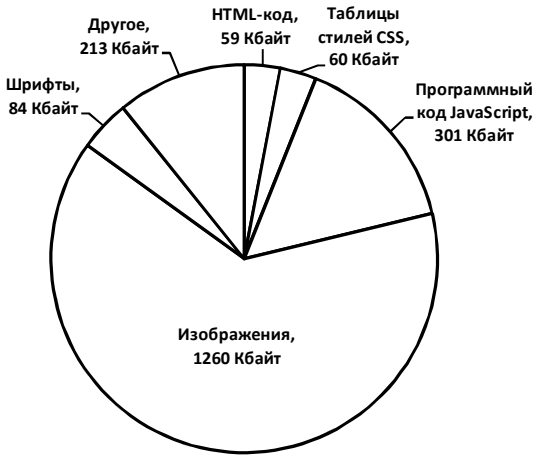


Рис. 2.16. Распределение загружаемого объема данных между элементами страницы

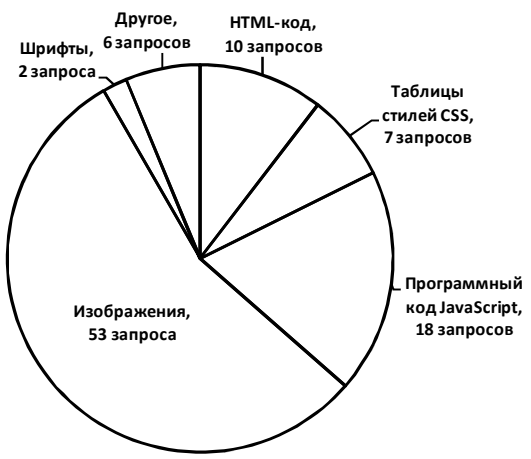


Рис. 2.17. Распределение общего количества запросов на загрузку данных с сервера между элементами страницы

Результаты примерно 46% запросов могли бы быть закэшированы локально [52].

Для сравнения, по состоянию на 15 ноября 2011 среднестатистический суммарный объем данных одной страницы составлял 702 Кбайта и загружался за 74 запроса [52].

Таким образом, за короткий срок среднестатистический размер страниц увеличился более чем в 2,8 раза, а количество элементов (запросов) страницы увеличилось почти на 30%.

Тенденция роста объема данных и количества элементов страниц прослеживается с момента начала наблюдения в 1995 году [30] до настоящего времени. Можно с уверенностью сказать, что данная тенденция продолжится и в будущем. Поэтому, несмотря на активное развитие аппаратных средств и сетей связи, необходимы программные решения для экономии трафика, снижения расхода энергии батареи мобильного устройства, резервирования данных и обеспечения автономной работы с данными, уже однажды полученными с сервера ИС.

Такие возможности открываются с внедрением АВП. Работа АВП с данными, получаемыми с сервера ИС, включает в себя два основных этапа: первичную инициализацию данных в $C^{пост.прил}$, которая выполняется один раз, и работу с использованием данных, загруженных в $C^{пост.прил}$. Отличия между режимами функционирования кэша $C^{пост.прил}$ 2, 3 и 4 обусловлены различным начальным объемом загружаемых с сервера данных и разными задачами использования мобильных устройств: в режимах 2 и 3 – это задачи ускорения обмена данными с сервером, экономии трафика и заряда батареи, защита от временных перебоев соединения; в режиме 4 – это задачи экономии заряда батареи и полностью автономной работы без обращения к серверу.

При расчетах расхода (2.2.1), (2.3.7) и экономии трафика (2.4.1), времени загрузки страниц (2.2.4), (2.3.10) при первичной инициализации и последующей работе АВП размер и состав главной страницы для всех сравниваемых приложений был принят равным измеренному среднестатистическому на 1 февраля 2015 года. Принятые объем и состав локально сохраняемых данных у АВП отличаются от традиционного веб-приложения в зависимости от режима работы $C^{пост.прил}$.

АВП, режим 2 – $C^{пост.прил} = 865$ Кбайт (все данные набора входят в общий объем данных главной страницы), $M^{кэш} = 1{,}5$ Кбайта при использовании сжатия GZIP [60] со сжатием на 70% (до сжатия 5 Кбайт), 1 дополнительный запрос.

АВП, режим 3 – $C^{пост.прил} = 10$ Мбайт (загружается дополнительно), $M^{кэш} = 1{,}5$ Кбайта при использовании сжатия GZIP со сжатием на 70% (до сжатия 5 Кбайт), 101 дополнительный запрос.

АВП, режим 4 – $C^{пост.прил} = 1$ Гбайт (загружается дополнительно), $M^{кэш} = 150$ Кбайт при использовании сжатия GZIP со сжатием на 70% (до сжатия 500 Кбайт), 10001 дополнительный запрос.

Важно отметить, что инициализация локального кэша приложения $C^{пост.прил}$ АВП происходит один раз при первом запуске АВП, а при последующих запусках начинается экономия трафика и появляются возможности автономной работы без подключения к серверу.

Традиционное веб-приложение загружает с сервера 1977 Кбайт, делает 96 запросов к серверу. Преобразование традиционного веб-приложения в АВП с теми же характеристиками объема и состава элементов одной страницы АВП при различных режимах работы кэша $C^{пост.прил}$ позволяет уменьшить количество запросов и объем загружаемых с сервера данных. Численные значения экономии варьируются в зависимости от выбранного режима работы кэша.

АВП, режим 2 с изображениями – загружает с сервера 1113,5 Кбайт, делает 52 запроса к серверу, $B = 0{,}44$, $H = 0{,}46$.

АВП, режим 3 с изображениями – загружает с сервера 913,5 Кбайт, делает 44 запроса к серверу, $B = 0{,}54$, $H = 0{,}54$.

АВП, режимы 2 и 3, только текст – загружает с сервера 60,5 Кбайт, делает 11 запросов к серверу, $B = 0{,}97$, $H = 0{,}89$.

АВП, режим 4, автономный – загружает с сервера 0 Кбайт, делает 0 запросов к серверу, $B = 1$, $H = 1$.

АВП, режим 4, с фоновой проверкой манифеста кэша – загружает с сервера 150 Кбайт, делает 1 запрос к серверу, $B = 0{,}92$, $H = 0{,}99$. При недоступности сервера продолжается работа со старым $M^{кэш}$. Расчет времени фоновой проверки манифеста кэша в автономном режиме приведен в таблицах и на графиках для демонстрации наличия проверки обновлений локальных данных АВП.

В таблице 2.2, а также на построенных по ее данным графиках на рисунках 2.18 и 2.19 показан суммарный расход трафика традиционными веб-приложениями (2.2.1) и АВП (2.3.7) при работе в различных режимах $C^{пост.прил}$ в зависимости от количества сеансов работы. Таблицы и графики построены с учетом того, что клиент загружает различные страницы, соответствующие по размеру и составу загружаемых данных среднестатистическим страницам в Интернете, а также что загрузки происходят в разных сеансах, поэтому данные этих страниц не содержатся в $C^{сеанс}$. При расчетах расхода и экономии трафика АВП в режимах работы с изображениями и с загрузкой с сервера только текста сравнение с традиционным веб-приложением происходит в его единственном стандартном режиме (текст и изображения). Традиционные веб-приложения не имеют режима «только текст», а при принудительном отключении изображений в браузере часто становятся неработоспособными, так как используют изображения в качестве элементов интерфейса. Режимы работы «только текст» АВП не являются отключением изображений в веб-приложении. В таких режимах АВП использует служебные изображения, предварительно сохраненные в $C^{пост.прил}$ при инициализации, а с сервера загружает только текстовые данные для просмотра или обработки на стороне клиента.

Таблица 2.2. Суммарный расход трафика традиционным веб-приложением и АВП при работе в различных режимах $C^{пост.прил}$, Кбайт

Тип веб-приложения и режим работы \ Число сеансов	1	10	50	100	500	1000
Традиционное веб-приложение	1977	19770	98850	197700	988500	1977000
АВП, режим 2, только текст	1978,5	2523	4943	7968	32168	62418
АВП, режим 2, с изображениями	1978,5	12000	56540	112215	557615	1114365
АВП, режим 3, только текст	12218,5	12763	15183	18208	42408	72658
АВП, режим 3, с изображениями	12218,5	20440	56980	102655	468055	924805
АВП, режим 4, автономный	1050703	1050703	1050703	1050703	1050703	1050703
АВП, режим 4, фоновая проверка манифеста кэша	1050703	1052053	1058053	1065553	1125553	1200553

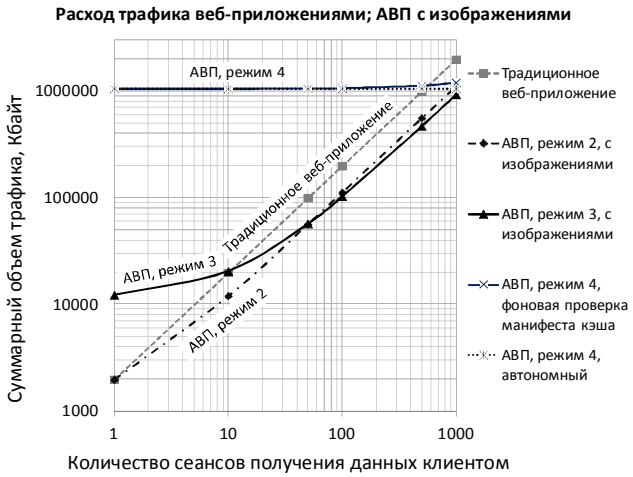


Рис. 2.18. Суммарный расход трафика традиционным веб-приложением и АВП при работе с изображениями

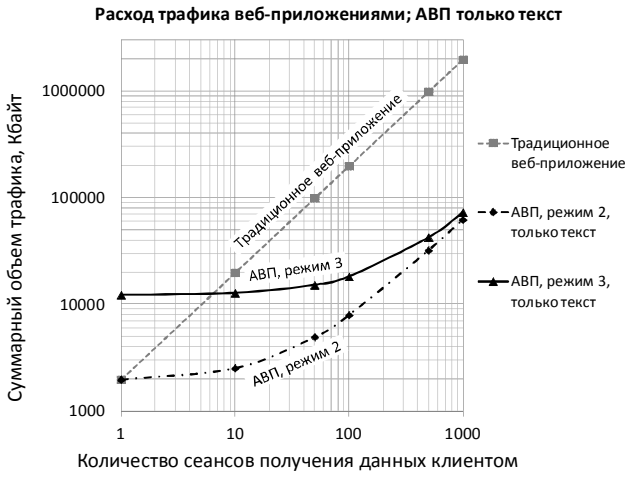


Рис. 2.19. Суммарный расход трафика традиционным веб-приложением и АВП при работе АВП в режиме загрузки только текстовой информации

АВП, режим 4 (автономный и фоновая проверка манифеста кэша) показаны на графике для демонстрации того, что, даже будучи настроенным для автономной работы с большим начальным кэшем, позволяет экономить трафик. Хотя основной функцией режима 4 АВП является не экономия трафика, а обеспечение полностью автономной работы с данными, когда подключение к серверу недоступно.

В таблице 2.3, а также на построенных по ее данным графиках на рисунках 2.20 и 2.21 показана суммарная экономия трафика АВП (2.4.1) при работе в различных режимах $C^{пост.прил}$ в сравнении с традиционным веб-приложением в зависимости от количества сеансов работы.

Таблица 2.3. Суммарная экономия трафика АВП по сравнению с традиционным веб-приложением при работе в различных режимах $C^{пост.прил}$, Кбайт

Тип веб-приложения и режим работы \ Число сеансов	1	10	50	100	500	1000
АВП, режим 2, только текст	-1,5	17247	93907	189732	956332	1914582
АВП, режим 2, с изображениями	-1,5	7770	42310	85485	430885	862635
АВП, режим 3, только текст	-10241,5	7007	83667	179492	946092	1904342
АВП, режим 3, с изображениями	-10241,5	-670	41870	95045	520445	1052195
АВП, режим 4, автономный	-1048726	-1030933	-951853	-853003	-62203	926297
АВП, режим 4, фоновая проверка манифеста кэша	-1048726	-1032283	-959203	-867853	-137053	776447

Положительные значения экономии трафика означают, что начиная с определенного количества сеансов АВП расходует меньше трафика, чем традиционное веб-приложение.

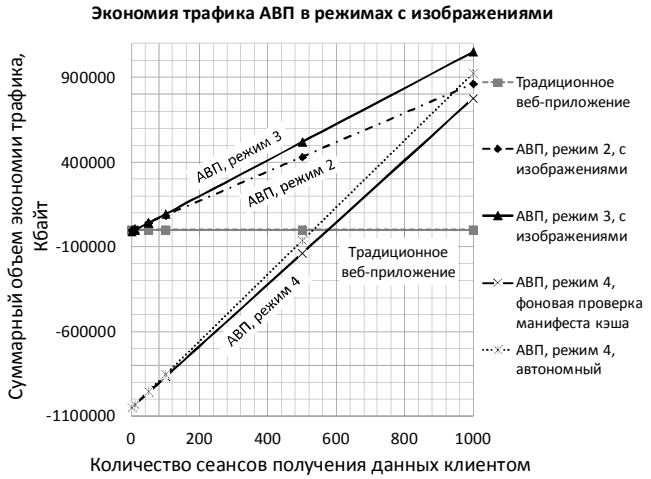


Рис. 2.20. Суммарная экономия трафика АВП при работе с изображениями в сравнении с традиционным веб-приложением

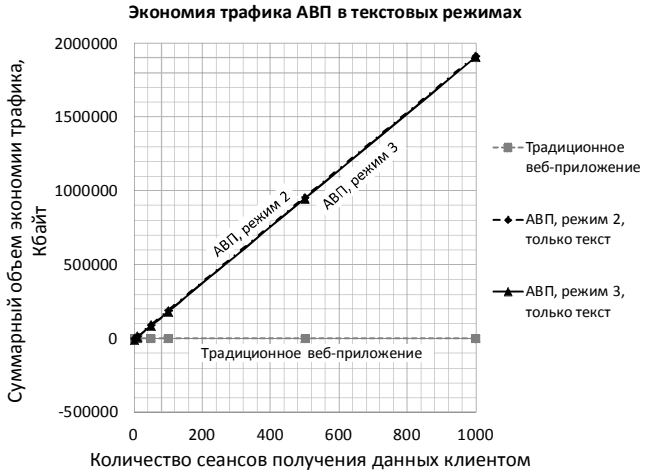


Рис. 2.21. Суммарная экономия трафика АВП при работе в режиме загрузки только текстовой информации в сравнении с традиционным веб-приложением

Суммарная экономия трафика АВП по сравнению с традиционным веб-приложением возрастает пропорционально количеству сеансов работы пользователя с данными ИС посредством АВП.

В таблице 2.4, а также построенном по ее данным графике на рисунке 2.22 показано, сколько времени занимает первичная загрузка данных с сервера для инициализации $C^{пост.прил}$ в АВП (рис. 2.10а), (2.3.10) при фиксированном времени отклика сервера t_C на различных каналах связи в сравнении с загрузкой средней страницы традиционного веб-приложения (рис. 2.8а), (2.2.4).

Время в данной таблице отражает продолжительность полной загрузки данных всех файлов первой отображаемой страницы и, для АВП, всех файлов, загружаемых дополнительно в $C^{пост.прил}$. При этом файлы, относящиеся к первой странице АВП, загружаются приоритетно по отношению к другим файлам в $M^{кэш}$, поэтому главные страницы АВП и традиционного веб-приложения отображаются клиенту одновременно.

Построенные таблица и график отражают зависимость суммарного времени в секундах загрузки данных с сервера – необходимых для отображения главной страницы в случае традиционного веб-приложения; отображения главной страницы и первоначальной инициализации $C^{пост.прил}$ в режимах работы кэша 2, 3 и 4 в случае АВП – от пропускной способности канала V_C при фиксированном значении времени отклика сервера t_C для сравнения традиционного веб-приложения и АВП.

Таблица 2.4. Время получения данных с сервера традиционным веб-приложением и АВП при инициализации $C^{пост.прил}$, с

V_C, Кбит/с / Тип веб-приложения	64	128	256	512	1024	2048	4096	10240	20480
Традиционное	248,115	124,553	62,771	31,881	16,435	8,713	4,851	2,535	1,762
АВП, режим 2	248,303	124,646	62,818	31,904	16,447	8,719	4,854	2,536	1,763
АВП, режим 3	1529,323	765,666	383,838	192,924	97,467	49,739	25,874	11,556	6,783
АВП, режим 4	131438,885	65769,948	32935,479	16518,244	8309,627	4205,319	2153,164	921,872	511,441

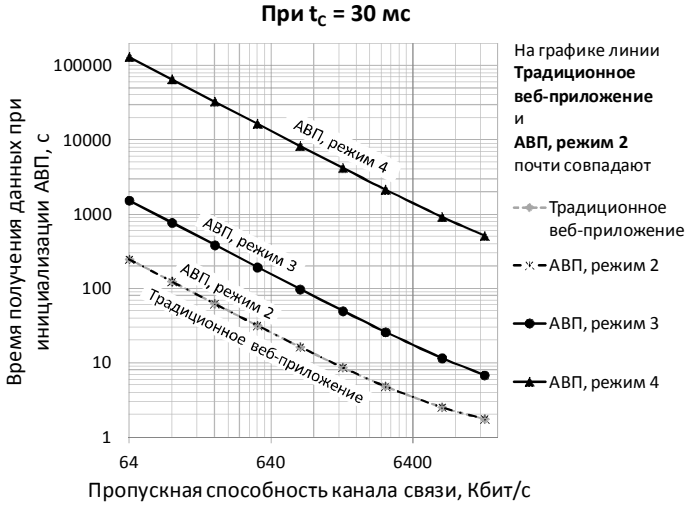


Рис. 2.22. Время получения данных с сервера традиционным веб-приложением и АВП при инициализации $C^{пост.прил}$

Из приведенных расчетов в таблице 2.4 и на графике на рисунке 2.22 видно, что при использовании каналов с малой пропускной способностью приходится долго ждать окончания загрузки объема данных, равного даже среднестатистической странице в 1977 Кбайт, не говоря уже о данных объемом в 1 Гбайт.

Учитывая большой объем предварительно загружаемых данных АВП, первый запуск такого приложения рекомендуется осуществлять при сетевом подключении с большой пропускной способностью V_C и минимальным временем отклика сервера t_C. Из таблицы 2.4 видно, что время предварительной загрузки данных АВП менее 10 минут при использовании канала с временем отклика $t_C = 30$ мс достигается при: $V_C \geq 64$ Кбит/с для АВП, режим 2; $V_C \geq 256$ Кбит/с для АВП, режим 3; $V_C \geq 20480$ Кбит/с для АВП, режим 4.

В отличие от традиционных веб-приложений, скорость работы которых неизменно зависит от характеристик канала связи, АВП после предварительной загрузки в локальное

хранилище $C^{пост.прил}$ необходимых данных с использованием каналов с большой пропускной способностью позволяют пользователю продолжать комфортную работу даже при наличии только медленных и нестабильных каналов связи или вовсе в автономном режиме.

Для количественной оценки выгоды от использования различных режимов работы локального кэша АВП рассмотрим таблицу 2.5 и графики на рисунках 2.23, 2.24, в которых показано время получения данных с сервера АВП при работе с использованием данных из $C^{пост.прил}$ (рис. 2.11а) при разном времени отклика сервера t_C на различных каналах связи в сравнении с временем загрузки средней страницы традиционного веб-приложения (рис. 2.8а). Время в данной таблице отражает продолжительность полной загрузки данных всех файлов отображаемой страницы. При этом в различных режимах АВП различная доля необходимых файлов берется из предварительно инициализированного кэша $C^{пост.прил}$.

Построенная таблица и графики отражают зависимость времени t_D (2.2.4), (2.3.10) – загрузки одной последовательности d_i данных с сервера, необходимой для отображения страницы традиционного веб-приложения или АВП, от пропускной способности канала V_C при фиксированном значении времени отклика сервера t_C для сравнения традиционного веб-приложения и АВП.

Таблица 2.5. Время t_D получения данных с сервера традиционным веб-приложением и АВП при работе с использованием данных из $C^{пост.прил}$, с

V_C, Кбит/с / Тип веб-приложения, режим работы	64	128	256	512	1024	2048	4096	10240	20480
Традиционное веб-приложение	248,115	124,553	62,771	31,881	16,435	8,713	4,851	2,535	1,762
АВП, режим 2, с изображениями	139,758	70,164	35,367	17,968	9,269	4,920	2,745	1,440	1,005
АВП, режим 3, с изображениями	114,668	57,574	29,027	14,753	7,617	4,048	2,264	1,194	0,837
АВП, режимы 2 и 3, только текст	7,893	4,111	2,221	1,275	0,803	0,566	0,448	0,377	0,354
АВП, режим 4, автономный	0	0	0	0	0	0	0	0	0
АВП, режим 4, фоновая проверка манифеста кэша	18,780	9,405	4,718	2,374	1,202	0,616	0,323	0,147	0,089

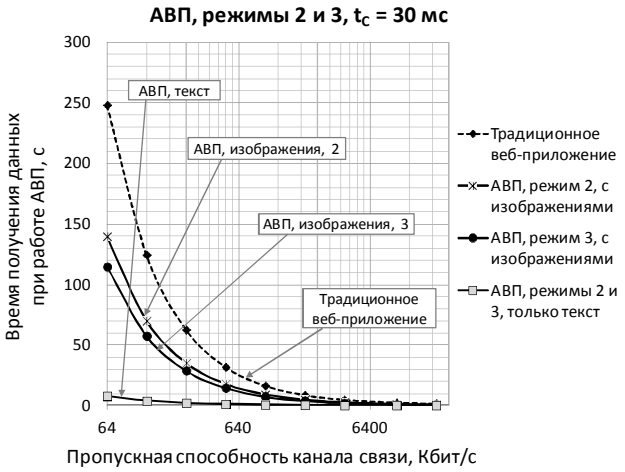


Рис. 2.23. Время получения данных с сервера традиционным веб-приложением и АВП при работе в режимах 2 и 3 с использованием данных из $C^{пост.прил}$

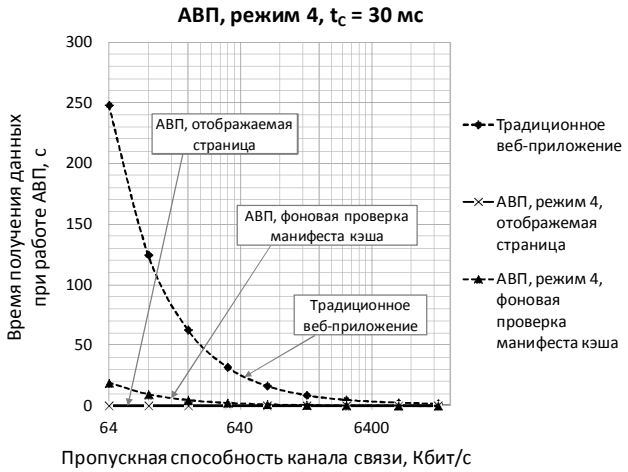


Рис. 2.24. Время получения данных с сервера традиционным веб-приложением и АВП при работе в режиме 4 с использованием данных из $C^{пост.прил}$

Приведенные в таблице 2.5 результаты расчетов и графики на рисунках 2.23, 2.24 показывают, что использование АВП позволяет значительно ускорить загрузку страниц веб-приложений клиентом за счет постоянного локального хранения части повторно загружаемых данных в $C^{пост.прил}$.

Численные значения экономии времени АВП зависят от режима работы $C^{пост.прил}$:

- АВП, режим 2, с изображениями – до 44%,
- АВП, режим 3, с изображениями – до 54%,
- АВП, режимы 2 и 3, только текст – до 97%,
- АВП, режим 4, отображаемая страница – до 100%.

Предлагаемые АВП наиболее эффективны, когда пользователь продолжительное время работает с одной и той же ИС. Для оптимального удовлетворения потребностей используются разные режимы $C^{пост.прил}$:

- режимы 2 и 3, с изображениями – для клиентов различных ИС (ERP, PLM и других) и работы с технической документацией с изображениями;
- режимы 2 и 3, только текст – для клиентов различных ИС (ERP, PLM и других) и СУБД, где изображения, кроме служебных, не используются;
- режим 4 – для автономной работы с данными ДЗЗ, технической документацией или учебными материалами с большим объемом изображений, в том числе без подключения к серверу.

Как видно из графиков на рисунках 2.23, 2.24 наибольшие абсолютные значения снижения времени загрузки данных (2.4.4) достигаются на каналах с малой пропускной способностью V_C. Это позволяет мобильным пользователям работать с АВП вне стен предприятия при использовании произвольных каналов связи, которые могут иметь малую пропускную способность или быть ненадежными с частыми обрывами соединения. Качество работы будет таким же, как если бы работа происходила за стационарным рабочим местом.

Пользователю больше не придется ждать загрузки страниц так долго, как это было с традиционными веб-приложениями. Если нужно что-то срочное и непредвиденное, у АВП есть режимы 2 и 3 с загрузкой только текста, а если нужны изображения, то в режимах 2 и 3 с изображениями в АВП все работает гораздо быстрее, чем в традиционном веб-приложении. При отсутствии соединения с сервером АВП обеспечивает полностью автономный режим 4.

Приведенные расчеты учитывают размер передаваемых данных, количество элементов данных и характеристики каналов передачи данных, что позволяет оценить минимальное необходимое время для загрузки. В расчетах принято, что сервер всегда отвечает в течение фиксированного времени отклика для каждого типа канала C без дополнительных задержек, а веб-клиент моментально отображает загруженные данные.

На практике из-за воздействий таких факторов, как загруженность сервера и клиента процессами, необходимость предварительной обработки отправляемых клиенту данных, необходимость предварительной обработки клиентом полученных с сервера данных, непостоянство характеристик каналов передачи данных из-за различной загрузки и других факторов, время отправки и получения данных с сервера на клиент и с клиента на сервер будет несколько больше расчетного, но соотношения времени загрузки данных традиционным веб-приложением и АВП останутся прежними.

Возможности полностью автономной работы АВП не зависят от факторов, влияющих на время загрузки данных с сервера. АВП – это такое клиент-серверное приложение, которое может становиться полностью клиентским в некоторых ситуациях.

Таким образом, АВП превосходит традиционное веб-приложение по надежности передачи данных на сервер за счет резервирования вводимых пользователем данных (режим 1), по скорости загрузки страниц и обеспечению работоспособности при нестабильных соединениях или соединениях с низкой пропускной способностью (режимы 2 и 3), а также, в отличие от традиционного веб-приложения, обладает возможностями полностью автономной работы (режим 4), в чем может конкурировать даже с нативными приложениями.

ГЛАВА 3. ПРОГРАММНЫЙ КОМПЛЕКС АВП

Согласно изложенному в разделе 1.4 функционирование АВП состоит в выполнении следующего набора процедур:

- **отправка на сервер с локальным сохранением вводимых данных;**
- **восстановление на сервер локально сохраненных данных;**
- **обновление данных в локальном хранилище Application Cache.**

В основе этих процедур лежат описанные в разделе 1.3 процессы:

- **отправка данных на сервер;**
- **получение данных от сервера;**
- **актуализация локального хранилища данных.**

В рамках каждой из процедур выполняется несколько процессов предложенного метода функционирования клиент-серверной системы (таблица 3.1).

Таблица 3.1. Использование базовых процессов процедурами АВП

Процессы / Процедуры	Отправка данных на сервер		Получение данных от сервера	Актуализация локального хранилища данных
	Запросов к серверу на получение данных	Введенных данных		
Процедура отправки на сервер с локальным сохранением вводимых данных	Запрос подтверждения доставки введенных данных	Сохранение введенных пользователем данных в Local Storage, отправка на сервер, удаление доставленных данных	Получение подтверждения от сервера доставки введенных данных	Изменение МК на сервере
Процедура восстановления на сервер локально сохраненных данных	Запрос проверки доступности сервера, запрос подтверждения доставки введенных данных	Чтение сохраненных введенных пользователем данных из Local Storage,отправка на сервер, удаление доставленных данных	Получение подтверждения от сервера доставки введенных данных	Изменение МК на сервере
Процедура обновления данных в локальном хранилище Application Cache	Запрос нового МК, запросы проверки обновления сохраненных элементов веб-страниц	–	Получение МК, сообщений обо всех проверяемых элементах, обновленных элементов веб-страниц	Обновление изменившихся элементов веб-страниц в Application Cache при изменении МК на сервере

3.1 Общая схема функционирования АВП

Общая схема функционирования АВП показана на рис. 3.1.

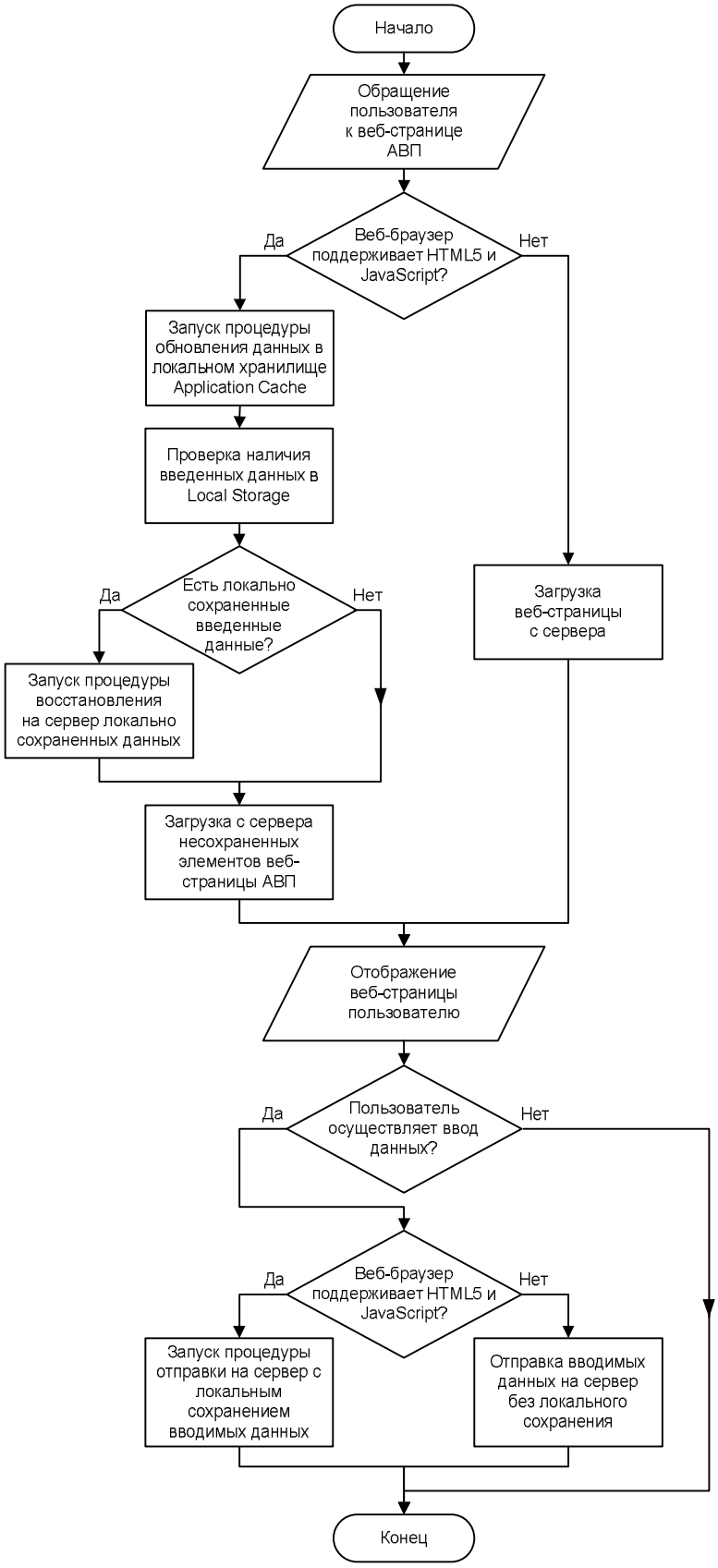


Рис. 3.1. Общая схема функционирования АВП

При обращении пользователя к любой странице АВП, если браузер поддерживает HTML5 и JavaScript, проверяется файл cache.manifest, чтобы выяснить, было ли уже установлено данное приложение в постоянный кэш приложений Application Cache браузера. Если клиентская часть АВП установлена, пользователю открывается страница, с которой он может работать. Если АВП не установлено в локальный кэш приложений браузера или имеет устаревшую версию, то происходит загрузка новой версии приложения в соответствии с актуальным на данный момент файлом cache.manifest.

Загруженное приложение проверяет, есть ли в локальном хранилище данных Local Storage браузера записи о зарезервированных введенных данных. Если приложение находит сохраненные данные, введенные пользователем, то сообщает об этом пользователю и начинает автоматическую фоновую процедуру восстановления сохраненных данных на сервер. По окончании отправки данных на сервер пользователю выдается сообщение о результате.

Если пользователь решает отредактировать страницу или оставить к ней комментарий, то программа запускает процедуру автоматического резервирования вводимых пользователем данных, чтобы предотвратить их потерю при нарушении соединения с сервером. Данная процедура выполняется в фоновом режиме до тех пор, пока не случается сбой и не появляется необходимость сообщить пользователю о том, что данные не смогли быть переданы на сервер и были сохранены локально.

Пользователь может продолжать работу с системой даже в случае, если соединение с сервером потеряно. Все страницы, где пользователь был, сохранены в локальном кэше приложения до тех пор, пока не будут обновлены. Данные, которые вводятся в веб-формы, тоже сохранены локально. Все сохраненные данные останутся на устройстве пользователя даже после закрытия браузера до их отправки на сервер или обновления с сервера.

Далее рассмотрим подробнее процедуры АВП, показанные в таблице 3.1 и на рис. 3.1.

3.1.1 Процедура отправки на сервер с локальным сохранением вводимых данных

Схема процедуры отправки на сервер с локальным сохранением вводимых данных показана на рис. 3.2.

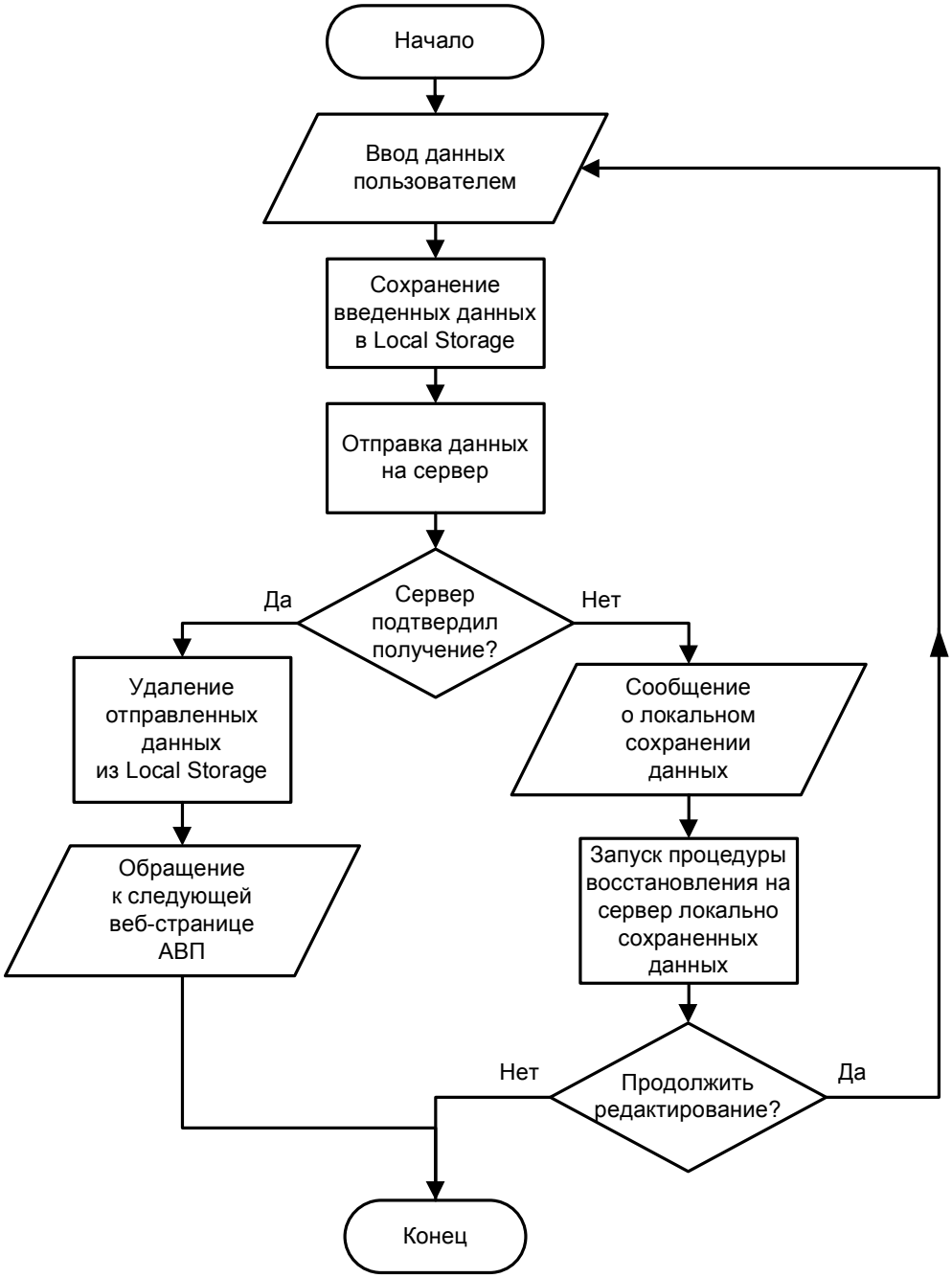


Рис. 3.2. Схема процедуры отправки на сервер с локальным сохранением вводимых данных

Набор разработанных средств для реализации процедуры отправки на сервер с локальным сохранением вводимых данных включает в себя модули: вывода информационных сообщений пользователю, контроллер резервирования данных в Local Storage, проверки доступности сервера, ответчик доступности сервера, асинхронный приемник данных.

Пока есть соединение с сервером, пользователь даже не замечает того, что работает с АВП. Средства резервирования данных выполняют функции посредника между вводом данных пользователем и их отправкой на сервер.

Поскольку до получения подтверждения получения данных от сервера неизвестно, будут ли они успешно доставлены на сервер, все вводимые пользователем данные автоматически заносятся в локальное хранилище Local Storage перед отправкой на сервер.

После подтверждения получения сервером отправленных пользователем данных соответствующая запись в локальном хранилище Local Storage удаляется, происходит обращение к следующей странице (рис. 3.1), которое начинается для АВП с запуска процедуры обновления данных в хранилище Application Cache, поскольку ввод данных пользователем мог затронуть закэшированные данные АВП.

При отсутствии соединения с сервером в момент отправки заполненной веб-формы и, соответственно, неполучении подтверждения от сервера, введенные данные сохраняются в локальном хранилище Local Storage в ожидании восстановления соединения с сервером. Пользователю выводится сообщение о том, что сервер недоступен, данные были сохранены локально и будут отправлены на сервер при возобновлении соединения, и что пользователь может продолжить работу с ранее посещенными страницами, включая даже ввод данных в автономном режиме. Это возможно, поскольку и данные, и программный код клиентской части АВП сохраняются локально на устройстве пользователя даже при закрытии браузера. В случае закрытия браузера при следующем заходе на сайт АВП будет автоматически запущена процедура проверки наличия сохраненных введенных пользователем данных и их восстановления на сервер.

Если данные были успешно получены сервером, но разрыв соединения произошел в момент отправки сервером подтверждения, и подтверждение не было получено клиентом, это вызовет повторную отправку тех же данных при восстановлении соединения и, в результате, должно вызвать дублирование их на сервере. Но в разработанной системе обработка подобных ситуаций реализована с помощью серверного механизма контроля и учета версий для всех текстовых материалов, что позволяет избежать потери внесенных изменений, дублирования информации в БД и коллизий совместного редактирования.

3.1.2 Процедура восстановления на сервер локально сохраненных данных

Схема процедуры восстановления на сервер локально сохраненных данных показана на рис. 3.3.

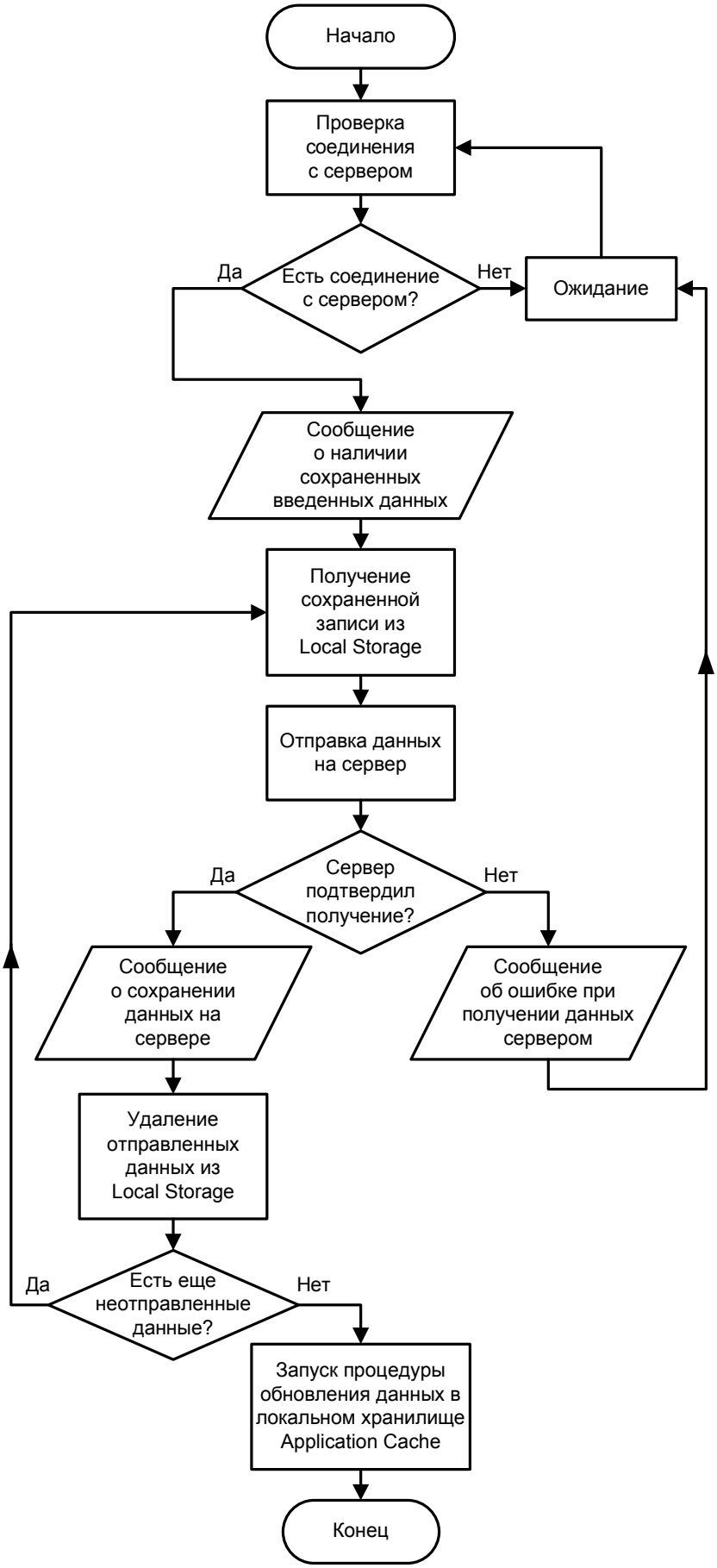


Рис. 3.3. Схема процедуры восстановления на сервер локально сохраненных данных

После того как зафиксирован разрыв соединения с сервером, а введенные пользователем данные были сохранены локально, что делает контроллер резервирования вводимых данных, запускается процедура восстановления на сервер локально сохраненных данных. В рамках этой процедуры модуль проверки доступности сервера автоматически посылает на сервер запросы к ответчику доступности сервера с заданной в настройках периодичностью, пока не получит ответ сервера, что означает восстановление соединения.

После этого пользователю выводится информационное сообщение о том, что есть локально сохраненные введенные пользователем данные и система отправит их на сервер.

Процедура автоматического восстановления локально сохраненных данных на сервер сопровождается выводом пользователю информационных сообщений о ходе процесса отправки данных на сервер и о завершении процесса отправки.

Если в процессе отправки сервер снова станет недоступен, то пользователю будет показано соответствующее сообщение и данные будут отправлены позже.

При отправке сохраненных данных на сервер, сохраненные записи из локального хранилища Local Storage посылаются по очереди на сервер и удаляются из хранилища при получении подтверждения сервера о получении (ответ сервера «200 OK»).

После отправки всех сохраненных в Local Storage записей запускается процедура обновления данных в локальном хранилище Application Cache, так как ввод данных пользователем мог вызвать изменение сохраненных в Application Cache данных.

Таким образом, реализация клиентских и серверных компонентов для резервирования вводимых пользователем данных позволяет полностью исключить потерю данных при разрыве соединения с сервером и обеспечить их отправку на сервер при восстановлении соединения.

3.1.3 Процедура обновления данных в локальном хранилище Application Cache

Схема процедуры обновления данных в локальном хранилище Application Cache показана на рис. 3.4.

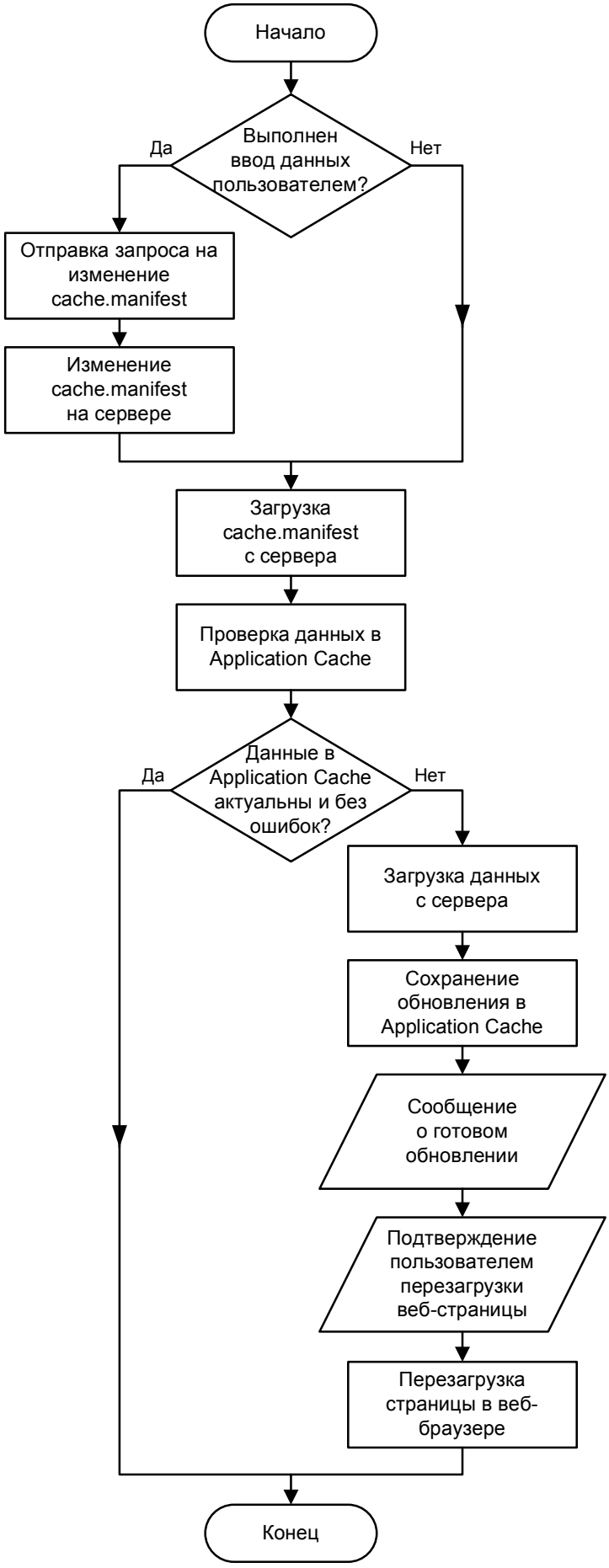


Рис. 3.4. Схема процедуры обновления данных в локальном хранилище Application Cache

Автономная работа клиентской части АВП при потере соединения с сервером и преимущества АВП перед традиционным веб-приложением по уменьшению количества запросов, снижению объема трафика, уменьшению суммарного времени ожидания ответа сервера, времени использования каналов и времени загрузки страниц основывается на использовании локального хранилища HTML5 – Application Cache, обеспечивающего долговременное хранение на клиентском устройстве программного кода клиентской части АВП и подборки получаемых с сервера данных, необходимых для работы.

Предлагаемый набор средств включает в себя контроллер обновления МК, актуализатор МК, МК – файл cache.manifest, который подключается на всех страницах, относящихся к веб-приложению и подлежащих локальному сохранению, а также содержит список других ресурсов веб-приложения, которые нужно сохранить.

Установка и обновление клиентской части АВП

Первичная инициализация локального хранилища Application Cache происходит при первом обращении к странице, на которой подключен МК.

Для пользователя это происходит незаметно, но все статические ресурсы, перечисленные в МК, скачиваются с сервера и размещаются в локальное хранилище Application Cache.

Также в локальном хранилище по мере их посещения пользователем размещаются страницы веб-сайта, в которых есть вызов cache.manifest, указывающий на принадлежность этих страниц к автономному веб-приложению.

При обращении к любой странице сначала проверяется МК на сервере, и если изменений в нем нет, все перечисленные в нем ресурсы и вызывавшие его страницы, посещенные ранее, подаются пользователю из локального хранилища. При отсутствии соединения с сервером и невозможности получения МК клиентское приложение подается пользователю из локального хранилища аналогично ситуации, когда МК не изменен.

Если полученный с сервера МК отличается от имеющегося на устройстве пользователя: старые ресурсы, которых нет в новом МК, удаляются из локального хранилища; новые ресурсы, перечисленные в МК, которые не загружались ранее, загружаются с сервера; для уже загружавшихся ранее и хранящихся локально ресурсов выполняется проверка их изменения с помощью отправки серверу запросов с заголовками If-Modified-Since [49, 66], что позволяет загружать только измененные ресурсы.

Ресурсы, которые уже содержались в локальном хранилище и не были изменены на сервере, продолжают подаваться пользователю из локального хранилища.

Изменение МК на сервере может быть вызвано администратором, программами на сервере или пользователями при отправке на сервер вводимых данных.

Обновление МК по команде клиентов при вводе данных позволяет всем клиентам ИС своевременно узнать о появлении обновлений в данных и загрузить их.

После отправки на сервер данных, введенных пользователем, клиентский контроллер обновления МК делает запрос к серверному актуализатору МК, который обновляет запись о времени последнего изменения в МК. Обновленный МК загружается с сервера и начинается обновление данных в локальном хранилище Application Cache, что происходит, если новый МК отличается от имеющегося на устройстве пользователя.

Удаление клиентской части АВП командой с сервера

В случае, если необходимо удалить веб-приложение из локального хранилища устройства пользователя, это можно сделать дистанционно в рамках процедуры обновления данных в локальном хранилище Application Cache, послав клиенту новый МК, в котором больше не будет указаний на необходимость локального сохранения каких-либо ресурсов.

Некоторым браузерам для удаления АВП из локального хранилища достаточно удаления МК с сервера, что приводит к игнорированию сохраненной программы в локальном хранилище и прекращению ее использования.

Чтобы гарантированно удалить сохраненную клиентскую часть веб-приложения из локального хранилища на устройстве пользователя, необходимо:

- изменить МК на сервере, чтобы в секции CACHE было пусто, что приведет к удалению всех сохраненных ресурсов из локального кэша;
- убрать вызов МК со страниц веб-приложения, чтобы браузер не использовал локальное хранилище Application Cache.

Удаление командой с сервера клиентской части веб-приложения и полученных с сервера данных из локального хранилища Application Cache никак не влияет на зарезервированные данные пользовательского ввода, хранящиеся в локальном хранилище Local Storage, что является защитой от случайных воздействий.

Таким образом, использование МК позволяет обеспечивать долговременное сохранение на стороне клиента получаемых с сервера данных и управление этим процессом с сервера, а разработанный компонент для актуализации МК при изменении данных на сервере позволяет совмещать автономную работу АВП с сохраненными данными с автоматическим поддержанием актуальности этих данных.

3.2 Программная реализация

Для реализации автоматического аварийного резервирования вводимых пользователем данных, возможности автономной работы с загруженными ранее с сервера данными (и, как

следствие, улучшения рассмотренных выше характеристик обмена данными между клиентом и сервером), обеспечения кроссплатформенности разработан ряд программных модулей, которые могут быть внедрены в ГИС с использованием данных ДЗЗ: клиентские библиотеки JavaScript для jQuery для резервирования пользовательского ввода и управления обновлением локального кэша АВП; средства серверной обработки AJAX-запросов и обновления манифеста кэша [7, 8].

3.2.1 Архитектура разработанных программных модулей

Архитектура и схема взаимодействия разработанных модулей между собой и с пользователем системы показаны на рис. 3.5 [9].

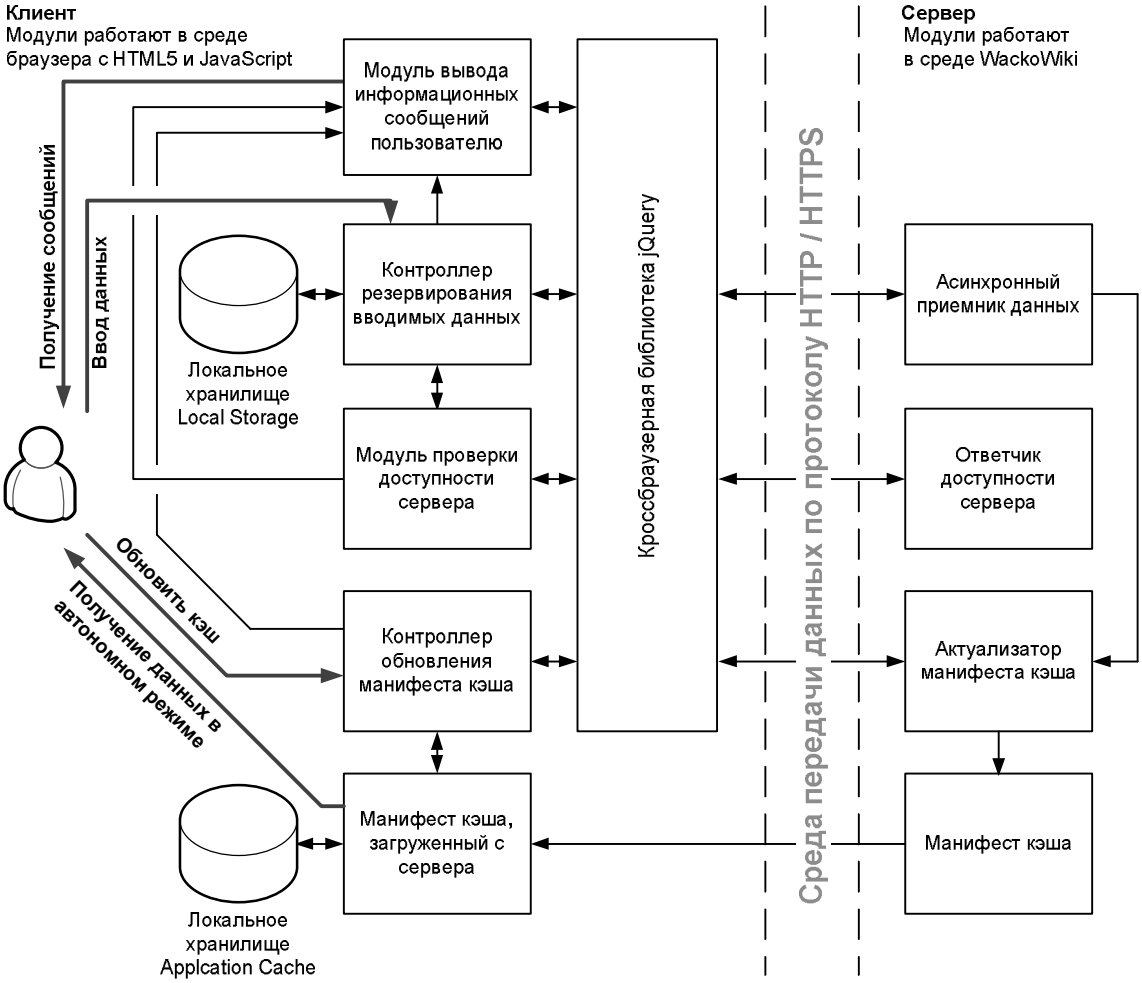


Рис. 3.5. Архитектура разработанных программных модулей и их взаимодействие между собой и с пользователем

Клиентская часть системы представлена четырьмя модулями JavaScript для jQuery и загруженной с сервера копией актуального МК, что обеспечивает ее работу на любых современных клиентских устройствах и унификацию взаимодействия с пользователем. Серверная часть представлена тремя модулями и МК. Модули на стороне клиента и сервера реализуют описанные в разделе 1.3 процессы.

3.2.2 Модули на стороне клиента

При реализации модулей на стороне клиента используются возможности современных браузеров, предоставляемые стандартом HTML5 [36], такие как долговременные хранилища данных Local Storage и Application Cache на стороне клиента [56, 79], кроссбраузерная библиотека jQuery для обеспечения унификации программного кода разработанных модулей на языке JavaScript для разных клиентских платформ и технология AJAX для организации асинхронной передачи данных между клиентом и сервером.

В таблице 3.2 показано использование названных средств при реализации модулей АВП.

Таблица 3.2. Использование модулями АВП на стороне клиента технологий стандарта HTML5 и возможностей современных браузеров

Технология / Модуль АВП	HTML5 Local Storage	HTML5 Application Cache	JavaScript jQuery	JavaScript AJAX
Модуль вывода информационных сообщений пользователю	–	–	Вывод сообщений пользователю без перезагрузки страницы в различных браузерах	–
Контроллер резервирования вводимых пользователем данных	Сохранение вводимых данных, извлечение для отправки на сервер после восстановления соединения, удаление после успешной доставки	–	Обеспечение работы с Local Storage и отправки данных в различных браузерах	Фоновая отправка ранее сохраненных данных без перезагрузки открытой страницы
Модуль проверки доступности сервера	–	–	Обеспечение проверки восстановления соединения с сервером в различных браузерах	Фоновая проверка восстановления соединения с сервером без перезагрузки открытой страницы
Контроллер обновления манифеста кэша	–	Инициализация обновления данных в Application Cache при вводе данных пользователем или получении нового МК	Обеспечение инициализации обновления данных в Application Cache в различных браузерах	Фоновые запросы к актуализатору МК для запуска принудительной актуализации хранилища Application Cache
Манифест кэша, загруженный клиентом с сервера	–	Управление долгосрочным кэширование данных, полученных с сервера, в Application Cache	–	–

Модуль вывода информационных сообщений пользователю

Модуль вывода информационных сообщений пользователю предназначен для отображения пользователю и вывода в консоль браузера сообщений, посылаемых другими модулями АВП без перезагрузки просматриваемой пользователем страницы.

Данный модуль участвует в процедурах: отправки на сервер с локальным сохранением вводимых данных, восстановления на сервер локально сохраненных данных и обновления данных в локальном хранилище Application Cache.

Взаимодействует с модулями:

- контроллером резервирования вводимых данных;
- модулем проверки доступности сервера;
- контроллером обновления манифеста кэша.

Для первых двух модулей выводит сообщения модулей пользователю и в консоль браузера, а для третьего дополнительно к названному отображает инструмент для подтверждения пользователем применения загруженного обновления локального хранилища Application Cache или для принудительного запроса обновления.

Модуль реализован на языке программирования JavaScript с использованием функций библиотеки jQuery, применяющуюся в АВП для обеспечения кроссплатформенности функционирования и асинхронного обмена данными с сервером в фоновом режиме, что делает возможным использование АВП на любых устройствах и автоматическое восстановление сохраненных во время сбоя соединения введенных пользователем данных при возобновлении соединения с сервером.

Основной задачей кроссбраузерной библиотеки jQuery является обеспечение взаимодействия программ на языке JavaScript с любыми элементами объектной модели документа HTML. С помощью библиотеки jQuery разработчик может получить доступ на чтение и запись к любым атрибутам элементов DOM (англ. Document Object Model – объектная модель документа) и их содержимому с помощью единого программного кода на языке JavaScript, позволяющего добиться одинакового результата в любых браузерах для ПК (Chrome 8+, Firefox 2+, Safari 3+, Opera 10.6+, Internet Explorer 6+) и мобильных устройств (iOS Safari 3.2+, Android Browser 2.1+, Opera Mobile 10+, Internet Explorer Mobile 10+).

Несмотря на то, что JavaScript является средством для решения многих задач [42, 45, 84, 85], до появления библиотеки jQuery разработчикам программ на языке JavaScript приходилось с большими трудозатратами учитывать различия диалектов JavaScript и вариантов представления объектной модели документа в разных браузерах.

Главной целью разработки библиотеки jQuery было собрать наиболее часто используемые программные элементы, чтобы в дальнейшем применять их без опасения за отличающийся уровень поддержки различными браузерами [46].

Библиотека jQuery позволяет отделить поведение объектов от структуры документа HTML. Вместо прямого обращения к объекту или обработчику события эти действия совершаются под управлением jQuery, которая преобразует обращения в специфический вид для конкретного браузера, и в полной мере реализовать концепцию «ненавязчивого JavaScript», заключающуюся в: отделении функциональности страницы от ее структуры, содержания и представления; обеспечении кроссбраузерности; возможности постепенного улучшения некоторого промежуточного компонента для обеспечения поддержи новых видов возникающих клиентских платформ без необходимости изменения программного кода, отвечающего за прикладной функционал АВП [55].

Для использования библиотеки jQuery в АВП файл с библиотекой подключается на каждой странице, где планируется использование программного кода с использованием jQuery.

Библиотека jQuery без модулей представляет собой отдельный файл JavaScript, который обычно сжимается специальным образом, чтобы уменьшить объем загружаемых данных. Модули для библиотеки jQuery также представляются в виде отдельных файлов, которые подключаются на страницах сайта после подключения основного файла библиотеки и тоже в предварительно сжатом виде.

Работа с библиотекой начинается с обращения к ней с помощью функции $. Для совместимости с другими библиотеками JavaScript, в которых $ может уже использоваться для других целей, при внедрении приложения на JavaScript с использованием jQuery в существующую систему более правильным является использование синонима $ – jQuery.

Способы использования jQuery делятся на два типа:

- обращение к объекту jQuery посредством передачи в функцию $() CSS-селектора для выборки элементов, удовлетворяющих заданному критерию с последующей работой с выбранными элементами с помощью различных методов объекта jQuery;
- вызов глобальных методов объекта jQuery (например, итераторов, обработчиков, массивов, событий и других).

Методы объекта jQuery возвращают либо значение, либо ссылку на объект, что позволяет формировать цепочки вызовов методов, реализуя концепцию текучего интерфейса.

В модуле вывода информационных сообщений основной функцией является функция отображения пользователю служебного сообщения и одновременного вывода его в консоль браузера.

Сами сообщения выбираются из единого массива служебных сообщений по коду, передаваемому вызывающим модулем. Это делается для обеспечения возможностей локализации на различные языки и для унификации сообщений, отображаемых различными модулями. В зависимости от типа сообщений, который устанавливается также по коду, доступны четыре вида оформления сообщений: информация, предупреждение, ошибка, нетиповое оформление (оформление сообщения получается из содержимого массива).

Контроллер резервирования вводимых пользователем данных

Контроллер резервирования вводимых пользователем данных предназначен для локального сохранения вводимых пользователем ИС в веб-формы данных, если в момент редактирования произошел сбой подключения клиента к серверу. При восстановлении соединения с сервером контроллер резервирования данных осуществляет отправку локально сохраненных данных на сервер.

Контроллер участвует в процедурах отправки на сервер с локальным сохранением вводимых данных и восстановления на сервер локально сохраненных данных.

Взаимодействует с модулями:

- модулем вывода информационных сообщений пользователю, отправляя пользователю сообщения;
- модулем проверки доступности сервера, запуская его при отсутствии подтверждения сервера о получении введенных пользователем данных и получая от него сигнал о возобновлении соединения с сервером для начала восстановления данных на сервер;
- контроллером обновления манифеста кэша для обновления изменившихся локально сохраненных элементов страниц в результате доставки на сервер введенных данных.

Отправляет введенные пользователем данные, включая ранее сохраненные, серверному асинхронному приемнику данных.

Модуль реализован на языке программирования JavaScript с использованием средств управления локального хранилища Local Storage стандарта HTML5 для создания локального клиентского кэша $C^{пост.дан}$ для введенных пользователем данных, AJAX для отправки данных на сервер, и функций библиотеки jQuery для обеспечения кроссплатформенности.

Local Storage – постоянное локальное хранилище объемом 2–10 Мбайт в зависимости от вида и версии браузера. Позволяет программе, являющейся частью страницы, локально сохранять данные в виде пар ключ-значение с помощью одного лишь браузера [58] и после ухода с сайта или закрытия окна браузера. Для каждого домена данные направляются в отдельное хранилище, которое доступно только для программ из этого домена. Данные в Local Storage никогда туда сами не записываются и не передаются с клиентского компьютера на веб-

сервер. Для использования Local Storage обязательно требуется клиентское веб-приложение на языке JavaScript, которое будет осуществлять все операции с данными в этом хранилище.

В отличие от других способов организации программно изменяемого постоянного локального хранилища, доступного для браузера, данная технология поддерживается всеми современными браузерами [35] для ПК (Chrome 4+, Firefox 3.5+, Safari 4+, Opera 10.5+, Internet Explorer 8+) и мобильных устройств (iOS Safari 3.2+, Android Browser 2.1+, Opera Mobile 11+, Internet Explorer Mobile 10+) без необходимости установки дополнительных плагинов. Перечисленные браузеры могут хранить в Local Storage 2–10 Мбайт текстовой информации. Учитывая, что Local Storage использует кодировку UTF-16 (то есть преимущественно по 2 Байта на букву), сохранить локально можно более одного миллиона знаков, введенных пользователем. Даже объема хранения в 2 Мбайта достаточно, поскольку в информационных системах отдельные неделимые документы и комментарии обычно не превышают указанного объема. Если объем введенных пользователем данных превышает миллион знаков, то, вероятнее всего, ввод производится из файла, уже сохраненного на его устройстве, а, значит, дополнительное резервирование данных этого файла не требуется.

Хранилище Local Storage представляет собой ассоциативный массив, в котором и ключи, и значения являются строками. Доступ к хранилищу из приложения на JavaScript осуществляется через обращение к объекту window.localStorage.

Хотя поддержка Local Storage является всеобщей для современных браузеров [3, 41, 78], на случай использования какого-нибудь старого браузера, не поддерживающего Local Storage, в АВП предусмотрена проверка на наличие хранилища с уведомлением пользователя, что у него АВП будет работать как традиционное веб-приложение, если такого хранилища нет.

Работа с хранилищем Local Storage осуществляется с помощью нескольких методов, которые позволяют добавлять пару ключ-значение к объекту хранилища, получать значение по ключу, производить удаление отдельной пары ключ-значение по ключу и проводить одновременное удаление всех пар ключ-значение (таблица 3.3).

Таблица 3.3. Методы локального хранилища Local Storage

Вызов метода	Описание метода
window.localStorage.setItem(key, value)	Сохраняет введенные пользователем данные в виде пары ключ-значение в объекте localStorage. В качестве ключа выступает хэш-код на основе адреса страницы и кода веб-формы, где были введены данные. Значением является массив, содержащий введенное сообщение, адрес страницы, идентификатор веб-формы, а также код версии для предотвращения коллизий совместного редактирования

window.localStorage.getItem(key)	Возвращает значение по указанному ключу, который использовался при сохранении значения. Применяется, если пользователь откроет в автономном режиме страницу, где он ранее редактировал и сохранял данные, чтобы АВП показало ему отредактированный локально сохраненный текст для продолжения редактирования
window.localStorage.key(n)	Возвращает значение для n-го ключа в ассоциативном массиве localStorage. Применяется при последовательной отправке на сервер нескольких сохраненных записей введенных пользователем данных
window.localStorage.removeItem(key)	Удаляет указанную по ключу пару ключ-значение из объекта localStorage после получения подтверждения от сервера о доставке данных
window.localStorage.clear()	Удаляет все пары ключ-значение из объекта localStorage. Дает возможность пользователю вручную очистить хранилище введенных данных, не затрагивая полученные с сервера сохраненные данные АВП

При добавлении пары ключ-значение к объекту localStorage можно использовать любой тип данных для определения значения, но храниться это значение будет как строка. В АВП сохраняемые данные представляют собой массив, состоящий из идентификаторов и значений полей веб-форм, а также служебной информации. Для хранения массивов используется объект JSON, позволяющий преобразовать данные массива или объекта в строку с помощью метода JSON.stringify. Обратное преобразование строки в массив или объект при получении данных из хранилища выполняется с помощью метода JSON.parse, что позволяет получить данные в первоначальном виде.

В случае необходимости использования устройств с публичным доступом предусмотрена возможность удаления всех введенных данных из локального хранилища АВП. Удаление осуществляется либо с помощью описанного метода clear(), либо с помощью встроенных средств браузеров по удалению личных данных.

При сохранении данных в локальном хранилище они защищены средствами операционной системы, разграничивающей доступ пользователей к персональным данным в браузере клиентского устройства. Таким образом, работая под своей отдельной учетной записью в операционной системе, пользователь ограничивает доступ к данным локального хранилища своего браузера других пользователей данного устройства.

Что касается доступа веб-приложений к данным в Local Storage, то данные доступны программам на JavaScript только того домена, программой с которого данные были туда записаны, то есть другие веб-приложения не могут получить доступ к чужим данным.

Для отправки на сервер введенных пользователем данных используется AJAX (англ. Asynchronous JavaScript and XML – асинхронный JavaScript и XML) – подход к организации интерактивного пользовательского интерфейса, основывающийся на асинхронном обмене данными между серверной и клиентской частями веб-приложения без перезагрузки страницы [50, 59]. Современными браузерами поддерживается асинхронная передача текстовых данных в любом формате, будь то простой текст, HTML, XML или JSON и др. Кроме того, полученный текст может быть интерпретирован как информационное содержимое страницы, таблица стилей или как дополнительный программный модуль.

В основе AJAX лежат два принципа: динамическое обращение к серверу с помощью XMLHttpRequest [25] без перезагрузки страницы; применение DHTML для динамического изменения содержания или представления страницы.

Библиотека jQuery обеспечивает унифицированный программный интерфейс для работы с AJAX в любых браузерах и, одновременно, позволяет полностью контролировать весь процесс передачи данных, то есть позволяет сделать веб-приложения с AJAX такими же надежными, как веб-приложения с перезагружаемыми полностью страницами.

В отличие от веб-приложений с полностью перезагружаемыми страницами веб-приложения, построенные с использованием технологии AJAX, позволяют обновлять в фоновом режиме отдельные части документа HTML без необходимости повторного получения с сервера остальных данных.

После первой загрузки веб-приложения с использованием AJAX на устройстве пользователя оказывается не только пользовательский интерфейс, но и программа-обработчик асинхронных вызовов. Этот обработчик получает от пользовательского интерфейса вызовы JavaScript, транслирует эти вызовы в HTTP/HTTPS-запросы к веб-серверу, а затем обрабатывает полученные от сервера данные и обновляет объекты пользовательского интерфейса без перезагрузки страницы.

Использование AJAX дает АВП следующие преимущества:

- уменьшение количества запросов на веб-сервер – высвобождает производительность клиента, ранее затрачиваемую на обслуживание этих запросов и отправки/получения данных, позволяет клиентскому устройству иметь больше свободных ресурсов для работающих программ на JavaScript;
- уменьшение сетевого трафика – из-за того, что загружаются лишь измененные части страниц или наборы данных для модификации страниц;
- снижение нагрузки на сервер – из-за уменьшения количества запросов от клиентов, иногда в разы; одновременно суммарное количество клиентов, которых может обслуживать сервер, пропорционально возрастает;

– повышение скорости реакции интерфейса – возможно приблизить скорость обновления интерфейса веб-приложения к скорости обновления интерфейса автономного приложения.

При использовании библиотеки jQuery работа с AJAX в веб-приложении строится вокруг метода $.ajax(url [, settings]) и вспомогательных методов (таблицы 3.4, 3.5) [54].

Таблица 3.4. Основные методы jQuery для работы с AJAX

Вызов метода	Описание метода
$.ajax(url [, settings])	Выполняет асинхронный HTTP-запрос (AJAX-запрос) по указанному адресу в соответствии с заданными или стандартными настройками отправки запроса при проверке доступности сервера и отправке введенных пользователем данных на сервер
$.ajaxSetup(settings)	Позволяет задать настройки для следующих за вызовом данного метода AJAX-запросов. Используется для указания различных обработчиков ошибок и ответов сервера, а также специфических параметров запросов АВП

В качестве обязательного аргумента url передается строка адреса на сервере, куда должен быть послан запрос, а необязательный аргумент settings представляет собой объект, состоящий в простейшем случае из пар ключ-значение, где каждая пара описывает различные особенности AJAX-запроса.

Стандартные значения некоторого произвольного подмножества элементов объекта settings могут быть заданы с помощью функции $.ajaxSetup(settings) – эти значения будут использоваться для всех AJAX-запросов веб-приложения, если иное не указано явно. Однако задание стандартных значений для всех AJAX-запросов не рекомендуется, так как в веб-приложении могут использоваться модули разных разработчиков, асинхронные запросы в которых могут ожидать других стандартных настроек и от их изменения перестанут работать.

Объект settings включает в себя следующие основные параметры: **beforeSend** – задает функцию для модификации отправляемого объекта перед отправкой; **cache** – задает настройки кэширования результатов запроса браузером; **complete** – задает функцию, вызываемую при любом завершении запроса, после success или error; **data** – содержит строку или объект с данными, которые должны быть отправлены на сервер; **error** – задает функцию, вызываемую в случае возникновения ошибки в ходе выполнения запроса; **success** – задает функцию, вызываемую в случае успешного завершения запроса; **timeout** – задает временной интервал, в течение которого приложение ожидает ответа от сервера после отправки запроса; **type** – задает тип метода для отправки запроса на сервер; **url** – задает адрес, куда должен быть послан запрос.

Для передачи информации о выполняющихся AJAX-запросах различным модулям клиентской части АВП применяются обработчики глобальных событий – событий, информация о которых доступна всем модулям АВП (таблица 3.5).

Таблица 3.5. Обработчики глобальных событий AJAX

Вызов обработчика	Описание обработчика
$(document).ajaxStart(handler())	Обработчик события ajaxStart, вызываемый при начале отправки первого из группы AJAX-запросов, инициирует сообщение пользователю о начале отправки на сервер группы записей ранее сохраненных данных
$(document).ajaxSend(handler(event, jqXHR, settings))	Обработчик события ajaxSend (его локальный аналог beforeSend), вызываемый перед отправкой AJAX-запроса, инициирует сообщение пользователю о начале отправки каждой отдельной записи ранее сохраненных данных
$(document).ajaxSuccess(handler(event, XMLHttpRequest, settings))	Обработчик события ajaxSuccess (его локальный аналог success), вызываемый при успешном завершении AJAX-запроса, инициирует сообщение пользователю об успешной доставке одной записи данных и удаление отправленной записи из Local Storage
$(document).ajaxError(handler(event, jqXHR, settings, thrownError))	Обработчик события ajaxError (его локальный аналог error), вызываемый в случае завершения AJAX-запроса с ошибкой, позволяет АВП понять, что соединение с сервером отсутствует, а данные не были отправлены
$(document).ajaxComplete(handler(event, XMLHttpRequest, settings))	Обработчик события ajaxComplete (его локальный аналог complete), вызываемый последним после завершения каждого AJAX-запроса, позволяет АВП понять, что запрос выполнен клиентской частью, а сообщения показаны пользователю
$(document).ajaxStop(handler())	Обработчик события ajaxStop, вызываемый после завершения всех AJAX-запросов, инициирует сообщение пользователю о результатах доставки всех записей данных

Задачей обработчиков глобальных событий AJAX является подключение пользовательских функций, заданных в handler(), к выполнению различных действий АВП, связанных с событиями AJAX.

Поскольку обработчики глобальных событий вызываются в контексте всего HTML-документа и для всех AJAX-запросов, то для идентификации и обработки конкретного запроса

в пользовательские функции передаются аргументы: **event** – идентификатор глобального события: ajaxStart, ajaxSend, ajaxSuccess, ajaxError, ajaxComplete, ajaxStop; **settings** – настройки AJAX-запроса, используемые в методе $.ajax(url [, settings]) при отправке запроса; **XMLHttpRequest** – объект, содержащий информацию о результате выполнения запроса, включая ответ сервера и различные служебные данные; **jqXHR** – объект jQuery XMLHttpRequest; **thrownError** – текстовое значение HTTP-статуса запроса, в котором произошла ошибка, например, «Не найдено», «Внутренняя ошибка сервера» и так далее.

Если в объекте settings указан параметр "global" : "false", то обработчики глобальных событий не будут запускаться.

Благодаря обработчикам глобальных событий, реализация AJAX с использованием библиотеки jQuery обеспечивает модули АВП информацией обо всех асинхронных запросах, отсылаемых к серверу, а также позволяет быстро добавлять при необходимости дополнительные модули, обрабатывающие информацию, получаемую в ходе выполнения AJAX-запросов. Это позволяет использовать в АВП данные, получаемые через асинхронные запросы, обеспечивать надежность доставки данных и интерактивную работу пользователей.

Помимо обработчиков глобальных событий, применяющихся для информирования модулей АВП о выполнении AJAX-запросов, в модулях, которые сами посылают такие запросы к серверу, обрабатываются локальные события, связанные с AJAX-запросами. В отличие от глобальных событий, информация о локальных событиях доступна только вызывающему их методу и появляется несколько раньше.

Отправка сохраненных данных пользовательского ввода осуществляется без перезагрузки открытой страницы. При этом используется сохраненный браузером сеанс авторизации пользователя. При отсутствии авторизации серверный асинхронный приемник данных потребует от пользователя сначала авторизоваться на сервере. Таким образом, внедрение АВП не снижает защищенности веб-приложения, модернизируемого до АВП.

Модуль проверки доступности сервера

Модуль проверки доступности сервера предназначен для фоновой проверки доступности сервера после сбоя соединения посредством отправки запросов к ответчику доступности сервера перед восстановлением локально сохраненных контроллером резервирования данных. Важен для предотвращения возможного кратковременного резкого снижения производительности (зависания) мобильных устройств при недоступности сервера в ходе отправки туда группы ранее сохраненных введенных сообщений или большого объема данных.

Модуль участвует в процедуре восстановления на сервер локально сохраненных данных.

Взаимодействует с модулями:

- контроллером резервирования вводимых пользователем данных, запуская его при восстановлении соединения с сервером;
- модулем вывода информационных сообщений пользователю для уведомления о необходимости авторизации.

Посылает запросы на сервер к ответчику доступности сервера.

Реализован на языке JavaScript с использованием AJAX для проверки доступности сервера, функций библиотеки jQuery для обеспечения кроссплатформенности.

Проверка доступности сервера осуществляется без перезагрузки открытой страницы и позволяет удостовериться, что соединение с сервером восстановлено, а пользователь все еще авторизован на сервере для отправки сохраненных данных. Если авторизация пользователя просрочена, то через модуль вывода информационных сообщений пользователю будет предложено повторно авторизоваться на сервере перед отправкой данных.

Контроллер обновления манифеста кэша

Контроллер обновления манифеста кэша предназначен для регистрации факта отправки на сервер введенных пользователем данных или авторизации пользователя и запуска процедуры обновления МК на сервере, чтобы позволить клиенту скачать обновленный МК и начать актуализацию локального кэша приложения.

Контроллер участвует в процедуре обновления данных в локальном хранилище Application Cache.

Взаимодействует с модулями:

- модулем вывода информационных сообщений пользователю для информирования пользователя о состоянии локального хранилища полученных с сервера данных Application Cache, наличии новой версии полученных с сервера данных и получения подтверждения пользователя на обновление отображаемой страницы;
- локально сохраненным МК для проверки загрузки перечисленных в нем файлов.

Отправляет на сервер запросы к актуализатору МК для запуска принудительной актуализации локального хранилища Application Cache клиента.

Модуль реализован на языке программирования JavaScript с использованием средств управления локального хранилища Application Cache стандарта HTML5 для создания локального клиентского кэша $C^{пост.прил}$ для полученных с сервера данных, AJAX для отправки фоновых запросов к актуализатору МК и функций библиотеки jQuery для обеспечения кроссплатформенности.

Для управления обновлением локального хранилища Application Cache из АВП используются методы update() и swapCache() объекта window.applicationCache (таблица 3.6).

Таблица 3.6. Методы локального хранилища Application Cache

Вызов метода	Описание метода
window.applicationCache.update()	Проверяет на сервере обновление МК. Инициирует обновление закэшированных данных в Application Cache, если МК изменился
window.applicationCache.swapCache()	Заменяет старую версию кэша на новую

Чтобы программа могла понять, когда нужно заменять старую версию кэша на новую, используется переменная window.applicationCache.status, принимающая текущие значения состояний хранилища (таблица 3.7).

Таблица 3.7. Состояния локального хранилища Application Cache

Состояние хранилища	Описание состояния
window.applicationCache.UNCACHED // UNCACHED == 0	Объект кэша не инициализирован
window.applicationCache.IDLE // IDLE == 1	Процесс обновления не проводится
window.applicationCache.CHECKING // CHECKING == 2	Проводится проверка наличия обновлений МК на сервере
window.applicationCache.DOWNLOADING // DOWNLOADING == 3	Осуществляется загрузка обновленных ресурсов с сервера в соответствии с обновленным МК
window.applicationCache.UPDATEREADY // UPDATEREADY == 4	Новая версия набора сохраненных данных загружена для обновления текущей версии кэша
window.applicationCache.OBSOLETE // OBSOLETE == 5	Текущая версия кэша устарела и должна быть удалена

Помимо работы с состояниями локального хранилища АВП обрабатывает программные события, которые инициирует браузер на разных этапах работы с локальным хранилищем Application Cache (таблица 3.8).

Таблица 3.8. События локального хранилища Application Cache

Программное событие	Описание события
cached	Инициируется после первого кэширования ресурсов из МК
checking	Инициируется при проверке обновления. Всегда первое событие в последовательности событий
downloading	Инициируется, если обновление найдено и браузер загружает ресурсы
error	Инициируется, если при загрузке ресурсов, перечисленных в МК,

	обнаружены ошибки (код 404 или 410), загрузка не удалась или МК успел измениться за время загрузки
noupdate	Инициируется, если обновление не требуется
obsolete	Инициируется, если при загрузке МК обнаружены ошибки (код 404 или 410). В этом случае содержимое локального хранилища Application Cache удаляется
progress	Инициируется при загрузке каждого ресурса, перечисленного в МК
updateready	Инициируется после того как ресурсы, перечисленные в МК, были заново загружены

Обновление данных в локальном хранилище Application Cache клиентской частью АВП осуществляется последовательным вызовом функций update(), которая вызывает проверку изменения МК и, при наличии изменения, загрузку обновленных ресурсов, перечисленных в измененном МК, и swapCache(), которая переключает браузер на использование обновленного кэша. Для отображения обновленных ресурсов пользователю требуется перезагрузка страницы, что выполняется программой после вызова swapCache() и получения подтверждения от пользователя, которое запрашивается для перезагрузки отображаемой страницы с измененными элементами, чтобы не потерять вводимые на этой странице данные, которые еще не были сохранены при перезагрузке отображаемой страницы.

Манифест кэша, загруженный с сервера

Загруженный с сервера манифест кэша предназначен для управления сохранением в локальном хранилище Application Cache при создании клиентского кэша $C^{пост.прил}$ полученных с сервера данных, включая программный код самого АВП, для обеспечения работоспособности АВП без подключения к серверу.

Участвует в процедуре обновления данных в локальном хранилище Application Cache.

Является текстовым конфигурационным файлом локального хранилища Application Cache стандарта HTML5. В соответствии с загруженным с сервера манифестом кэша контроллер обновления МК проверяет состояние хранилища данных Application Cache.

Кроме сохранения вводимых пользователем данных в случае потери соединения клиента с сервером в АВП используется другой вид постоянного хранилища, который позволяет его клиентской части АВП находиться на устройстве пользователя после первой загрузки из сети, запускаться и выполнять часть функций автономно.

При запуске АВП без подключения к серверу оно запускается из локальной копии, сохраненной при первоначальной загрузке с сервера.

Application Cache – постоянное локальное хранилище объемом от 5 Мбайт до неограниченного в зависимости от вида и версии браузера.

Технология Application Cache с недавнего времени поддерживается всеми современными браузерами для ПК (Chrome 4+, Firefox 3.5+, Safari 4+, Opera 10.6+, Internet Explorer 10+) и мобильных устройств (iOS Safari 3.2+, Android Browser 2.1+, Opera Mobile 11+, Internet Explorer Mobile 10+) без необходимости установки дополнительных плагинов [34].

Состав АВП регулируется МК, содержащим список всех необходимых веб-приложению файлов-элементов веб-старниц, которые должны быть закэшированы браузером [40, 76] и доступны без подключения к серверу.

МК подключается к каждой странице веб-приложения посредством указания атрибута manifest в теге <html>. Можно указывать как относительный, так и абсолютный путь к МК. Домен в абсолютном пути к МК должен совпадать с доменом размещения самого веб-приложения.

Адрес страницы, которая должна быть доступна в автономном режиме, не обязательно указывать в МК, достаточно, чтобы эта страница имела атрибут manifest тега html, указывающий на МК. Тогда браузер автоматически поместит данную страницу в хранилище Application Cache. Это позволяет управлять тем, какие страницы нужно кэшировать и избавляет от необходимости указывать адреса всех кэшируемых страниц в МК. Данный подход применяется для кэширования уже посещенных страниц, когда браузер узнает об их существовании и необходимости их кэширования только при переходе пользователя на эти страницы по ссылкам и при обнаружении там указания на МК.

Если АВП состоит из нескольких страниц, которые обязательно должны быть одновременно доступны для работы в автономном режиме, необходимо явно указать их относительные адреса в МК, иначе браузер не будет знать, что они нужны, и не будет загружать их сразу до перехода туда пользователя.

МК может находиться в любой папке веб-сервера, доступной через браузер, но существует ряд обязательных требований, которые необходимо выполнить:

- имя МК должно заканчиваться на “.manifest” (для более старых версий браузеров с поддержкой Application Cache. В новых браузерах МК может иметь любое расширение, оно лишь должно совпадать с указанием MIME-типа);
- МК должен отправляться клиенту с указанием MIME-типа “text/cache-manifest**”**.

Для этого при использовании веб-сервера Apache необходимо добавить в конфигурационный файл сервера httpd.conf или файл .htaccess в корневой папке веб-сервера АВП директиву AddType для указания MIME-типа, то есть того, какие файлы браузеру считать МК. Там же следует запретить браузерам кэшировать сами МК по их MIME-типу, чтобы пользователю не приходилось дожидаться обновления сеансового кэша браузера, если в ходе сеанса был обновлен манифест кэша Application Cache [27].

Аналогичные действия нужно осуществлять и для других видов ПО для веб-сервера в соответствии с их синтаксисом управления заголовком Content-Type.

Манифест кэша состоит из трех разделов:

CACHE – в этом разделе указываются пути к ресурсам (статическим файлам и динамическим источникам информации), которые должны быть закэшированы в Application Cache и доступны в автономном режиме;

NETWORK – в этом разделе указываются пути к ресурсам, которые требуют обязательного подключения к серверу, не должны кэшироваться и не доступны автономно;

FALLBACK – в этом разделе указываются пути к сохраненным файлам-заменителям, которыми должны быть заменены недоступные в автономном режиме ресурсы.

Строки, начинающиеся с символа решетка (#), являются комментариями, которые используются для указания версий перечисленных ресурсов, указания даты и времени последнего обновления МК или его контрольной суммы, так как для обновления закэшированных ресурсов нужно, чтобы изменился сам МК.

МК начинается со строки: CACHE MANIFEST. Со следующей строки перечисляются адреса ресурсов относительно корневой папки сайта (по одному в строке), которые необходимо закэшировать, после чего они станут доступными автономно.

Список кэшируемых ресурсов может находиться в любом месте МК, но, находясь не в начале файла, должен начинаться с заголовка CACHE. В одном МК может быть и несколько списков кэшируемых ресурсов под заголовками CACHE. Единственное требование – все ресурсы, перечисленные в списке для кэширования, должны существовать на сервере.

В разделах NETWORK и FALLBACK при указании пути используется символ подстановки звездочка (*), заменяющий произвольную часть адреса ресурса, позволяющий группировать ресурсы для применения к ним одинаковых действий.

Только звездочка (*) вместо всех записей в разделе NETWORK указывает браузеру, что все данные, отсутствующие в локальном хранилище Application Cache, должны загружаться по их адресам на сервере при наличии подключения к серверу. Без указания звездочки в разделе NETWORK веб-приложение не будет загружать внешние некэшированные файлы при наличии соединения. Это позволяет хранить страницы локально на клиентском устройстве, получая подключенные к ним файлы, не указанные в МК, при наличии подключения к сети.

При попытке загрузки некэшированных файлов без подключения к серверу возникает ошибка, однако предусмотрен способ предупредить пользователя об этом или предоставить ему автономную замену взаимодействия с сервером.

Информация о локальной замене недоступных в автономном режиме ресурсов размещается в разделе FALLBACK манифеста кэша. Для этого в строках раздела FALLBACK

указываются разделенные пробелом пары адресов на сервере: шаблон адреса некэшируемого ресурса и точный адрес его локальной замены, по одной паре на строку.

Это позволяет не нарушать автономную работу АВП при отсутствии каких-то ресурсов, которые не могут быть закэшированы.

Как было отмечено ранее, все статические файлы, подключаемые на страницы, а именно: таблицы стилей CSS, программный код на языке JavaScript, изображения и другие должны быть явно описаны в МК для того, чтобы быть закэшированными. Пример файла МК АВП приведен в таблице 3.9.

Таблица 3.9. Содержание файла манифеста кэша АВП

Строки МК	Значение
CACHE MANIFEST	Первая строка МК
# 2015-01-01 13:37:11	Дата и время начала актуальности МК
CACHE	Раздел перечисления сохраняемых ресурсов АВП
/scripts/jquery.js	Сохранение библиотеки jQuery
/scripts/awa.js	Сохранение набора модулей клиентской части АВП
/styles/awa.css	Сохранение таблицы стилей служебных сообщений АВП
/images/online.png /images/process.png /images/offline.png	Сохранение изображений для служебных сообщений АВП: сервер доступен, отправка данных, сервер недоступен
/scripts/menu.js /styles/style.css /images/logo.jpg /images/button.png /index.html …	Сохранение элементов страниц преобразуемого в АВП традиционного веб-приложения, которые должны быть доступны автономно
NETWORK	Раздел перечисления несохраняемых ресурсов, требующих подключения к сети Интернет
*	Указание, что все несохраненные элементы требуют подключения к сети Интернет
FALLBACK	Раздел указания автономной замены для несохраненных элементов
/ /offline.html	Указание выводить на замену всех несохраненных страниц традиционного веб-приложения сохраненную страницу /offline.html (в CACHE не указывается)

В автономном режиме пользователю будут доступны все посещенные им страницы веб-сайта и подключенные к ним статические файлы, перечисленные в МК.

Разные разделы веб-сайта могут иметь разные МК, что позволяет создавать для одного веб-сайта несколько независимых локальных хранилищ данных, то есть, другими словами, позволяет размещать несколько АВП на одном веб-сайте.

Также для разных авторизованных пользователей могут использоваться разные МК, позволяя подавать на устройство авторизованного пользователя АВП, специфичные для данного конкретного пользователя.

Если МК или какой-то из указанных ресурсов не удается загрузить с сервера, то продолжает использоваться старая версия кэша.

АВП будет сохраняться в кэше постоянно, пока не произойдет одно из событий:

- удаление пользователем данных хранилища Application Cache в браузере (происходит аналогично удалению данных Local Storage);
- удаление на сервере администратором МК и его упоминания со страниц, что влечет за собой удаление и всех закэшированных файлов АВП на стороне клиента.

Аналогично Local Storage данные в Application Cache защищены средствами разграничения доступа пользователей ОС к сохраненным данным в браузерах.

Использование МК делает возможным кэширование страниц, загружаемых через SSL, что обеспечивает доступ в автономном режиме к страницам, загруженным по протоколу HTTPS. Для конечного пользователя это означает возможность безопасно загружать АВП на свои устройства, не боясь вторжения на этапе передачи данных, используя стандартные способы защиты соединений, предоставляемые веб-серверами – надежные и не требующие дополнительных действий от пользователей.

Технология Application Cache в современных браузерах в сочетании с возможностями HTML5 и CSS по созданию различных интерфейсов и включения произвольных объектов в текст страниц, а также использованием языка JavaScript и кроссбраузерных библиотек типа jQuery, позволяет создавать кроссплатформенные бездистрибутивные (не имеющие файла установщика и процедуры установки) АВП.

Таким образом, с принятием стандарта HTML5 и появлением во всех браузерах локальных хранилищ Local Storage и Application Cache становится возможным создание кроссплатформенных АВП с управляемым локальным кэшем полученных с сервера и введенных пользователем данных.

Это расширяет круг поддерживаемых терминальных устройств ГИС с использованием данных ДЗЗ за счет всех существующих и будущих устройств с поддержкой HTML5, позволяет обеспечить автоматическое резервирование и восстановление на сервер введенных

пользователем данных, а также возможность автономной работы с полученными с сервера данными.

3.2.3 Модули на стороне сервера

Модули на стороне сервера могут быть реализованы на любом языке программирования в зависимости от требований к используемым средствам программирования и специфики конкретного традиционного веб-приложения, модернизируемого до АВП. В данной работе модули реализованы на языке PHP, поскольку модернизируемая система WackoWiki написана на языке PHP.

Модули на стороне сервера структурно представлены:

- асинхронным приемником данных;
- ответчиком доступности сервера;
- актуализатором манифеста кэша;
- манифестом кэша.

Асинхронный приемник данных

Асинхронный приемник данных предназначен для обработки AJAX-запросов с введенными пользователем данными. Серверная специфика обработки AJAX-запросов заключается в том, что в ответ на такой запрос посылается не страница подтверждения, а небольшой текстовый файл, содержащий ответ сервера.

Асинхронный приемник данных участвует в процедурах отправки на сервер с локальным сохранением вводимых данных и восстановления на сервер локально сохраненных данных.

Получает данные от клиентского контроллера резервирования вводимых пользователем данных.

Взаимодействует с актуализатором МК, вызывая через него обновление даты и времени актуальности МК, если получение сервером введенных пользователем данных требует изменения сохраненных данных в хранилище Application Cache на стороне клиента.

Интегрируется со стандартным публикатором введенных пользователем данных традиционного веб-приложения. Позволяет получать данные, отправленные клиентом в фоновом режиме с помощью AJAX. Для этого данный модуль встраивается в традиционное веб-приложение после вызова функций сохранения текстовых данных в БД и до отправки страницы подтверждения. При получении данных модуль проверяет, отправлены ли они обычным способом или с помощью AJAX, и если они отправлены с помощью AJAX, то вместо страницы подтверждения отправляется специальный ответ для AJAX-запроса.

В результате данной модификации публикатор введенных данных традиционного веб-приложения получает возможность обрабатывать AJAX-запросы от клиентской части АВП.

Ответчик доступности сервера

Ответчик доступности сервера предназначен для приема AJAX-запроса проверки доступности сервера от клиентской части АВП и отправки в ответ текстового файла подтверждения, содержащего состояние авторизации пользователя АВП на сервере.

Ответчик доступности сервера участвует в процедуре восстановления на сервер локально сохраненных данных.

Получает запросы о доступности сервера от клиентского модуля проверки доступности.

Интегрируется со средствами авторизации пользователя традиционного веб-приложения, чтобы по данным, включаемым браузером в AJAX-запрос, определить, является ли пользователь авторизованным на сервере. При получении запроса модуль вызывает проверку авторизации пользователя традиционного веб-приложения на основе автоматически включаемых браузером в тело запроса данных. Если пользователь авторизован, отправляется ответ о том, что можно отправлять данные на сервер, в противном случае – что требуется авторизация.

Ответ о доступности сервера не кэшируется ни в локальном хранилище приложений Application Cache, ни в сеансовом кэше браузера. Его размер достаточно мал, поэтому он загружается с минимальным расходом сетевого трафика не более 1 Кбайт.

Актуализатор манифеста кэша

Актуализатор манифеста кэша предназначен для обновления даты и времени актуальности МК, чтобы вызвать проверку обновления данных в локальном хранилище Application Cache на стороне клиента.

Актуализатор МК участвует в процедуре обновления данных в локальном хранилище Application Cache на стороне клиента.

Взаимодействует с асинхронным приемником данных, вызывается им для обновления даты и времени актуальности МК, если получение сервером введенных пользователем данных требует изменения сохраненных данных в хранилище Application Cache на стороне клиента.

Получает запросы от клиентского контроллера обновления МК на обновление даты и времени актуальности МК для принудительной проверки обновления сохраненных данных в хранилище Application Cache на стороне клиента по команде пользователя. Актуализатор МК может быть запущен любыми программными компонентами на стороне сервера в случае, если

изменения данных возникли не в результате ввода их пользователем, а в результате, например, синхронизации нескольких БД.

При запуске актуализатор МК устанавливает дату и время текущего изменения в МК. После этого МК считается измененным всеми клиентами, и их локальный кэш АВП будет перепроверен на наличие обновлений на сервере. Таким образом, клиенты поддерживают у себя в кэше в актуальном состоянии всю обновляющуюся информацию в системе.

Манифест кэша

Назначение манифеста кэша было описано в разделе 1.3, а его структура и правила функционирования на стороне клиента в разделе 3.2.2.5. Как уже было сказано, МК является статическим файлом конфигурации, который управляет процессом сохранения получаемых от веб-сервера данных АВП в постоянном хранилище Application Cache клиента.

Файл МК участвует в процедуре обновления данных в хранилище Application Cache.

Взаимодействует с актуализатором МК на стороне сервера, который изменяет дату и время начала аткуальности МК для того, чтобы дать команду клиентам обновить кэш сохраненного приложения при появлении обновлений в системе.

В АВП МК одновременно существует в двух экземплярах: на стороне сервера и на стороне клиента. Серверный МК загружается клиентом при обращении к любой странице АВП и сравнивается с сохраненной версией. Если версии идентичны, то изменений нет. Если МК отличаются, то запускается процедура обновления данных в локальном хранилище Application Cache, в ходе которой загруженный с сервера МК заменяет предыдущий, сохраненный на стороне клиента.

3.2.4 Модернизация традиционного веб-приложения до АВП

Из описания серверных модулей АВП следует, что для модернизации традиционного веб-приложения до АВП требуется минимальная доработка традиционного веб-приложения.

Разработанный набор программных модулей интегрируются и взаимодействует с серверной частью и веб-интерфейсом традиционного веб-приложения.

Для преобразования существующего традиционного веб-приложения в АВП разработана процедура установки, включающая в себя следующие действия:

1. на сервере размещаются файлы библиотеки клиентках компонентов АВП, библиотеки jQuery, набора серверных компонентов так, чтобы обращение к ним было доступно клиентам;

2. на сервере создается МК, который содержит в разделе CACHE указание на библиотеку клиентских компонентов АВП и библиотеку jQuery, в разделе NETWORK – звездочку,

комментарий с датой и временем актуализации МК, а также указание других элементов в зависимости от конкретных потребностей;

3. асинхронный приемник данных интегрируется со средствами сохранения данных традиционного веб-приложения для передачи полученных введенных данных серверной части традиционного веб-приложения и отправки клиенту ответов на AJAX-запросы;

4. ответчик доступности сервера интегрируется со средствами авторизации пользователя;

5. актуализатор МК интегрируется со средствами отслеживания изменений серверной части традиционного веб-приложения;

6. в шаблон страниц традиционного веб-приложения добавляются вызовы: МК, библиотеки клиентских компонентов АВП и библиотеки jQuery.

После того как эти действия выполнены, традиционное веб-приложение преобразовано в АВП и начинает обладать всеми его преимуществами.

3.3 Интеграция модулей АВП с платформой WackoWiki

Даже полностью статический веб-сайт пригоден для модернизации до АВП. Для демонстрации модернизации традиционного веб-приложения до АВП выбрана платформа традиционного веб-приложения WackoWiki, которая позволяет просматривать информацию, получаемую с сервера и отправлять на сервер вводимые пользователем данные. Данная платформа: реализует большинство операций пользователей веб-приложений с данными [39]; является системой с открытым исходным кодом; хорошо структурирована и имеет модульную архитектуру; отличается достаточной простотой, чтобы наиболее наглядно продемонстрировать применение технологии преобразования произвольной ИС с веб-интерфейсом в АВП и работу его компонентов. Рассматривается взаимодействие пользователя с системой, которая может содержать на своих страницах различные типы данных, поэтому данный пример справедлив и для системы с преобладанием определенного типа данных, а именно, для ГИС с использованием данных ДЗЗ.

Система WackoWiki обеспечивает авторизацию пользователей и гибкое разграничение их прав доступа. Данные, передаваемые по HTTP по публичным каналам связи, дополнительно могут быть защищены шифрованием с использованием протоколов SSL или TLS [33]. В результате передача данных между клиентом и сервером происходит по защищенному протоколу HTTPS. Это реализуется стандартными средствами любого веб-сервера, поддерживается всеми современными браузерами и позволяет обеспечить надежность передачи данных, как если бы она происходила в локальной сети.

Система контроля версий страниц WackoWiki ограждает пользователей от возможных ошибок форматирования, случайного редактирования и позволяет сравнивать внесенные правки.

Система WackoWiki сохраняет все правки своих документов и комментариев к ним в виде отдельных версий в хронологическом порядке. Это позволяет сравнивать версии за любой период, обрабатывать случаи коллизий совместного редактирования документов и объединять внесенные изменения в окончательной версии документа. История всех версий хранится постоянно и может быть просмотрена пользователем в любой момент.

Для отслеживания и предотвращения коллизий совместного редактирования системой осуществляется контроль преемственности правок. Если данные, пришедшие из формы редактирования, содержат дату предыдущей правки, то они признаются системой следующей версией документа. Дата этой новой принятой версии становится датой предыдущей правки для данных, которые придут позднее. В случае одновременного редактирования документа двумя или более пользователями данные, пришедшие первыми, станут следующей версией документа, а пользователю, пославшему свои данные позже, будет предложено сравнить свою версию документа с появившейся (или появившимися) за время между его просроченной датой предыдущей версии и датой имеющейся версии. При этом все необходимые средства для сравнения версий документов предоставляет система WackoWiki.

WackoWiki представляет собой традиционное веб-приложение, клиентская часть которого отвечает за ввод и отображение данных, а серверная часть осуществляет хранение, обработку введенных данных и управление доступом пользователей к информации и функционалу. Программный код организован по модульному принципу. Состоит из ядра, обеспечивающего подключение модулей и предоставляющего им необходимое окружение, и модулей трёх основных типов: обработчиков сценариев взаимодействия с системой; встраиваемых в текст страницы сервисов, действий; преобразователей текста, форматеров.

Обработчики сценариев осуществляют обработку страниц системы как единиц хранения данных. Это создает дополнительный уровень абстракции БД и позволяет использовать WackoWiki для хранения данных любой структуры без ограничений.

Действия, вызываемые в тексте страницы, активируют сервисы, совершающие в системе какие-либо операции и выводящие результат своей работы на вызывающей странице. Этот механизм позволяет автоматически формировать оглавление документа, список страниц кластера или загруженных файлов и многое другое.

Форматеры реализуют автоматическую разметку структурированных текстов, что позволяет наглядно визуализировать различного рода листинги, журналы, отчеты, программные коды и любые другие тексты, в которых можно выявить единую структуру.

Изменения в архитектуре системы WackoWiki после интеграции с разработанными модулями АВП показаны на рис. 3.6. Выделены те элементы, которые были добавлены или изменены. Это демонстрирует, как разработанные компоненты могут встраиваться в архитектуру существующей ИС.

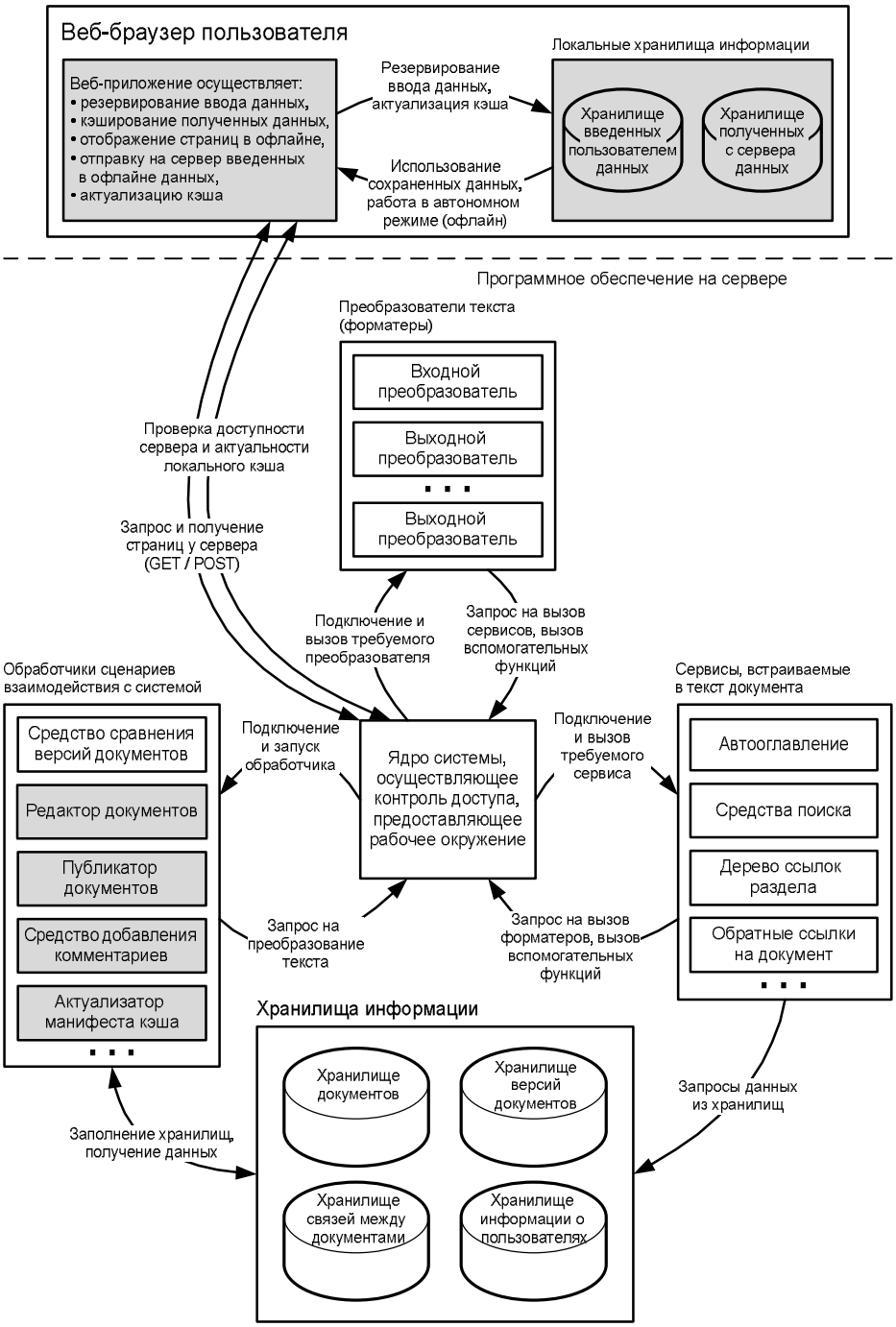


Рис. 3.6. Архитектура программного комплекса WackoWiki с разработанным АВП

Благодаря вынесению большой части функционала разработанного АВП на клиентский уровень и реализации серверной части АВП в виде набора модулей, предоставляющих интерфейс для работы с данными в БД и управляющих их кэшированием на стороне клиента, в

данной разработке удалось показать, что произвольная модульная ИС с веб-интерфейсом может быть преобразована в АВП.

3.4 Тестирование

Согласно изложенному выше, разработанное программное обеспечение АВП представляет собой совокупность программных модулей, функционирующих в среде WackoWiki с использованием механизмов стандарта HTML5 и технологий современных браузеров. Язык реализации модулей клиентской части – JavaScript, модулей серверной части – PHP. Данное программное обеспечение было протестировано на связке тестового веб-сервера с различными клиентскими устройствами на базе Windows, Linux, iOS и Android с установленными браузерами Internet Explorer, Opera, Firefox, Chrome, Safari и Android Browser последних версий, а также устаревшими браузерами без поддержки HTML5.

Методика тестирования работоспособности полученного АВП включает в себя открытие страниц и ввод пользователем данных на разных клиентских платформах.

В устаревших браузерах, не поддерживающих HTML5, было проверено, что АВП ведет себя как традиционное веб-приложение и не ограничивает исходную функциональность.

В современных браузерах с поддержкой HTML5 помимо этого проверялись:

1. загрузка АВП в локальное хранилище с последующим отключением соединения с сервером и доступом к локально сохраненным данным;

2. обновление АВП локально сохраненных данных при изменении этих данных на сервере с последующим отключением соединения с сервером и доступом к локально сохраненным данным;

3. резервирование вводимых пользователем данных при отключенном соединении с сервером, их автоматическая отправка на сервер при возобновлении соединения; повторно п.2, если ввод данных требовал обновления локально сохраненных данных.

Предложенная методика тестирования позволяет обеспечить безошибочный переход от традиционного веб-приложения к АВП.

Протестированные сценарии работы созданной системы приведены в таблице 3.10.

Таблица 3.10. Протестированные сценарии работы разработанной системы

Страница	Условия тестирования	Действия пользователя	Ожидаемый результат
Любая страница	Браузер не поддерживает локальные хранилища HTML5 и/или JavaScript	Открытие, просмотр страницы или ввод данных	Отображение страниц и ввод данных возможны, как при работе с традиционным веб-приложением
Далее во всех случаях браузер поддерживает локальные хранилища HTML5 и JavaScript			
Любая страница	Есть связь с сервером	Первое открытие, просмотр страницы	1. Отображение страницы пользователю. 2. Сохранение страницы и ресурсов, перечисленных в связанном МК, в локальное хранилище данных веб-приложения (ресурсы сохраняются однократно)
Любая страница	1. Есть связь с сервером / нет связи с сервером. 2. МК на сервере не отличается от сохраненного локально	Открытие, просмотр посещенной ранее или указанной в МК страницы	Отображение страницы пользователю из локального хранилища данных веб-приложения
Любая страница	1. Есть связь с сервером. 2. МК на сервере отличается от сохраненного локально	Открытие, просмотр посещенной ранее или указанной в МК страницы	1. Обновление МК и локального хранилища данных веб-приложения или удаление всех данных из локального хранилища данных веб-приложения, если новый МК пуст. 2. Отображение страницы пользователю из локального хранилища, если эта страница не была изменена, и с сервера, если страница была изменена или локальное хранилище было очищено
Страница редактирования документа системы или добавления комментария	Есть связь с сервером	Ввод данных в форму и нажатие кнопки сохранения	1. Сохранение введенных данных на сервер. 2. Обновление МК и локального хранилища данных веб-приложения
Страница редактирования документа системы или добавления комментария	Нет связи с сервером	Ввод данных в форму и нажатие кнопки сохранения	1. Сохранение введенных данных в локальном хранилище данных пользовательского ввода. 2. Вывод пользователю информационного сообщения о недоступности сервера, локальном сохранении данных и дальнейших возможных действиях пользователя

Любая страница	1. Есть связь с сервером. 2. В локальном хранилище данных пользовательского ввода есть отредактированные документы и/или комментарии	Открытие, просмотр страницы	1. Отправка локально сохраненных данных пользовательского ввода на сервер. 2. Удаление отправленных на сервер записей из хранилища данных пользовательского ввода. 3. Обновление МК и локального хранилища данных веб-приложения. 4. Вывод пользователю информационного сообщения об отправке данных
Любая страница	1. Есть связь с сервером. 2. В локальном хранилище введенных данных есть отредактированные документы и/или комментарии. 3. Некоторые записи в локальном хранилище данных пользовательского ввода имеют более раннюю версию, чем документ на сервере (то есть могли устареть)	Открытие, просмотр страницы	1. Отправка локально сохраненных данных на сервер. 2. Удаление отправленных на сервер записей из хранилища данных пользовательского ввода. 3. Обновление МК и локального хранилища данных веб-приложения. 4. Вывод пользователю информационного сообщения об отправке данных на сервер со ссылкой на сравнение версий документа, в котором возникла неупорядоченность правок

Результаты тестирования показали, что использование устаревших версий браузеров, не поддерживающих локальные хранилища данных HTML5 или JavaScript, не препятствует использованию системы в целом, а только ограничивает использование новых функций резервирования данных и автономной работы. Это означает, что внедрение разработанной системы поверх существующих веб-приложений не повышает минимальных системных требований к клиентским устройствам.

Что касается современных браузеров с поддержкой HTML5, то результаты тестирования показали достижение поставленных перед АВП целей, а именно: кроссплатформенности (использовались клиентские устройства на базе Windows, Linux, iOS и Android с установленными браузерами Internet Explorer, Opera, Firefox, Chrome, Safari и Android Browser) , отказоустойчивости и возможности автономной работы с сохраненной на клиентском устройстве информацией и её синхронизации с данными на сервере (таблица 3.10).

3.5 Решение практических задач с помощью АВП

Рассмотрим примеры, подтверждающие заявленные возможности АВП, применительно к решению двух классов задач:

а) мониторинга земной поверхности в интересах лесного хозяйства. Данный класс задач характеризуется требованиями к пространственному разрешению данных ДЗЗ от 1 до 10 м и необходимым периодом обновления данных не более 1 суток;

б) проведения геодезических работ. Данный класс задач характеризуется требованиями к пространственному разрешению данных ДЗЗ от 0,5 до 10 м и необходимым периодом обновления данных от 1 до 6 месяцев.

3.5.1 Использование АВП при работе с данными ДЗЗ в задачах мониторинга лесного хозяйства

Рассмотрим пример работы АВП с данными ДЗЗ при решении задач мониторинга вырубки леса, включающего в себя отслеживание выполнения плана вырубки и мониторинг незаконных рубок в рамках выделенного участка.

Рассматривается участок леса площадью 9000 га, арендованный для вырубки сроком на 5 лет, то есть на участке запланирована ежедневная вырубка 4,932 га леса без выходных. Помимо запланированной разрешенной рубки на выделенном участке вероятно появление участков незаконной рубки, площадь которых по статистике нарушений за 2009 год может достигать в среднем 3,34% от разрешенного объема [29]. Таким образом, для данного примера площадь незаконной рубки может составлять 300,6 га, то есть примерно 0,165 га леса в день.

Для мониторинга используются многослойная интерактивная карта местности площадью 9000 га, сочетающая в себе растровые данные ДЗЗ с пространственным разрешением 1–10 м [12, 29] (уровни детализации с 14 по 18) и векторные слои, на которых нанесены схемы лесных кварталов и выделов, план разрешенных рубок, схема временных дорог и коммуникаций, а также блоки текстово-графических описаний для точек местности и элементов слоев (200 объектов на векторных слоях).

Ежедневно поступают новые данные ДЗЗ, учитывающие изменения на площади разрешенных и незаконных вырубок общей площадью 5,097 га в сутки (что составляет 0,057% общей площади карты). Благодаря серверной обработке полученных данных ДЗЗ на карте у конечного пользователя изменения затрагивают только участки, на которых произошла вырубка за прошедшие сутки. На векторном слое также могут быть нанесены метки и обведены участки незаконной вырубки, чтобы привлечь внимание пользователя даже к малозаметным участкам, поскольку на этапе серверной обработки может проводиться более точное сравнение новых и предыдущих снимков, чем это может сделать человек. С вводом текстовых описаний

пользователем и получением новых пометок от сервера векторные слои и текстовые обозначения могут меняться в течение каждого рабочего дня (будем считать, что на 10% ежедневно). Применение АВП позволяет реализовать следующий сценарий работы специалиста по мониторингу использования разрабатываемого участка (рис. 3.7, таблица 3.11).

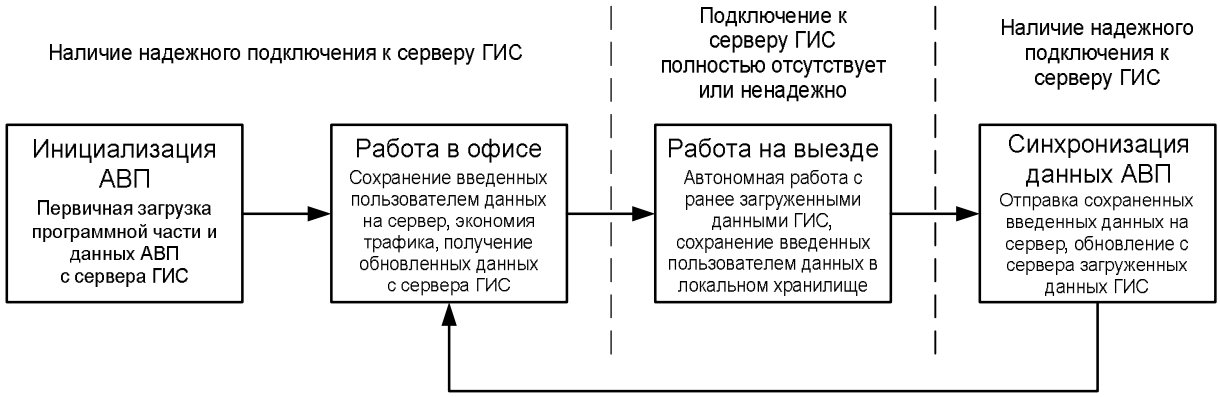


Рис. 3.7. Процесс работы пользователя в ГИС с АВП

Таблица 3.11. Процесс работы пользователя в ГИС с АВП в сравнении с традиционным веб-приложением

№	Действия пользователя	АВП	Традиционное веб-приложение
1.	Первый переход пользователя по адресу веб-сервера ГИС с произвольного клиентского устройства со стандартным веб-браузером.	Инициализация: первичная загрузка программной части и данных АВП с сервера ГИС в локальное хранилище клиентского устройства. Рекомендуется использовать надежное подключение к Интернет с высокой пропускной способностью, чтобы инициализация прошла быстрее. Во время инициализации АВП работает так же как и традиционное веб-приложение; можно сразу приступать к работе.	Загрузка главной страницы и отображаемых на ней данных.
2.	Работа при наличии подключения к Интернет. Просмотр интерактивной карты, ввод и редактирование описаний элементов карты. Например, в офисе.	Проверка обновлений и загрузка обновленных данных с сервера ГИС, отправка введенных пользователем данных на сервер. Экономия трафика и защита от сбоев соединения благодаря использованию данных из локального хранилища и резервированию вводимых пользователем данных.	Загрузка данных с сервера по мере обращения к ним пользователя, отправка введенных пользователем данных на сервер. Сбои соединения с сервером вызывают неработоспособность
3.	Работа при отсутствии подключения к Интернет. Просмотр интерактивной карты, ввод и редактирование описаний элементов карты. Например, на выезде.	Автономная работа с ранее загруженными данными ГИС, сохранение введенных пользователем данных в локальном хранилище.	Не работает

4.	Работа при возобновлении подключения к Интернет.	Синхронизация данных: отправка сохраненных введенных пользователем данных на сервер, проверка обновления и загрузка с сервера изменившихся данных ГИС.	Повторение загрузки страницы, при отображении которой соединение исчезло, и соответствующих ее адресу данных ГИС.
5.	Переход к шагу 2.		

Перед выездом на проверку (на определенный участок вырубки) специалист из офиса или любого другого места с надежным подключением к серверу ГИС с использованием персонального устройства заходит через Интернет на сервер ГИС, содержащий векторную карту этого участка территории с растровой подложкой из данных ДЗЗ, полученных со спутника. Рассматриваемый веб-ресурс представляет собой АВП, поэтому позволяет сохранить локально необходимую часть данных на его клиентском устройстве. Это осуществляется средствами одного лишь веб-приложения и происходит в браузере без необходимости установки какого-либо дополнительного программного обеспечения на клиентское устройство, что позволяет использовать различные типы клиентских устройств.

После авторизации пользователя на его устройство автоматически загружается и сохраняется индивидуально сконфигурированная клиентская часть АВП с необходимыми этому пользователю в автономном режиме данными и программными средствами для работы с ними.

После первичной загрузки данных АВП пользователь может полноценно работать с сохраненными данными в условиях полного отсутствия или плохого соединения с сервером, а также вводить свои пометки (например, для плана вырубок: текстовые комментарии, обновления атрибутов объектов, изменения границ объектов).

Специалист находит на вверенном участке объекты, чьи границы на векторном слое не совпадают с актуальными данными ДЗЗ, отмечает их в АВП, например, выделяя цветом или задавая новые границы, указывая недорубы, участки неразрешенной рубки, не вывезенный лес, свалки и другое. Если изменения на участке по данным ДЗЗ соответствуют плану на векторном слое, то в выезде на участок нет необходимости. Когда специалисту приходится выезжать на участок рубки для уточнения деталей: визуального подтверждения наличия объектов, их очертаний, актуальности атрибутов, GPS-координат точек объектов и другого, используемое на выезде персональное устройство (ноутбук, планшет), несмотря на полное или частичное отсутствие соединения с сервером, при работе с АВП содержит все необходимое для работы.

Осматривая объект, специалист делает пометки на электронном плане и вносит обновления границ и атрибутов объектов в свое устройство, после чего возвращается в зону надежного подключения к серверу через сеть Интернет (например, в офис) для синхронизации с сервером. Все введенные пользователем в автономном режиме данные сохранены АВП и будут

отправлены на сервер. Все сохраненные данные, которые были изменены на сервере за время автономной работы клиента, будут автоматически обновлены на клиентском устройстве. Обновление данных происходит у всех пользователей. Все получают актуальную информацию.

Синхронизация клиентской части АВП каждого сотрудника с сервером ГИС осуществляется автоматически при наличии подключения к серверу через сеть Интернет. Элементы всех слоев кэшируются для экономии трафика и обеспечения возможности автономной работы. При этом период актуальности составляет не менее одного рабочего дня, поэтому специалист может работать полностью автономно кроме ежедневной синхронизации.

Во всех примерах: программный код АВП для отображения слоев и обеспечения взаимодействия с ними пользователей не меняется при изменении отображаемых данных, поэтому может быть долговременно закэширован до изменения программной части АВП; АВП функционирует с полным набором сохраненных необходимых данных и программного кода, и возможностью автономной работы с резервированием вводимых пользователем данных (режимы 1 и 4); для сравнения с традиционным веб-приложением (помимо отсутствия у него возможностей автономной работы и резервирования вводимых пользователем данных) принято, что в традиционном веб-приложении ежедневно загружается около 20% от общего объема растровых и векторных данных карты участка (то есть пользователь работает с различными районами того же участка) и весь программный код традиционного веб-приложения.

Именно здесь проявляются преимущества АВП, обусловленные сохранением на устройстве пользователя необходимых данных на период их актуальности (таблица 3.12). Эти преимущества действуют для различных каналов связи.

Таблица 3.12. Преимущества АВП перед традиционным веб-приложением в конкретных примерах использования

Пример, №	Площадь растровой подложки, га (км^2)	Количество элементов векторных слоев, ед.	Пропускная способность канала передачи данных, Кбит/с	Экономия времени ожидания загрузки данных одного сеанса АВП по сравнению с традиционным веб-приложением, %	Экономия трафика АВП по сравнению с традиционным веб-приложением, %	
					За первую неделю	За первый месяц
1.	9000 (90)	200	64	**92,28**	**31,84**	**79,40**
			1024	**92,54**		

В примере №1 используемое программное решение АВП содержит:

- растровый слой данных ДЗЗ уровней детализации с 12 по 18 (для обеспечения необходимого пространственного разрешения и удобства навигации), состоящий из 5334 тайлов общим объемом 106680 Кбайт;

- векторные слои, содержащие 200 объектов общим объемом 16384 Кбайт;
- программный код и элементы оформления АВП, включающие в себя средства для отображения данных и взаимодействия с ними пользователя, всего 30 элементов общим объемом 5120 Кбайт;
- манифест кэша (МК), управляющий локальным кэшированием полученных данных на стороне клиента, объемом 280 Кбайт.

При первичной загрузке (инициализации) АВП на устройство пользователя загрузится 128464 Кбайт = 125,45 Мбайт = 0,12 Гбайт (что эквивалентно 1/133 доли объема собственной памяти устройства Apple iPad с 16 Гбайт памяти).

Примерно 1699,21 Кбайт обновляются каждый день.

Будем считать, что за рассматриваемый период программный код и элементы оформления АВП не обновляются. Сохраненные элементы подаются из локального хранилища на стороне клиента и отображаются пользователю практически мгновенно. При каждом обращении к АВП с сервера загружается МК, который позволяет клиентской части АВП проверить актуальность локально сохраненных данных и управляет обновлением данных в локальном хранилище, если это необходимо.

Таким образом, суммарный трафик при работе с АВП составит: за неделю (7 дней) – 140339,25 Кбайт, за месяц (31 день) – 187840,23 Кбайт.

Традиционным веб-приложением ежедневно загружается 20% от общего объема растровых и векторных данных карты участка (21336 + 3276,8 Кбайт) и программный код, который состоит из 25 элементов общим объемом 4800 Кбайт, МК – нет. Всего ежедневно загружается примерно 29412,8 Кбайт.

В результате суммарный трафик при работе с традиционным веб-приложением составит: за неделю (7 дней) – 205890 Кбайт, за месяц (31 день) – 911797 Кбайт.

По данным примера построены графики на рис. 3.8, 3.9, 3.10, 3.11, которые иллюстрируют преимущества АВП при работе с данными ДЗЗ в геоинформационной системе с веб-интерфейсом перед традиционным веб-приложением.

Из рис. 3.8, 3.9 следует, что начиная уже с пятого дня работы АВП обладает преимуществом перед традиционным веб-приложением и в дальнейшем только увеличивает его.

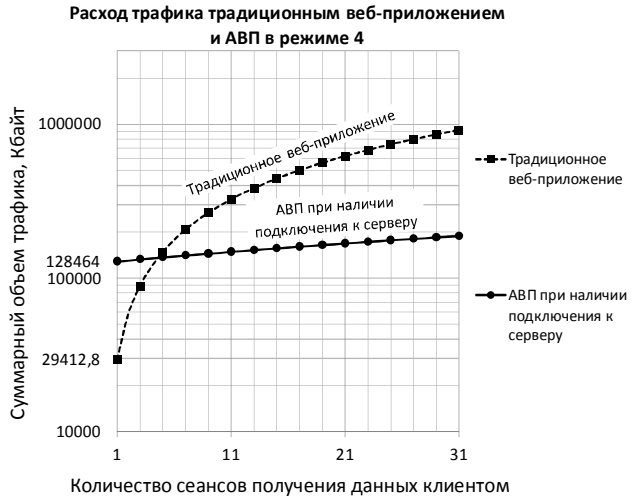


Рис. 3.8. Суммарный расход трафика традиционным веб-приложением и АВП при работе с данными примера №1

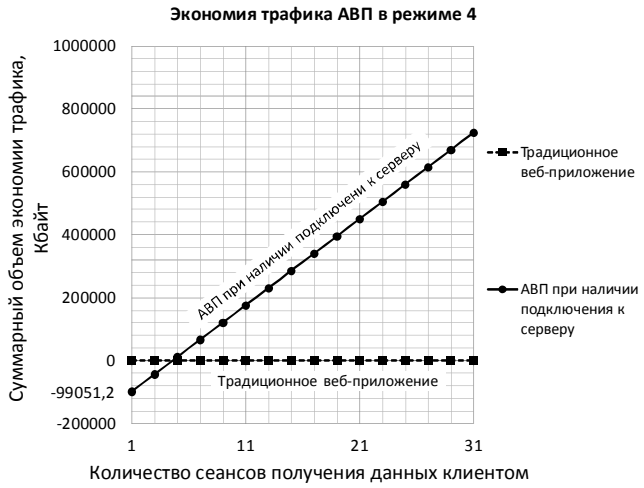


Рис. 3.9. Суммарная экономия трафика АВП при работе с данными примера №1 в сравнении с традиционным веб-приложением

Из рис. 3.10 видно, что время инициализации АВП сравнимо со временем загрузки данных одного сеанса традиционного веб-приложения для рассматриваемого примера (и превышает его всего примерно в 5 раз).

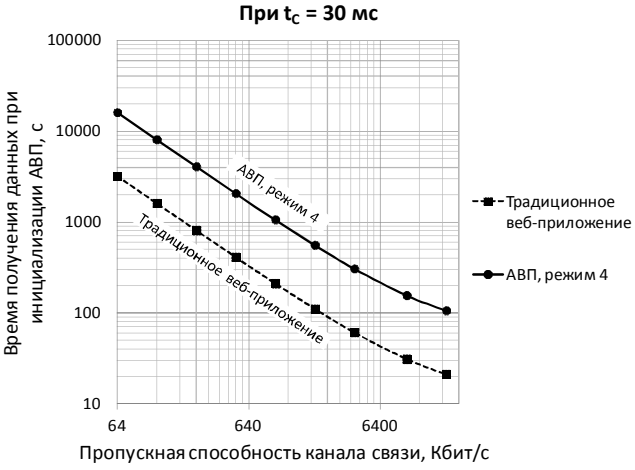


Рис. 3.10. Время получения данных с сервера традиционным веб-приложением и АВП при инициализации $C^{пост.прил}$ примера №1

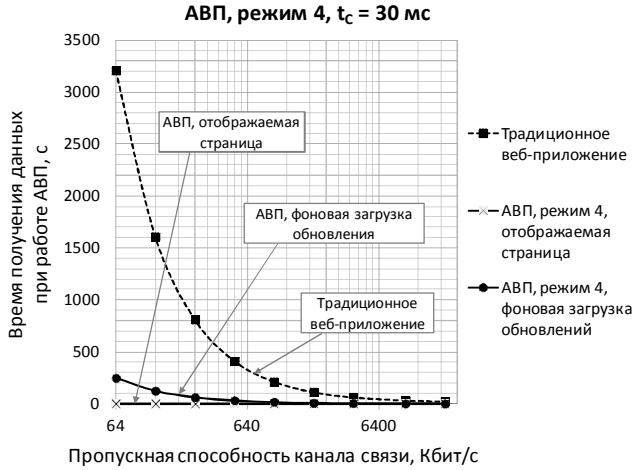


Рис. 3.11. Время получения данных с сервера традиционным веб-приложением и АВП при работе в режиме 4 с использованием данных из $C^{пост.прил}$ примера №1

В результате экономия трафика при использовании АВП по сравнению с традиционным веб-приложением в рассматриваемом примере составляет 31,84% (65550,4 Кбайт) за неделю использования и 79,40% (723957 Кбайт) за месяц.

Суммарная экономия трафика АВП по сравнению с традиционным веб-приложением возрастает пропорционально количеству сеансов работы пользователя с данными, а относительный расход трафика на один сеанс снижается.

При использовании АВП с максимально возможным объемом закэшированных данных (режим 4), учитывая, что все обновления данных АВП и МК загружаются в фоновом режиме, а страницы отображаются пользователю практически мгновенно (рис. 3.11) даже с учетом загрузки обновлений данных экономия времени загрузки данных АВП составляет более 92%.

Аналогичное АВП при изменении пространственного разрешения используемых данных ДЗЗ, периода их обновления и состава векторных слоев может использоваться для различных задач спутникового мониторинга лесного хозяйства: определения границ схода снежного покрова и сроков наступления пожароопасного сезона, определения степени увлажнения лесных горючих материалов, выявления очагов возгорания, слежения за динамикой лесных пожаров, определения зон задымленности от лесных пожаров, оценки изменений в лесном фонде от лесных пожаров и многих других [29].

3.5.2 Использование АВП при работе с данными ДЗЗ при проведении геодезических работ

Рассмотрим примеры использования АВП с данными ДЗЗ при проведении геодезических работ, что обусловлено высокой важностью автоматизации геодезической, картографической и кадастровой деятельности для государственного управления [4, 16, 18, 21], а также внедрением в данной отрасли традиционных веб-приложений, без ограничений пригодных для модернизации до АВП [19].

Для различных типов геодезических и кадастровых задач характерен период актуальности данных ДЗЗ от 1 месяца до 5 лет [12]. Поэтому данные ДЗЗ могут локально кэшироваться АВП на период их актуальности. Для профессионального использования данных ДЗЗ при решении таких задач характерна работа с данными уровней детализации с 14 по 19, обеспечивающих необходимое пространственное разрешение 0,5–10 м [12]. Для экономии необходимого объема локального хранилища данных кэшированию подлежат индивидуальные наборы данных, с которыми работают конкретные специалисты – с ограничением по площади территории и необходимым уровням детализации. Поскольку сохраненное АВП может запускаться сразу с необходимого уровня детализации, то кэширование других уровней не требуется.

Векторные слои и текстовые обозначения могут меняться в течение каждого рабочего дня, но поскольку существует территориальное разделение между специалистами, то изменение информации одними сотрудниками не может повлиять на качество автономной работы других,

а при синхронизации с сервером ГИС все получают актуальную информацию. Синхронизация клиентской части АВП каждого сотрудника с сервером ГИС осуществляется автоматически при наличии подключения к серверу через сеть Интернет. Элементы векторных слоев тоже кэшируются для экономии трафика и обеспечения возможности автономной работы. При этом период актуальности составляет не менее одного рабочего дня.

Применение АВП позволяет реализовать аналогичный предыдущему примеру сценарий работы специалиста (геодезиста – при измерении / выносе участка в натуру или кадастрового инженера – при учете и согласовании границ участков), использующего геоданные, мобильные устройства, и которому это может понадобиться вдали от надежного подключения к серверу ГИС, например, при топографической съемке местности (включая различные объекты ситуации местности – особенности рельефа, растительность, коммуникации, постройки), корректировке топографических планов и обновлении ситуаций местности, подготовке строительства, разрешении спорных ситуаций, других видах геодезических и кадастровых работ.

В качестве примеров проведения геодезических работ рассмотрены работы:

1. на участке площадью 25 км^2, что эквивалентно участку автодороги или трассы коммуникаций протяженностью 50 км. Векторные слои содержат 500 объектов (чертежи расположения объектов и коммуникаций);
2. на участке линии электропередачи (ЛЭП) 330кВ, протяженностью 100 км. Векторные слои содержат 250 объектов (опоры ЛЭП);
3. на участке Москва–Тверь федеральной автомобильной дороги М-10 протяженностью 168 км. Векторные слои содержат 100 объектов (автозаправочные станции).

Отображение автодорог и трасс коммуникаций включает в себя по 0,25 км прилегающей территории в обе стороны от осевой линии [26]. Во всех примерах в качестве растровой подложки для векторных тематических слоев обозначений используются данные ДЗЗ пространственного разрешения не более 10 м. Данные растрового слоя имеют период актуальности 6 месяцев [12], хотя на самом деле могут обновляться не полностью, а только некоторые тайлы, содержащие изменения за указанный период. Относительно векторного слоя допустим, что 10% его объектов обновляются каждые 3 дня.

В таблице 3.13 приведены численные значения преимуществ АВП перед традиционным веб-приложением, обусловленные сохранением АВП на устройстве пользователя необходимых данных на период их актуальности.

Таблица 3.13. Преимущества АВП перед традиционным веб-приложением в примерах проведения геодезических работ

Пример, №	Площадь растровой подложки, км^2	Количество элементов векторных слоев, ед.	Пропускная способность канала передачи данных, Кбит/с	Экономия времени ожидания загрузки данных одного сеанса АВП по сравнению с традиционным веб-приложением, %	Экономия трафика АВП по сравнению с традиционным веб-приложением, %	
					За первую неделю	За первый месяц
2.	25	500	64	**86,46**	**31,89**	**81,04**
			1024	**86,88**		
3.	50	250	64	**94,90**	**31,32**	**82,72**
			1024	**95,10**		
4.	84	100	64	**97,80**	**30,33**	**83,14**
			1024	**97,90**		

Рассмотрим подробнее пример №2 из таблицы 3.13.

В данном примере используемое программное решение АВП содержит:

- растровый слой данных ДЗЗ уровней детализации с 14 по 19, состоящий из 5849 тайлов общим объемом 116980 Кбайт;
- векторный слой, содержащий 500 объектов общим объемом 40960 Кбайт;
- программный код и элементы оформления АВП, включающие в себя средства для отображения данных и взаимодействия с ними пользователя, всего 30 элементов общим объемом 5120 Кбайт;
- МК, управляющий локальным кэшированием полученных данных на стороне клиента, объемом 320 Кбайт.

При первичной загрузке (инициализации) АВП на устройство пользователя загрузится 163380 Кбайт = 159,55 Мбайт = 0,16 Гбайт (что эквивалентно 1/100 доли объема собственной памяти устройства Apple iPad с 16 Гбайт памяти).

Примерно 4096 Кбайт обновляются каждые 3 дня.

Будем считать, что за рассматриваемый период программный код и элементы оформления АВП не обновляются. Сохраненные элементы подаются из локального хранилища на стороне клиента и отображаются пользователю практически мгновенно. При каждом обращении к АВП с сервера загружается МК, который позволяет клиентской части АВП проверить актуальность локально сохраненных данных и управляет обновлением данных в локальном хранилище, если это необходимо.

Таким образом, суммарный трафик при работе с АВП (2.3.7) составит: за неделю (7 дней) – 173492 Кбайт, за месяц (31 день) – 213940 Кбайт.

Традиционным веб-приложением ежедневно загружается 20% от общего объема растровых и векторных данных карты участка (23396 + 8192 Кбайт) и программный код, который состоит из 25 элементов общим объемом 4800 Кбайт, МК – нет. Всего ежедневно загружается примерно 36388 Кбайт.

В результате суммарный трафик при работе с традиционным веб-приложением (2.2.1) составит: за неделю (7 дней) – 254716 Кбайт, за месяц (31 день) – 1128028 Кбайт.

По данным примера построены графики на рис. 3.12, 3.13, 3.14, 3.15, которые иллюстрируют преимущества АВП (2.4.1), (2.4.4) при работе с данными ДЗЗ в геоинформационной системе с веб-интерфейсом перед традиционным веб-приложением.

Из рис. 3.12, 3.13 следует, что начиная уже с пятого дня работы АВП обладает преимуществом перед традиционным веб-приложением и в дальнейшем только увеличивает его.

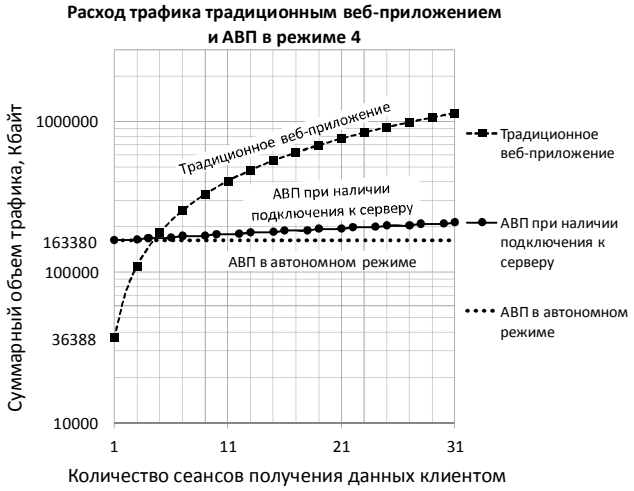


Рис. 3.12. Суммарный расход трафика традиционным веб-приложением и АВП при работе с данными примера №2

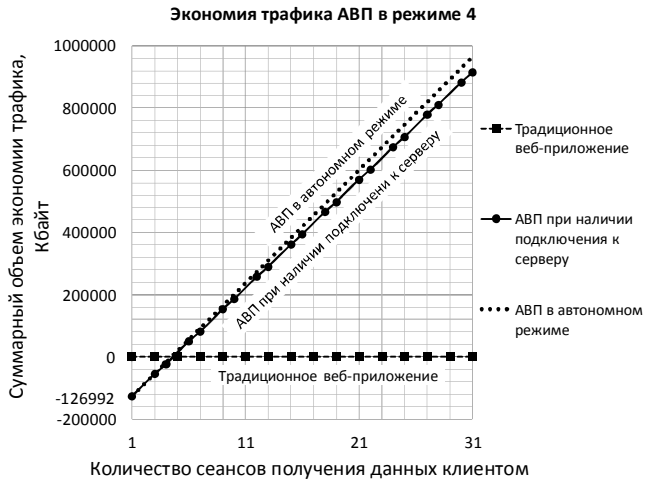


Рис. 3.13. Суммарная экономия трафика АВП при работе с данными примера №2 в сравнении с традиционным веб-приложением

Из рис. 3.14 видно, что время инициализации АВП сравнимо со временем загрузки данных одного сеанса традиционного веб-приложения для рассматриваемого примера (и превышает его всего примерно в 5 раз).

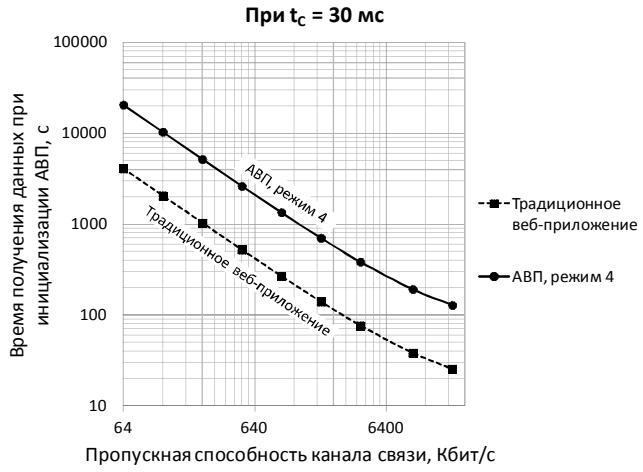


Рис. 3.14. Время получения данных с сервера традиционным веб-приложением и АВП при инициализации $C^{пост.прил}$ примера №2

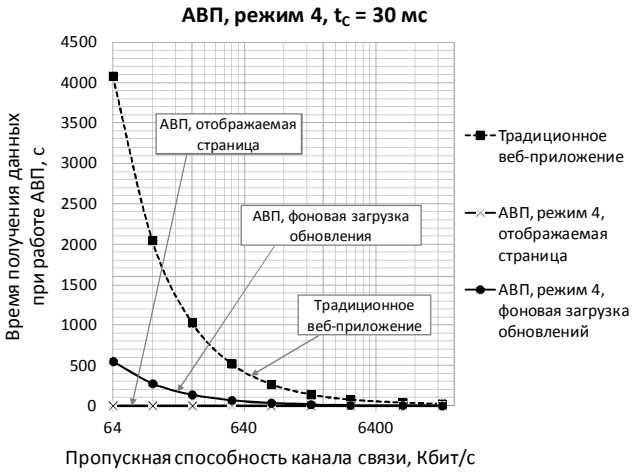


Рис. 3.15. Время получения данных с сервера традиционным веб-приложением и АВП при работе в режиме 4 с использованием данных из $C^{пост.прил}$ примера №2

В результате экономия трафика (2.4.1) при использовании АВП по сравнению с традиционным веб-приложением в рассматриваемом примере составляет 31,89% (81224 Кбайт) за неделю использования и 81,04% (914088 Кбайт) за месяц.

Суммарная экономия трафика АВП по сравнению с традиционным веб-приложением возрастает пропорционально количеству сеансов работы пользователя с данными, а относительный расход трафика на один сеанс снижается.

При использовании АВП с максимально возможным объемом закэшированных данных (режим 4), учитывая, что все обновления данных АВП и МК загружаются в фоновом режиме, а страницы отображаются пользователю практически мгновенно (рис. 3.15) даже с учетом загрузки обновлений данных экономия времени загрузки данных АВП составляет более 86%.

Разработанные АВП наиболее эффективны в случаях, когда пользователь продолжительное время работает с повторно используемыми наборами данных. Разница становится особенно заметной на загруженных каналах связи с малой пропускной способностью.

Результаты сравнительного тестирования АВП и традиционного веб-приложения при работе пользователей ГИС в различных условиях приведены в таблице 3.14.

Таблица 3.14. Результаты сравнительного тестирования АВП и традиционного веб-приложения при работе пользователей ГИС в различных условиях

<table>
<tr><th rowspan="2">Место работы
и виды основных
операций</th><th colspan="2">Хорошая связь
с сервером</th><th colspan="2">Плохая связь
с сервером</th><th colspan="2">Отсутствует связь
с сервером</th></tr>
<tr><th>Традиционное веб-приложение</th><th>АВП</th><th>Традиционное веб-приложение</th><th>АВП</th><th>Традиционное веб-приложение</th><th>АВП</th></tr>
<tr><td>Работа в офисе
1. Загрузка данных с сервера.
2. Отправка данных на сервер.</td><td rowspan="2">Работает</td><td rowspan="2">Работает. Обеспечивает количественные преимущества</td><td rowspan="2">Работает с перебоями. Срок ожидания загрузки данных увеличивается пропорционально количеству и продолжительности разрывов соединения, т.к. для отображения требуется полный набор данных. Вводимые данные могут быть потеряны</td><td rowspan="2">Работает автономно с резервированием вводимых пользователем данных. При появлении соединения с сервером отправляет данные на сервер, проверяет обновления. Обеспечивает качественные и количественные преимущества</td><td rowspan="2">Неработоспособно</td><td rowspan="2">Работает автономно с резервированием вводимых пользователем данных. Обеспечивает качественные преимущества</td></tr>
<tr><td>Работа на выезде
1. Просмотр спутниковых снимков в интересующей точке.
2. Пометка интересующих точек/областей.
3. Просмотр метки и описания, введенных для точки/области.
4. Нанесение своих точек/областей, меток и описаний, а также редактирование имеющихся и обеспечение доступности этой информации для других пользователей данной ГИС.</td></tr>
</table>

Преимущества АВП перед традиционным веб-приложением подтверждены тестовой эксплуатацией на примерах внедрения. В целях определения реальных значений t_C были проведены измерения времени ответа тестового сервера по протоколу ICMP [64] эхо-запросами длиной 64 байта с IP-заголовками 20 байт [63] с использованием распространенных технологий передачи данных мобильной связи: LTE, HSPA+, UMTS, EDGE. Замеры производились в течение 3 недель в г. Москве, Московской и Калужской областях с 8:00 до 23:00 часов по московскому времени с использованием персональных мобильных устройств, которые позволяют работать с АВП: смартфон Sony Xperia Z3, планшет Apple iPad mini, телефон Nokia 3120 Classic в качестве модема для ПК. Всего проведено 592 измерения.

Построены распределения частости измеренных значений t_C по диапазонам времени отклика сервера отдельно для каждой рассмотренной технологии передачи данных. Распределения показаны на общем графике на рис. 3.16.

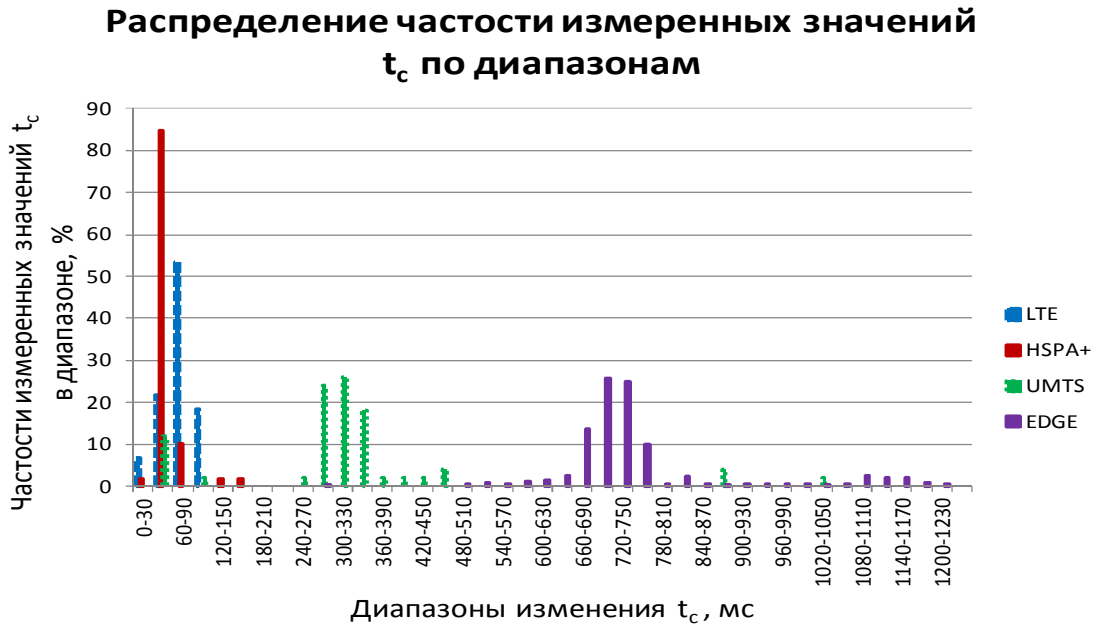


Рис. 3.16. Распределение частости измеренных значений t_C по диапазонам

При работе с АВП заранее неизвестно, по какой из технологий мобильной передачи данных будет осуществляться связь в рамках сеанса. Технологии связи могут произвольно сменяться в рамках одного сеанса в зависимости от условий приема, а современные устройства, используемые для работы с АВП, могут поддерживать сразу все или некоторое подмножество перечисленных технологий мобильной передачи данных.

На графике (рис. 3.16) показано, что лишь в 6,67% запросов для LTE и 1,69% запросов для HSPA+ время отклика сервера было менее 30 мс. При использовании других технологий время отклика сервера было более 30 мс.

Средние значения измеренного времени отклика сервера составили: для LTE 72,04 мс, для HSPA+ 49,89 мс, для UMTS 323,92 мс, для EDGE 790,78 мс. Это означает, что преимущества АВП перед традиционным веб-приложением, оцененные для различных значений пропускной способности каналов передачи данных при t_C = 30 мс, в реальности еще больше, так как измеренное среднее время отклика сервера при использовании мобильной связи значительно превышает 30 мс.

Тестирование показало, что при хорошей связи с сервером АВП обеспечивает количественные преимущества по сравнению с традиционным веб-приложением, при плохой – качественные и количественные, а при ее полном отсутствии – качественные.

ЗАКЛЮЧЕНИЕ

1. Выполнен анализ специфики геоданных и структуры веб-представлений геоданных с учетом комбинации растровых и векторных слоев многослойной интерактивной карты.

2. В работе показано, что за счет использования выявленной специфики геоданных и структуры веб-представлений геоданных, а также технологий, базирующихся на стандарте HTML5, возможно создание нового типа веб-приложений – АВП, которое позволяет расширить множество возможных терминальных устройств для систем мониторинга земной поверхности и ГИС с использованием данных ДЗЗ.

3. Разработаны метод функционирования и архитектура АВП, которое может использоваться в качестве клиентского приложения для систем мониторинга земной поверхности и ГИС с использованием данных ДЗЗ для автономной работы с полученными с сервера данными, автоматического аварийного резервирования вводимых пользователем данных и поддержки различных клиентских платформ.

4. Предложенное решение распространяется, в первую очередь, на мобильные устройства, для которых особенно актуальна возможность резервирования вводимых пользователем данных для достижения отказоустойчивости и возможность автономной работы.

5. Построены и проанализированы математические модели для оценки количественных преимуществ использования АВП для доступа к геоданным. Использование разработанных моделей показало преимущества АВП по сравнению с традиционным веб-приложением: экономию трафика – более 30%; уменьшение времени загрузки обновлений данных – более 86%; возможность моментального запуска.

6. Предложена архитектура АВП, основу которой составляют объединенные разработанными в диссертации алгоритмами элементы стандарта HTML5 (Local Storage, Application Cache), библиотека jQuery, а также концепция построения интерактивных веб-интерфейсов AJAX. В ходе экспериментов предложенная архитектура показала свою работоспособность для решения поставленных задач.

7. Согласно предложенной архитектуре реализован и протестирован программный комплекс АВП, который использовался для проведения названных выше натурных экспериментов и подтверждения достоверности разработанных моделей.

8. Заявленные преимущества АВП подтверждены в ходе натурных экспериментов на базе реализованных АВП на примерах решения задач: а) мониторинга лесного хозяйства; б) проведения геодезических работ.

9. Развитие исследуемой темы может осуществляться в следующих направлениях:

- использование в АВП других технологий локального сохранения данных средствами веб-приложения;
- повышение автономности клиентов АВП за счет обеспечения прямого обмена данными между ними;
- развитие применения АВП в других областях, для которых характерна работа пользователей с большими объемами данных подверженными постоянным небольшим изменениям, получаемым с сервера.

СПИСОК ЛИТЕРАТУРЫ

1. Векторные данные [Электронный ресурс] // Документация QGIS2.8. – Режим доступа: http://docs.qgis.org/2.8/ru/docs/gentle_gis_introduction/vector_data.html (дата обращения: 24.02.2016).
2. Заичко, В. А. Российская космическая система ДЗЗ: состояние и перспективы развития [Электронный ресурс] / В. А. Заичко // II Международный форум «Интеграция геопространства – будущее информационных технологий» 23 апреля 2014 г. – Систем. требования: Adobe Acrobat Reader. – Режим доступа: ftp://ftp.sovzond.ru/forum/2014/reports/Zaichko.pdf (дата обращения: 24.02.2016).
3. Знакомство с хранилищем DOM [Электронный ресурс] // Microsoft Developer Network (MSDN). – Режим доступа: http://msdn.microsoft.com/ru-ru/library/cc197062(v=VS.85).aspx (дата обращения 24.02.2016).
4. Золотова, Е. В. Основы кадастра: Территориальные информационные системы: Учебник для вузов / Е. В. Золотова. – М.: Академический Проект; Фонд «Мир», 2012. – 416 с.
5. Геоинформатика: Учеб. для студ. вузов / Е. Г. Капралов, А. В. Кошкарев, В. С. Тикунов [и др.]; Под ред. В. С. Тикунова. – М: Академия, 2005. – 480 с.
6. Гинзбург, И. Б. Автономные отказоустойчивые веб-приложения для систем обеспечения доступа к данным дистанционного зондирования Земли [Электронный ресурс] / И. Б. Гинзбург // Журнал «Труды МАИ». – 2015. – №84. – Систем. требования: Adobe Acrobat Reader. – Режим доступа: http://www.mai.ru/science/trudy/published.php?ID=63149 (дата обращения: 24.02.2016).
7. Гинзбург, И. Б. Концепция построения распределенных систем информационной поддержки технического обслуживания аэрокосмической техники с использованием функционально насыщенных веб-клиентов / И. Б. Гинзбург // Научно-технический вестник Поволжья. – 2014. – №5. – С. 159 –161.
8. Гинзбург, И. Б. Автономные веб-приложения для систем обработки космической информации [Электронный ресурс] / И. Б. Гинзбург, С. Н. Падалко // Журнал «Труды МАИ». – 2015. – №82. – Систем. требования: Adobe Acrobat Reader. – Режим доступа: http://www.mai.ru/science/trudy/published.php?ID=58832 (дата обращения: 24.02.2016).
9. Гинзбург И. Б. Состав и архитектура взаимодействия модулей функционально насыщенного автономного веб-приложения для распределенных систем информационной поддержки различных этапов жизненного цикла аэрокосмической техники / И. Б. Гинзбург // Научно-технический вестник Поволжья. – 2014. – №6. – С. 130–133.

10. Департамент информационных технологий совместно с Роскомнадзором подвели итоги измерений качества сотовой связи в пилотной зоне Москвы [Электронный ресурс] // Департамент информационных технологий города Москвы. – Режим доступа: http://dit.mos.ru/presscenter/news/detail/799428.html (дата обращения: 24.02.2016).
11. Дискретные и непрерывные данные [Электронный ресурс] // Справка ArcGIS 10.1. – Режим доступа: http://resources.arcgis.com/ru/help/main/10.1/index.html#//009t00000007000000 (дата обращения: 24.02.2016).
12. Концепция развития российской космической системы дистанционного зондирования Земли на период до 2025 года [Электронный ресурс] // Геоинформационный портал ГИС-Ассоциации. – Систем. требования: Microsoft Word или LibreOffice Writer. – Режим доступа: http://www.gisa.ru/file/file766.doc (дата обращения: 24.02.2016).
13. Коптев, Ю. Н. Космонавтика России на рубеже XXI столетия / Ю. Н. Коптев, В. В. Алавердов, Б. В. Бодин // Исследования по истории и теории развития авиационной и ракетно-космической науки и техники. – Вып. 8-10. – М.: Наука, 2001.
14. Космонавтика XXI века. Попытка прогноза развития до 2101 года / Под ред. Б. Е. Чертока. – М.: РТСофт, 2010. – 864 с.
15. Лупян, Е. А. Современные подходы и технологии организации работы с данными дистанционного зондирования Земли для решения научных задач / Е. А. Лупян, В. П. Саворский, Ю. И. Шокин, А. И. Алексанян, Р. Р. Назиров, И. В. Недолужко, О. Ю. Панова // Современные проблемы дистанционного зондирования Земли из космоса. – 2012. – Т. 9. – № 5. – С. 21-44.
16. Министерство экономического развития Российской Федерации – Деятельность – Страница Картография [Электронный ресурс] // Официальный сайт Минэкономразвития России. – Режим доступа: http://economy.gov.ru/minec/activity/sections/geodesyandcartography/cartography/ (дата обращения 24.02.2016).
17. Народная карта Яндекса [Электронный ресурс]. – Режим доступа: https://n.maps.yandex.ru/ (дата обращения: 24.02.2016).
18. Постановление Правительства Российской Федерации от 17.12.2014 № 1390 "О публичном использовании данных дистанционного зондирования Земли из космоса, получаемых с зарубежных космических аппаратов и российских космических аппаратов гражданского назначения" [Электронный ресурс] // Официальный интернет-портал правовой информации. Государственная система правовой информации. – Режим доступа: http://publication.pravo.gov.ru/Document/View/0001201412190033 (дата обращения 24.02.2016).
19. Публичная кадастровая карта [Электронный ресурс] // Росреестр. – Режим доступа:

http://maps.rosreestr.ru/PortalOnline/ (дата обращения: 24.02.2016).

20. Раклов, В. П. Географические информационные системы в тематической картографии: Учебное пособие для вузов / В. П. Раклов. – 4-е изд. – М.: Академический проект, 2014. – 176 с.
21. Распоряжение Правительства Российской Федерации от 17 декабря 2010 г. N 2378-р г. Москва [Электронный ресурс] // Интернет-портал "Российской газеты". – Режим доступа: http://www.rg.ru/2011/01/11/geodeziya-site-dok.html (дата обращения 24.02.2016).
22. Руководство пользователя [Электронный ресурс] // Геопортал Роскосмоса. – Режим доступа: http://gptl.ru/help/help.html (дата обращения: 24.02.2016).
23. Сервис космических снимков [Электронный ресурс] // Геопортал Роскосмоса. – Режим доступа: http://gptl.ru (дата обращения: 24.02.2016).
24. Сети сотовой подвижной связи. Методика проведения оценочных испытаний и нормы на показатели качества услуг связи стандарта GSM/GPRS/EDGE/UMTS [Электронный ресурс] // Департамент информационных технологий города Москвы. – Систем. требования: Adobe Acrobat Reader. – Режим доступа: http://dit.mos.ru/signal/documents/Методика_июнь_2013.pdf (дата обращения: 24.02.2016).
25. Технология AJAX – усовершенствования подключения в Internet Explorer 8 [Электронный ресурс] // Microsoft Developer Network (MSDN). – Режим доступа: http://msdn.microsoft.com/ru-ru/library/cc304129(v=vs.85).aspx (дата обращения 24.02.2016).
26. Федеральный закон от 08.11.2007 N 257-ФЗ (ред. от 13.07.2015) "Об автомобильных дорогах и о дорожной деятельности в Российской Федерации и о внесении изменений в отдельные законодательные акты Российской Федерации" (с изм. и доп., вступ. в силу с 15.11.2015). Статья 26. Придорожные полосы автомобильных дорог [Электронный ресурс] // Официальный сайт компании "КонсультантПлюс". – Режим доступа: https://www.consultant.ru/cons/document/cons_doc_LAW_72386/5cc1c49fd81cc0437144e5ddf4902fdf0fe0a7ea/ (дата обращения 24.02.2016).
27. Хокинс, С. Администрирование веб-сервера Apache и руководство по электронной коммерции / С. Хокинс. – М.: Вильямс, 2001. – 336 с.
28. Что такое растровые данные? [Электронный ресурс] // Справка ArcGIS 10.1. – Режим доступа: http://resources.arcgis.com/ru/help/main/10.1/index.html#//009t00000002000000 (дата обращения: 24.02.2016).
29. Шимов, С. В. Технология мониторинга вырубок леса с использованием космических снимков высокого пространственного разрешения [Электронный ресурс] / С. В. Шимов, Ю. В. Никитина // ГЕОМАТИКА. – 2011. – №3. – Систем. требования: Adobe Acrobat Reader. – Режим доступа:

http://en.sovzond.ru/upload/iblock/cd2/7shimov_tekhnologia_monitoringa_virybok_lesa.pdf (дата обращения 02.06.2016)

30. Average Web Page Breaks 1600K [Электронный ресурс] // Web Site Optimization: Speed Up Your Site website optimization web speed optimize web site performance company. – Режим доступа: http://www.websiteoptimization.com/speed/tweak/average-web-page/ (дата обращения 24.02.2016).
31. Bolstad, P. GIS Fundamentals: A First Text on Geographic Information Systems / P. Bolstad. – 4th ed. – Ann Arbor, MI: XanEdu Publishing, 2012. – 688 p.
32. Boutell, T. WWW FAQs: What is the maximum length of a URL? [Электронный ресурс] / T. Boutell // The New WWW FAQs. – Режим доступа: http://www.boutell.com/newfaq/misc/urllength.html (дата обращения 24.02.2016).
33. Brody, H. How HTTPS Secures Connections: What Every Web Dev Should Know [Электронный ресурс] / H. Brody // Hartley Brody's Blog. – Режим доступа: http://blog.hartleybrody.com/https-certificates/ (дата обращения: 24.02.2016).
34. Can I use offline web applications? [Электронный ресурс] // Can I use... Support tables for HTML5, CSS3, etc. – Режим доступа: http://caniuse.com/offline-apps (дата обращения 24.02.2016).
35. Can I use Web Storage? [Электронный ресурс] // Can I use... Support tables for HTML5, CSS3, etc. – Режим доступа: http://caniuse.com/namevalue-storage (дата обращения 24.02.2016).
36. Cameron, D. A Software Engineer Learns HTML5, JavaScript and jQuery / D. Cameron. – Wellington, New Zealand: Cisdal Publishing, 2013. – 256 p.
37. Campbell, J. B. Introduction to Remote Sensing / J. B. Campbell, R. H. Wynne. – 5th ed. – New York, NY: The Guilford Press, 2011. – 667 p.
38. Categories of Free and Nonfree Software [Электронный ресурс] // GNU Project - Free Software Foundation. – Режим доступа: http://www.gnu.org/philosophy/categories.en.html (дата обращения: 24.02.2016).
39. Cunningham, W. The Wiki Way: Quick Collaboration on the Web / W. Cunningham, B. Leuf. – Reading, MA: Addison-Wesley, 2001. – 464 p.
40. Dixit, S. Running Your Web Applications Offline With HTML5 AppCache [Электронный ресурс] / S. Dixit // Dev.Opera. – Режим доступа: http://dev.opera.com/articles/offline-applications-html5-appcache/ (дата обращения 24.02.2016).
41. Dixit, S. Web Storage: Easier, More Powerful Client-Side Data Storage [Электронный ресурс] / S. Dixit // Dev.Opera. – Режим доступа: http://dev.opera.com/articles/web-storage/ (дата обращения 24.02.2016).

42. Flanagan, D. JavaScript: The Definitive Guide / D. Flanagan. – 6th ed. – Sebastopol, CA: O'Reilly Media, 2011. – 1096 p.
43. Flinn, J. Reducing the energy usage of office applications / J. Flinn, E. de Lara, M. Satyanarayanan, D. S. Wallach, W. Zwaenepoel // Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms, ser. Middleware – Heidelberg '01, 2001. – pp. 252–272.
44. Forms in HTML documents. HTML 4.01 Specification [Электронный ресурс] // World Wide Web Consortium (W3C). – Режим доступа: http://www.w3.org/TR/html401/interact/forms.html#h-17.13.4.2 (дата обращения 24.02.2016).
45. Freeman, A. Pro JavaScript for Web Apps / A. Freeman. – New York, NY: Apress, 2012. – 280 p.
46. Freeman, A. Pro jQuery / A. Freeman. – New York, NY: Apress, 2012. – 1016 p.
47. Fu, P. Getting to Know Web GIS / P. Fu. – Redlands, CA: Esri Press, 2015. – 400 p.
48. Fu, P. Web GIS: Principles and Applications / P. Fu, J. Sun. – Redlands, CA: Esri Press, 2010. – 312 p.
49. Gourley, D. HTTP: The Definitive Guide / D. Gourley, B. Totty. – Sebastopol, CA: O'Reilly Media, 2002. – 635 p.
50. Holzner, S. Ajax Bible / S. Holzner. – Indianapolis, IN: Wiley, 2007. – 695 p.
51. HTML5. A vocabulary and associated APIs for HTML and XHTML. W3C Recommendation 28 October 2014 [Электронный ресурс] / Editors: I. Hickson, R. Berjon, S. Faulkner, T. Leithead, E. D. Navara, E. O'Connor, S. Pfeiffer // World Wide Web Consortium (W3C). – Режим доступа: http://www.w3.org/TR/html5/ (дата обращения: 24.02.2016).
52. HTTP Archive – Trends [Электронный ресурс]. – Режим доступа: http://httparchive.org/trends.php (дата обращения: 24.02.2016).
53. Internet Explorer error "connection timed out" when server does not respond [Электронный ресурс] // Microsoft Support. – Режим доступа: https://support.microsoft.com/en-us/kb/181050 (дата обращения 24.02.2016).
54. jQuery.ajax() API Documentation [Электронный ресурс] // jQuery Core API Documentation. – Режим доступа: http://api.jquery.com/jquery.ajax/ (дата обращения 24.02.2016).
55. Lengstorf, J. Pro PHP and jQuery / J. Lengstorf. – New York, NY: Apress, 2010. – 400 p.
56. Loading Web pages – HTML5. Offline Web applications. [Электронный ресурс] // World Wide Web Consortium (W3C). – Режим доступа: http://www.w3.org/TR/html5/browsers.html#offline (дата обращения 24.02.2016).
57. OpenGIS Web Map Tile Service Implementation Standard [Электронный ресурс] // Open Geospatial Consortium. – Режим доступа: http://www.opengeospatial.org/standards/wmts (дата обращения: 21.04.2016).

58. Pilgrim, M. HTML5: Up and Running / M. Pilgrim. – Sebastopol, CA: O'Reilly Media, 2010. – 222 p.
59. Powell, T. A. Ajax: The Complete Reference / T. A. Powell. – New York, NY: McGraw-Hill, 2008. – 654 p.
60. RFC 1952 GZIP File Format Specification version 4.3 [Электронный ресурс] // zlib Home Site. – Режим доступа: http://www.zlib.org/rfc-gzip.html (дата обращения 24.02.2016).
61. RFC 2616 - Hypertext Transfer Protocol - HTTP/1.1 (RFC2616) [Электронный ресурс] // Internet FAQ Archives - Online Education. – Режим доступа: http://www.faqs.org/rfcs/rfc2616.html (дата обращения 24.02.2016).
62. RFC 7230 - Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing. Request Line [Электронный ресурс] // Internet Engineering Task Force (IETF). – Режим доступа: http://tools.ietf.org/html/rfc7230#section-3.1.1 (дата обращения 24.02.2016).
63. RFC 791 - Internet Protocol [Электронный ресурс] // Internet Engineering Task Force (IETF). – Режим доступа: https://tools.ietf.org/html/rfc791 (дата обращения: 21.04.2016).
64. RFC 792 - Internet Control Message Protocol [Электронный ресурс] // Internet Engineering Task Force (IETF). – Режим доступа: https://tools.ietf.org/html/rfc792 (дата обращения: 21.04.2016).
65. Rice, A. Measuring mobile phone energy consumption for 802.11 wireless networking / A. Rice, S. Hay // Pervasive and Mobile Computing. – 2010. – Vol. 6. – pp. 593–606.
66. Richardson, L. RESTful Web Services / L. Richardson, S. Ruby. – Sebastopol, CA: O'Reilly Media, 2008. – 454 p.
67. Rivera, J. Gartner Says Worldwide Device Shipments to Grow 1.5 Percent, to Reach 2.5 Billion Units in 2015 [Электронный ресурс] / J. Rivera, L. Goasduff // Technology Research. Gartner Inc. – Режим доступа: http://www.gartner.com/newsroom/id/3088221 (дата обращения: 24.02.2016).
68. ScanEx Web Geomixer - просмотр карты [Электронный ресурс] // Веб-ГИС GeoMixer. – Режим доступа: http://maps.kosmosnimki.ru/api/index.html (дата обращения: 24.02.2016).
69. Sitemaps XML format [Электронный ресурс] // Sitemap Protocol. – Режим доступа: http://www.sitemaps.org/protocol.html (дата обращения 24.02.2016).
70. Slippy map tilenames [Электронный ресурс] // OpenStreetMap Wiki. – Режим доступа: http://wiki.openstreetmap.org/wiki/Slippy_map_tilenames (дата обращения: 24.02.2016).
71. Souders, S. Even Faster Web Sites: Performance Best Practices for Web Developers / S. Souders. – Sebastopol, CA: O'Reilly Media, 2009. – 256p.
72. Souders, S. High Performance Web Sites: Essential Knowledge for Front-End Engineers / S. Souders. – Sebastopol, CA: O'Reilly Media, 2007. – 170p.

73. Souders, S. Roundup on Parallel Connections [Электронный ресурс] / S. Souders // Steve Souders – High Performance Web Sites. – Режим доступа: http://www.stevesouders.com/blog/2008/03/20/roundup-on-parallel-connections/ (дата обращения 24.02.2016).
74. TileCache, from MetaCarta Labs [Электронный ресурс] // TileCache Contributors. – Режим доступа: http://tilecache.org/ (дата обращения: 21.04.2016).
75. Tile Map Service Specification [Электронный ресурс] // OSGeo Wiki. – Режим доступа: http://wiki.osgeo.org/wiki/Tile_Map_Service_Specification (дата обращения: 21.04.2016).
76. Using the application cache [Электронный ресурс] // Mozilla Developer Network (MDN). – Режим доступа: https://developer.mozilla.org/en-US/docs/Web/HTML/Using_the_application_cache (дата обращения 24.02.2016).
77. Web Map Service [Электронный ресурс] // Open Geospatial Consortium. – Режим доступа: http://www.opengeospatial.org/standards/wms (дата обращения: 21.04.2016).
78. Web Storage API [Электронный ресурс] // Mozilla Developer Network (MDN). – Режим доступа: https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API (дата обращения 24.02.2016).
79. Web Storage (Second Edition) [Электронный ресурс] // World Wide Web Consortium (W3C). – Режим доступа: http://www.w3.org/TR/webstorage/ (дата обращения: 24.02.2016).
80. WhooTS a small wms to tile proxy – WMS in Potlatch [Электронный ресурс] // thinkwhere. A Geo blog from Tim Waters. – Режим доступа: https://thinkwhere.wordpress.com/2010/07/15/whoots-a-small-wms-to-tile-proxy-wms-in-potlatch/ (дата обращения: 21.04.2016).
81. Wikimapia - Let's describe the whole world! [Электронный ресурс] – Режим доступа: http://wikimapia.org/ (дата обращения: 24.02.2016).
82. WMS [Электронный ресурс] // OpenStreetMap Wiki. – Режим доступа: http://wiki.openstreetmap.org/wiki/WMS (дата обращения: 21.04.2016).
83. WMS Tiling Client Recommendation [Электронный ресурс] // OSGeo Wiki. – Режим доступа: http://wiki.osgeo.org/wiki/WMS_Tiling_Client_Recommendation (дата обращения: 21.04.2016).
84. Zakas, N. C. High Performance JavaScript / N. C. Zakas. – Sebastopol, CA: O'Reilly Media, 2010. – 231 p.
85. Zakas, N. C. Professional JavaScript for Web Developers / N. C. Zakas. – 3rd ed. – Indianapolis, IN: Wrox, 2012. – 960 p.
86. Zoom levels [Электронный ресурс] // OpenStreetMap Wiki. – Режим доступа: http://wiki.openstreetmap.org/wiki/Zoom_levels (дата обращения: 24.02.2016).