МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт **математики, физики и информационных технологий**
Кафедра **«Прикладная математика и информатика»**

01.03.02 ПРИКЛАДНАЯ МАТЕМАТИКА И ИНФОРМАТИКА

СИСТЕМНОЕ ПРОГРАММИРОВАНИЕ И КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ

БАКАЛАВРСКАЯ РАБОТА

на тему Разработка мобильного приложения для информационной вики-системы с доступом в режиме без прямого подключения

Студент Г. В. Гринько

Руководитель А. В. Очеповский

Допустить к защите
Заведующий кафедрой к.тех.н, доцент, А.В. Очеповский _____

«_____»_____20_____г.

Тольятти 2016

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Тольяттинский государственный университет»

Институт **математики, физики и информационных технологий**

Кафедра **«Прикладная математика и информатика»**

УТВЕРЖДАЮ

Зав. кафедрой «Прикладная математика и информатика»
_____ А.В. Очеповский
«____» _____ 2016 г.

ЗАДАНИЕ

на выполнение бакалаврской работы

Студент: Гринько Григорий Викторович

1. Тема

Разработка мобильного приложения для информационной вики-системы с доступом в режиме без прямого подключения

2. Срок сдачи студентом законченной бакалаврской работы

24.06.2016

3. Исходные данные к бакалаврской работе

Вики-система на базе вики-движка MediaWiki, MediaWiki API, мобильное устройство на базе операционной системы Android OS, Android API

4. Содержание бакалаврской работы (перечень подлежащих разработке вопросов, разделов)

Введение

Глава 1 Теоретические аспекты создания мобильного приложения для вики-системы

- 1.1 Предпосылки вики-системы
- 1.2 Вики-система
- 1.3 Вики-система и мобильные устройства
- 1.4 Формирование требований к разрабатываемому приложению

Глава 2 Анализ и применение технологий вики-систем

- 2.1 Выбор вики-системы для разработки мобильного приложения
- 2.2 Основные особенности вики-движка MediaWiki
- 2.3 MediaWiki API

2.4 Использование MediaWiki API в приложении

Глава 3 Проектирование и разработка мобильного приложения для работы с вики-системой

3.1 Доступ к статьям вики-системы

3.2 Подгрузка статьи с помощью MediaWiki API

3.3 Разработка мобильного приложения с помощью Android SDK

3.3.1 Выбор версии Android API

3.3.2 Проектирование основной части мобильного приложения

3.3.3 Отображение статьи в приложении

Заключение

Список используемой литературы

5. Ориентировочный перечень графического и иллюстративного материала

Блок-схемы, диаграммы, графическое представление мобильного приложения

6. Дата выдачи задания «11» января 2016 г.

Руководитель бакалаврской работы	_____	А. В. Очеповский
Задание принял к исполнению	_____	Г. В. Гринько

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Тольяттинский государственный университет»

Институт **математики, физики и информационных технологий**

Кафедра **«Прикладная математика и информатика»**

УТВЕРЖДАЮ

Зав. кафедрой «Прикладная математика и информатика»
_____ А.В. Очеповский
«____» _____ 2016 г.

КАЛЕНДАРНЫЙ ПЛАН

выполнения бакалаврской работы

Студента Гринько Григория Викторовича

по теме Разработка мобильного приложения для информационной вики-системы с доступом в режиме без прямого подключения

Наименование раздела работы	Плановый срок выполнения раздела	Фактический срок выполнения раздела	Отметка о выполнении	Подпись руководителя
Изучение строения вики-системы	11.01.2016	11.01.2016	Выполнено	
Изучение MediaWiki API	19.01.2016	19.01.2016	Выполнено	
Описание алгоритма работы приложения	8.03.2016	8.03.2016	Выполнено	
Оформление первой главы	22.03.2016	22.03.2016	Выполнено	
Оформление второй главы	10.04.2016	10.04.2016	Выполнено	
Реализация приложения	28.04.2016	28.04.2016	Выполнено	
Оформление третьей главы	6.05.2016	6.05.2016	Выполнено	

Оформление заключения	9.05.2016	9.05.2016	Выполнено	
Создание презентационного материала	10.05.2016	10.05.2016	Выполнено	
Предварительная защита бакалаврской работы	31.05.2016 - 14.06.2016	31.05.2016 - 14.06.2016	Выполнено	
Проверка на наличие заимствований (плагиата) в системе antiplagiat.ru	17.06.2016	17.06.2016	Выполнено	
Сдача на кафедру отзыва научного руководителя и ознакомление с ним	20.06.2016	20.06.2016	Выполнено	
Сдача на кафедру комплекта документов для защиты	24.06.2016	24.06.2016	Выполнено	
Защита бакалаврской работы	27.06.2016 - 30.06.2016	29.06.2016	Выполнено	

Руководитель бакалаврской работы _____ А. В. Очеповский

Студент _____ Г. В. Гринько

Аннотация

Тема: Разработка мобильного приложения для информационной вики-системы с доступом в режиме без прямого подключения.

Работа выполнена студентом Тольяттинского государственного университета, института математики, физики и информационных технологий, группы ПМИб-1201, Гринько Григорием Викторовичем.

Объект исследования: приложение для мобильных устройств, вики-система.

Предмет исследования: система доступа к вики-системе, а также к информации, хранящейся на ней в автономном доступе, на мобильном устройстве.

Цель исследования: обеспечение доступа к информации, хранящейся на вики-системе, с мобильного устройства, с последующим доступом к сохранённым в памяти устройства статьям без прямого подключения и синхронизацией изменений в статьях между хранилищем мобильного устройства и вики-системой.

Поставленная перед исследованием цель требует решить следующие задачи:

- рассмотрение работы текущих популярных вики-систем и установка необходимых требований для вики-системы;
- выбор вики-движка, отвечающего требованием и подробное ознакомление с выбранным движком;
- разработка алгоритма доступа к содержимому на мобильном устройстве без прямого подключения с системой синхронизации содержимого статей с последующей реализацией.

Работа состоит из введения, трёх глав и заключения.

Первая глава посвящена теоретическим аспектам создания мобильного приложения для с вики-системы. В тексте главы разбираются технологии вики,

вики-системы, их текущее взаимодействие с мобильными устройствами и теоретические предпосылки к разработке мобильного приложения.

Вторая глава определяет требования к вики-движку как к части системы с мобильным приложением. В главе проходит анализ существующих вики-движков, выбор лучшего вики-движка, исходя из выдвинутых требований, и подробный разбор API, предоставляемого этим движком.

Третья глава описывает процесс проектирования и разработки мобильного приложения как части системы. В тексте главы описываются алгоритмы доступа к статьям, реализация приложения на Android SDK.

Бакалаврская работа представлена на 50z страницах, включает в себя 18 иллюстраций, 3 таблицы; список используемой литературы содержит 23 источника.

Оглавление

Введение

Актуальность бакалаврской работы заключается в растущем объёме рынка мобильных устройств и при удобстве использования таких устройств на ходу. Со временем появился сильный тренд переносить всё больше и больше функционала, доступного ранее на рабочих станциях, непосредственно на мобильное устройство [16]. Существующие встроенные решения для «смартфонов» в вики-системах реализованы далеко не самым лучшим способом и рассчитаны на пребывание в зоне действия сети для проведения операций с системой, к примеру, запрос статьи [10].

Мобильный браузер плохо справляется с поставленной задачей обрабатывать большие скрипты встроенного редактора вики-системы, размером в десятки тысяч строк [4], что приводит к многочисленным ошибкам в работе и даже к сбою всей системы, вплоть до перезагрузки устройства. По мнению автора, мобильное приложение, использующее преимущества конкретной системы, должно решить проблему автономного доступа к контенту вики-системы, добавляя к этому возможность редактирования содержимого прямо на устройстве пользователя, с дальнейшей синхронизацией.

Требование пользователей иметь доступ к необходимой информации везде и всегда обуславливает **актуальность** бакалаврской работы, так как в настоящее время устройствами на операционной системе Android OS владеет подавляющее число пользователей, ввиду большого фактора доступности, лёгкости в использовании и большой базы приложений, разнящихся от мобильных игр до целых офисных пакетов [16].

Новизна исследования заключается в факторе использования мобильного приложения с вики-системой посредством предоставляемого API, а не генерируемой вики-движком страницы, использующую дополнительные скрипты и таблицы стилей для отображения.

Фактор использования вики-систем как хранилища знаний уже давно привлекает всё новых пользователей. Как и в Википедии, самой крупной вики-системе, в пользовательских вики возможно использование вики-движков, позволяющих структурировать данные с использованием мощного API, доступного через HTTP с момента установки и конфигурации развёрнутой вики-системы [8]. Даже развёрнутая пользователем вики-система способна работать

через запросы к API, чтобы совершать работу с контентом даже с мобильных устройств посредством отправки запросов, к примеру, из адресной строки. Однако такие HTTP-запросы возможно совершать и из приложения, написанного с использованием Android API, что позволит обрабатывать полученную от вики-системы API информацию в формат, удобный для непосредственного чтения любому пользователю [16].

Объектом исследования – процесс создания приложения для мобильных устройств, вики-система.

Предметом исследования является система доступа к вики-системе, а также к информации, хранящейся на ней в автономном доступе, на мобильном устройстве.

Цель исследования — обеспечение доступа к информации, хранящейся на вики-системе, с мобильного устройства с последующим доступом к сохранённым в памяти устройства статьям без прямого подключения и синхронизацией изменений в статьях между хранилищем мобильного устройства и вики-системой.

Задачами бакалаврской работы являются:

1. Рассмотрение работы текущих популярных вики-систем и установка необходимых требований для вики-системы.
2. Выбор вики-движка, отвечающего требованием и подробное ознакомление с выбранным движком.

3. Разработка алгоритма доступа к содержимому на мобильном устройстве без прямого подключения с системой синхронизации содержимого статей с последующей реализацией.

Первая глава посвящена теоретическим аспектам создания мобильного приложения для с вики-системы. В тексте главы разбираются технологии вики, вики-системы, их текущее взаимодействие с мобильными устройствами и теоретические предпосылки к разработке мобильного приложения.

Вторая глава определяет требования к вики-движку как к части системы с мобильным приложением. В главе проходит анализ существующих вики-движков, выбор лучшего вики-движка, исходя из выдвинутых требований, и подробный разбор API, предоставляемого этим движком.

Третья глава описывает процесс проектирования и разработки мобильного приложения как части системы. В тексте главы описываются алгоритмы доступа к статьям, реализация приложения на Android SDK.

Глава 1 Теоретические аспекты создания мобильного приложения для вики-системы

1.1 Предпосылки вики-системы

В современном мире знания ценятся довольно высоко, ими делятся, их собирают, классифицируют, отдают в общий доступ, чтобы каждый желающий мог получить интересующую его информацию в любое время и максимально быстро. Такую задачу издревле исполняла библиотека – огромное собрание трудов человечества: рукописи, манускрипты, книги. Но даже библиотека не всегда доступна для мгновенного использования, вмешивались человеческие факторы, факторы расстояния и т.д.

С появлением сети Интернет положение вещей в плане доступности информации существенно изменилось. Ещё в 1974 году, советский физик-теоретик Андрей Дмитриевич Сахаров, писал в своей работе «Мир через полвека»: «В перспективе, быть может, поздней, чем через 50 лет, я предполагаю создание всемирной информационной системы (ВИС), которая сделает доступным для каждого в любую минуту содержание любой книги, когда-либо и где-либо опубликованной, содержание любой статьи, получение любой справки» [11]. Как, собственно, и произошло: находящиеся в разных точках земного шара рабочие станции получали возможность «общаться» друг с другом, пересылая сигналы через глобальную сеть. Пользователь такого персонального компьютера, имеющего доступ во всемирную сеть, получал возможность поиска любой информации, которая, конечно, предварительно имелась в сети.

Однако, факт того, что информацию из сети можно получить, не означает, что её туда нельзя добавить. Поначалу библиотеки в сотрудничестве с крупнейшими университетами мира оцифровывали имеющиеся экземпляры научных трудов и литературных работ, а позже и сами пользователи объединялись ради создания электронных энциклопедий.

Для современного пользователя персонального компьютера сеть Интернет уже давно является чем-то обыденным, неким инструментом для помощи в работе, источником развлечений и общения. Поэтому, доступ к информации просто не должен представлять собой большой и трудоёмкий процесс, так как необходимые знания классифицировались и выкладывались в открытый доступ ещё со времён становления сети Интернет как глобальной сети.

Однако всю имеющуюся информацию стоит каким-то образом хранить, причём предоставляя быстрый, удобный и эффективный поиск с выдачей наиболее подходящих результатов, к тому же выдавать результат как удобно структурированный источник информации, ничем не уступающий печатным изданиям, например, энциклопедиям.

1.2 Вики-система

Основой для системы является вики (wiki) – гипертекстовая среда для сбора и структурирования письменных сведений, структуру и содержимое которой пользователи могут самостоятельно изменять с помощью инструментов, предоставляемых самим сайтом. Первое упоминание такого термина приходится на 1995 год, когда его употребили в значении «простейшей онлайн-базы данных, которая может функционировать» [11]. С тех пор система вики сильно разрослась, сформировала свои непосредственные признаки и концепцию, по которой:

- любой пользователь имеет возможность редактировать и создавать статьи с помощью предоставляемых инструментов, работающих в стандартной комплектации браузера (без расширений);
- любая страница может быть связана с любой другой страницей с помощью системы ссылок, при этом отсутствующие звенья должны быть наглядно отображены (к примеру, ссылка на несуществующую статью)

- сама вики не должна быть тщательно изготовленным сайтом для случайных посетителей – вики должна меняться путём созидания и изменения созданного самими пользователями.

Вики стала довольно популярной системой и в силу того, насколько просто происходит процесс добавления новых статей и противостояние вандализму. Благодаря отказу от постоянного патрулирования каждого изменения страницы, вандализм имеет место быть, однако система ревизий позволяет очень быстро вернуться к необходимой версии статьи, ещё не тронутой вандалами.

Позже этому принципу вики стали следовать всё больше и больше обществ, и теперь этот принцип используется в том числе и в корпоративной среде. Особенно важно в таком случае иметь доступ к корпоративной базе информации как можно быстрее и с какой-угодно точки планеты. К сожалению, даже с появлением беспроводных сетей и мощных, и в то же время компактных, переносных компьютеров, получить необходимую информацию довольно проблематично: покрытие сети не везде реализовано на должном уровне, а компьютеры не настолько компактны, чтобы пользоваться ими на ходу. Однако, с появлением мобильных телефонов нового поколения, так называемых «смартфонов», которые соединяли в себе функции обычного мобильного телефона и функции, которые ранее были доступны только на персональном компьютере, большое количество операции, ранее проводимых исключительно на рабочих станциях, было переведено для работы на мобильных устройствах. Даже если фактор неудобства использования убирается из поля зрения, остаётся факт, что доступ к сети не всегда имеется.

Вики с самого начала позиционировалась как система для работы с контентом через веб-инструменты времени Web 2.0, поэтому для изменения текста статей преимущественно используются специальные визуальные редакторы What You See Is What You Get (WYSIWYG), позволяющие с помощью предоставленного инструментария редактировать статью, приводя её внешний вид к желаемому с помощью непосредственного визуального

редактирования, причём внутренние средства сами преобразовывают конечный визуальный вариант содержимого в поддерживаемый данной вики-системой синтаксис [7].

На рисунке 1.1 представлена одна из реализаций WYSIWYG-редактора.

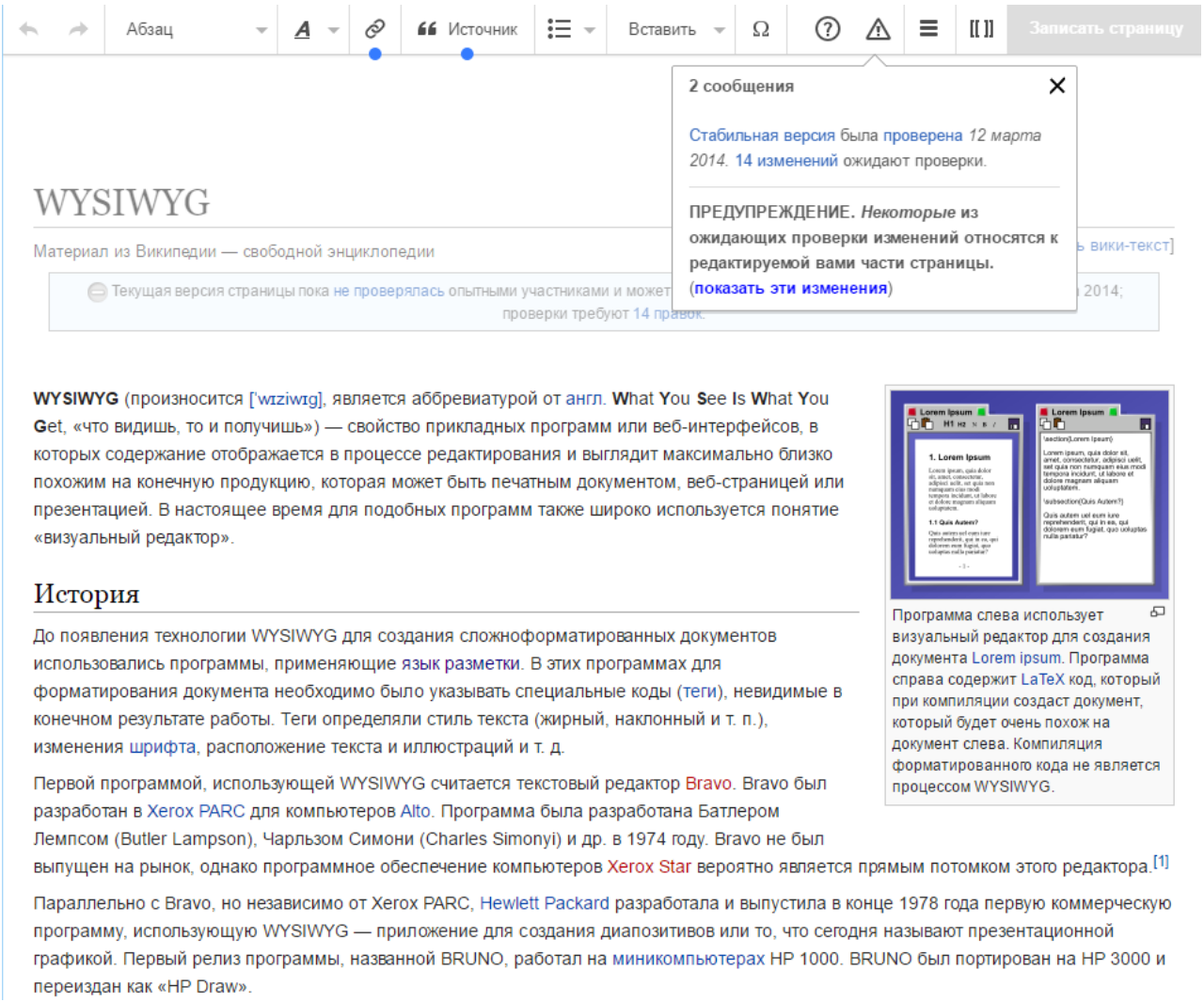


Рисунок 1.1 – WYSIWYG-редактор с содержимым статьи “WYSIWYG” в Wikipedia

В современных системах внутри содержимого также используются и элементы HTML-вёрстки, в основном для расширения базового функционала редактора и для упрощения конечной вёрстки содержимого статьи. Однако не все разработчики вики-движков считают это необходимой мерой и зачастую даже отказываются от такого использования в угоду безопасности вики-системы.

Навигация по вики-системе происходит с помощью системы гипертекста, связывающей огромное количество страниц друг с другом. Порой таких ссылок приходится очень много на одну статью, из-за чего ориентироваться в непосредственном содержимом статьи человеку становится немного сложно, однако вики-системы сконфигурированы таким образом, чтобы поддерживать большое количество таких ссылок, как находящихся в непосредственном содержимом рассматриваемой статьи, так и ссылки, указывающие на рассматриваемую статью из других источников, в том числе и переадресации [3].

В настоящее время вики-системы обретают свои собственные системы поиска контента, однако до сих пор популярно решение отдавать возможность поиска сторонним поисковым движкам, к примеру, Google Поиск. Для собственных реализаций системы поиска большим фактором является эффективность работы вики-системы с базой данных или другим хранилищем, необходимым для развёртывания вики-системы на сервере.

Система ревизий, показанная на рисунке 1.2, позволяет пользователям отслеживать изменения содержимого статей на протяжении больших промежутков времени. Система выводит наглядную таблицу с содержимым статей двух разных ревизий и выделяет все отличия между ними [7, 10].

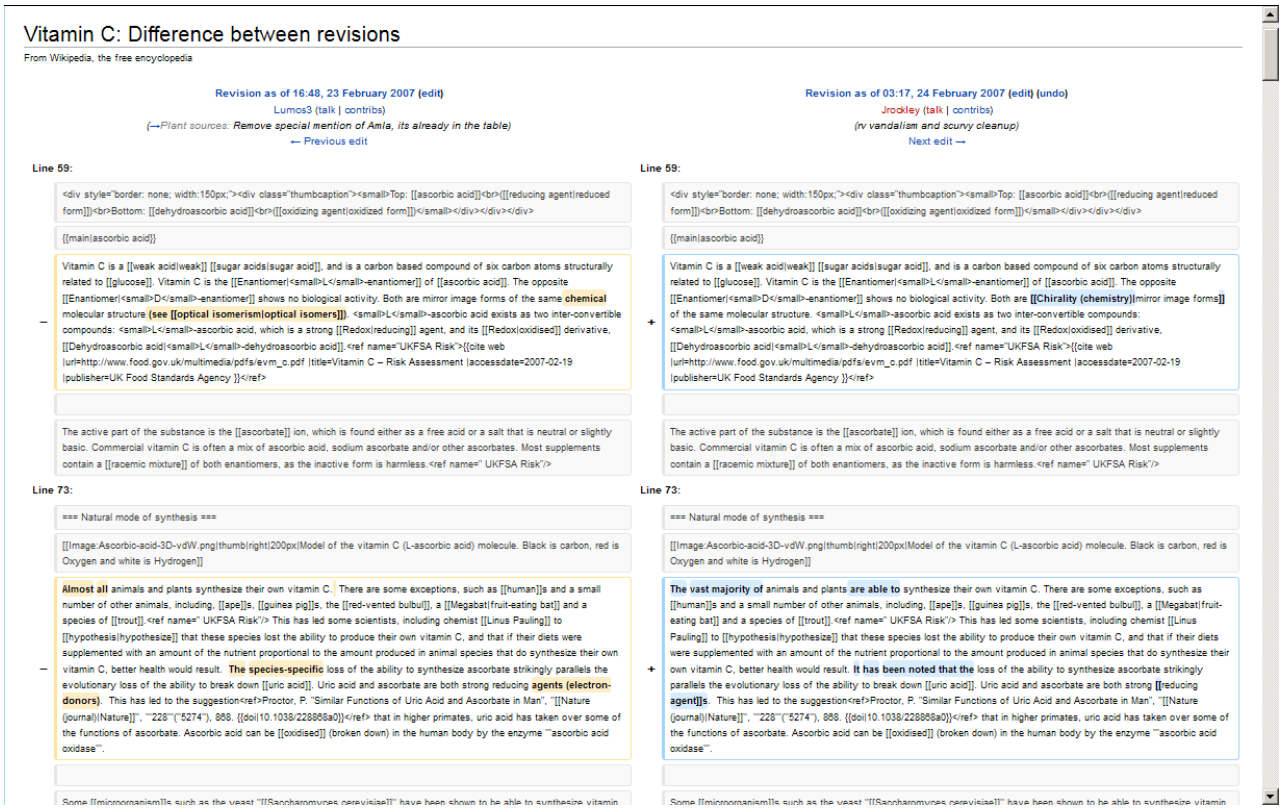


Рисунок 1.2 – Система ревизий выявляет изменения между двумя ревизиями

При возникающих конфликтах версий система отдаёт решение ответственным лицам (администраторы, модераторы, кураторы, проверенные пользователи) для поддержания информативности статьи и для уменьшения шанса вандализма.

1.3 Вики-система и мобильные устройства

Предпосылкой к созданию мобильного приложения, в первую очередь, является растущая популярность мобильных устройств среди населения. Технологии становятся доступнее и дешевле, появляются фантастические «смартфоны» по цене в 4$, производители девайсов повышают планку качества сборки и преподносят технические улучшения с каждой новой моделью. Смартфоны и планшеты выглядят более привлекательными в потребительском плане, благодаря довольно низкой цене и большому спектру функций, которые эти устройства предлагают без предварительной настройки.

Основной проблемой использования оригинальной версии вики, которая представляет из себя веб-интерфейс, работающий в браузере, содержащий в

себе стандартные средства редактирования для этой вики, является частая несовместимость с мобильными устройствами [15]. Зачастую администраторы вики не предусматривают возможности поддержки специального мобильного интерфейса вики с заточенными под управление касаниями инструментами редактирования. В некоторых случаях, мобильный интерфейс предоставлен в пакете установки по умолчанию, но и он не всегда способен отзывчиво работать в браузерах мобильных устройств.

Но даже если такая поддержка осуществляется, встаёт вопрос о доступе к статьям без подключения к сети. Встроенная в браузеры функция «Сохранить страницу» зачастую сохраняет страницу вместе с абсолютно лишними элементами, которые могут занимать большой объём в памяти устройства, к примеру Flash-элементы, большие по размеру GIF-анимации, большое количество сопровождающих элементов интерфейса, направленных на создание визуальных эффектов, и т.д., а также сохранять страницу непосредственно в PDF-формат, где конвертация в большинстве случаев происходит с несоблюдением разметки на странице, превращая статью в нечитабельную.

Однако, у браузерного решения есть и свои достоинства: для использования мобильной версии вики пользователю нет необходимости устанавливать отдельное приложение на своё устройство; браузерная версия заточена не под отдельную платформу на устройстве, а больше под браузер, в большинстве случаев с учётом стандартного браузера, идущего в комплекте с системой по умолчанию. В новых версиях мобильных браузеров появилась возможность выносить на домашние экраны подобие «ярлыков» для страниц в сети Интернет, причём эти ярлыки стараются маскироваться под отдельные приложения [20].

К сожалению, чтобы поддерживать все функции вики-движка, браузер довольно сильно нагружает центральный процессор «смартфона» и занимает много памяти в ОЗУ, что несомненно сказывается на быстродействии устройства и, в некоторых случаях, делает работу с вики просто невозможной,

вплоть до полного сбоя системы устройства. Приложение, в данном случае, существенно снижает нагрузки на систему, и, к тому же, помогает сэкономить мобильный трафик, не загружая визуальные стили, лишние элементы интерфейса и дополнительные скрипты для редактирования статей, так как необходимый функционал уже включён в приложение и не требует дополнительной загрузки. Приложение запрашивает через API непосредственное содержимое статьи, а не страницы, на которой статья выводится в браузере, поэтому весь код документа остаётся редактируемым.

Необходимость доступа к статьям может возникнуть даже при нахождении вне сети, когда мобильный браузер не сможет никаким образом обратиться к вики-системе и будет ограничен просмотром уже сохранённых в нём статей. Естественно, сохранить полностью всю вики-систему на устройстве с довольно ограниченным количеством памяти невозможно, однако приложение автоматически кеширует все статьи, в которых пользователь когда-либо делал правки, статьи, которые он поместил в «отслеживаемые» и, конечно, статьи по желанию, и делает их доступными для чтения в любой момент, даже если активное подключение к сети отсутствует. При появлении активного соединения приложение автоматически запускает процесс синхронизации сохранённых статей с их актуальными версиями на сервере.

Сохранённые в приложении статьи могут быть доступны и для редактирования. При редактировании, статья помечается как изменённая, и, при синхронизации, приложение оповестит пользователя о том, что версия статьи, сохранённой на устройстве, отличается от версии статьи, сохранённой на сервере. В зависимости от отличий могут быть предложены следующие версии:

- если версия статьи на сервере не изменилась с момента последней синхронизации, а на устройстве статья была модифицирована, версия с устройства автоматически, или по указанию пользователя, отправляется на сервер:

- если версия статьи на сервере выше или равна модифицированной на устройстве, пользователю предлагается рассмотреть изменения и изменить сохранённую на устройстве статью, чтобы избежать ошибок при слиянии версий;
- если при этом и в статье на устройстве и в статье на сервере изменялась одна и та же строчка, выводится предупреждение о конфликте версий и сравнение изменений в сравнении с последней синхронизированной версией.

Ещё одно интересное свойство приложений – режим фоновой работы. В нём приложение сможет отслеживать появление активного соединения, попробовать соединиться с вики и осуществить процесс синхронизации. Предупреждения и события из приложения могут быть показаны в качестве Push-уведомления, помогая пользователю отслеживать изменения в статьях, в которых он заинтересован [4, 12].

При синхронизации теоретически могут возникнуть конфликты ревизий, о чём мобильное приложение тоже сможет сообщить с помощью Push-уведомлений, доступных на большинстве устройств с различными операционными системами, и в настоящее время поддерживаемые даже популярными браузерами в качестве desktop-уведомлений [4, 14].

1.4 Формирование требований к разрабатываемому приложению

Для наиболее эффективной работы с информацией, хранящейся в вики-системе, на мобильном устройстве следует обеспечить:

- автономность, благодаря сохранению необходимых статей в хранилище мобильного устройства;
- быстродействие путём уменьшения входной нагрузки на процессор мобильного устройства;

- возможность соединения с сетью Интернет для загрузки и синхронизации статей
- соответствие интерфейса мобильным стандартам для максимального удобства работы с информацией;

Мобильные устройства были созданы с идеей портативности в концепции, поэтому такие устройства легко переносить в карманах одежды, что подразумевает перемещение пользователя, в том числе и на большие расстояния. Использование беспроводных соединений в современных устройствах позволяет выходить в сеть Интернет даже с отдалённых точек земного шара при условии наличия таких соединений. Однако не вся поверхность Земли имеет покрытие мобильными и/или беспроводными сетями, а значит есть возможность не обнаружить сигнала сети. Автономность приложения подразумевает использование функций приложения даже без прямого подключения к сети, что определённо поддерживает концепцию мобильности устройства.

Обеспечение быстродействия мобильного приложения является одной из самых приоритетных задач в мобильном программировании. Современные мобильные устройства хоть и становятся всё производительнее, большая часть ресурсов устройства всё равно направлена на поддержание самых необходимых функций: мобильная связь, беспроводные соединения, работа самой системы, прорисовка изображения на экране, работа системы глобального позиционирования (GPS, ГЛОНАСС) и т.д. Одна из основных черт мобильных устройств – сравнительно малые размеры относительно своей функциональности, что позволяет работать с приложениями на ходу, отнимая сравнительно малое количество времени относительно использования персонального компьютера. Исходя из самой концепции мобильного устройства, следует обеспечить быстродействие приложения для ускорения получения необходимой информации на экране устройства.

Использование мобильного устройства эволюционировало с течением времени: начиная от больших кнопок на пятикилограммовых телефонах компании Motorola в 80-х, появления простейших дисплеев для отображения информации, до практически полной замены передней панели мобильных устройств на экран с поддержкой касаний, размером во всю фронтальную панель. Даже клавиатура была запрограммирована для появления исключительно на экране устройства и оперируется касаниями пальцев по экрану в местах, относительно отображению необходимых виртуальных клавиш на экране. Дизайн новых приложений требовал переработки, чтобы отвечать концепции управления интерфейсом с помощью нажатий пальцами по экрану, учитывая относительно небольшой размер поверхности для нажатий и отображений интерфейса.

Исходя из поставленных проблем доступа к вики-системе с мобильных устройств и поддержки мобильным браузером высоких нагрузок, оказываемых стандартной мобильной версией вики-системы, к приложению выдвинуты следующие требования:

- использование API, предоставляемого вики-движком;
- использование внутреннего хранилища устройства для сохранения кэшированных статей и настроек приложения
- использование сетевых возможностей устройства для загрузки и синхронизации загруженных статей;
- использование функций и библиотек Android API;
- построение интерфейса согласно дизайнерским требованиям Material Design от компании Google.

На рисунке 1.3 представлена Use Case диаграмма работы мобильного приложения с вики-системой.

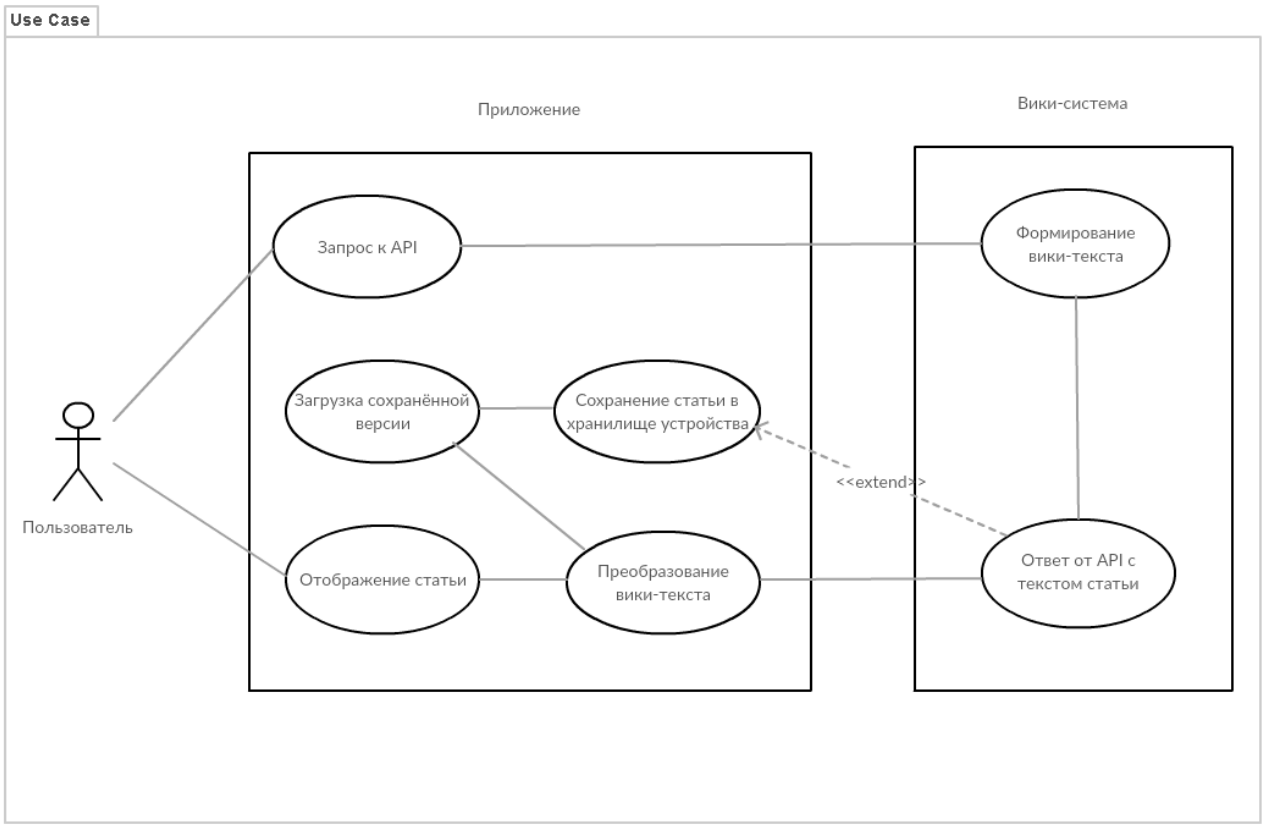


Рисунок 1.3 – Use Case диаграмма работы приложения с вики-системой

Глава 2 Анализ и применение технологий вики-систем

2.1 Выбор вики-системы для разработки мобильного приложения

Вики-движок в своей реализации является неким подобием CMS (Content Management System – система управления контентом), поэтому для развёртывания вики-системы на оборудовании, существуют минимальные требования, которые следует удовлетворить для номинальной работы вики-системы:

- база данных, требуемая вики-движком для функционирования, или другая система управления данными на сервере;
- наличие веб-сервера;
- необходимых надстроек и/или плагинов для поддержки функционала вики-системы.

У разных вики-движков могут быть дополнительные требования для функционирования, однако в большинстве случаев вышеперечисленный список требований присутствует как обязательный [18].

Различные вики-движки реализованы с помощью большого спектра языков программирования, начиная от PHP, заканчивая языком программирования Squeak, диалектом Smalltalk.

Развёртывание вики-системы необходимо проводить на поддерживаемой системе, либо использовать SaaS (англ. Software as a Service – программное обеспечение как услуга), предоставляемое компанией-разработчиком и развёртываемое на серверах поставщика услуги.

Немаловажным фактором для юридических лиц, образовательных и государственных учреждений, является наличие необходимой лицензии для использования вики-системы, и зачастую именно это является основанием для выбора той или иной системы для использования.

Для того, чтобы полная система, состоящая из вики-системы и мобильного приложения, функционировала необходимым способом, обе части системы должны отвечать некоторым требованиям.

Требования к вики-системе:

- вики-движок должен поддерживать самые распространённые веб-серверы, к примеру, Apache HTTP Server версии не ниже 2.0;
- вики-движок должен устанавливать стабильное соединение с хранилищем данных, к примеру, с базой данных MySQL;
- возможность развёртывания на популярных операционных системах;
- наличие свободной лицензии;
- развёрнутая вики-система должна поддерживать либо web-API, либо REST API для взаимодействия с внешним приложением;
- развёрнутая с помощью вики-движка вики-система должна быть доступна в глобальной сети Интернет либо из локальной сети;
- статьи, представленные в системе, должны использовать систему ревизий для обеспечения корректной синхронизации содержимого, получаемого вики-системой с хранилища мобильного приложения.

Для работы вики-системы в связке с мобильным приложением этого достаточно, но сфера работы вики-системы мобильными устройствами не ограничивается, поэтому вступают требования и для стандартного интерфейса:

- What You See Is What You Get (WYSIWYG) редактор для удобства построения статьи наиболее приближено к задуманному внешнему виду;
- импорт и экспорт статей и настроек в случае сбоя и повторного развёртывания вики-системы;
- использование HTML-кода в теле статьи для расширение функционала разметки;
- наличие настраиваемого пользовательского интерфейса;

- наличие новостной ленты для отслеживания свежих правок;
- наличие системы защиты от спама;
- осуществление контроля доступа к статьям;
- поддержка альтернативных вики-синтаксисов для переноса статей в вики-систему, развёрнутой с помощью вики-движка, отличного от оригинального;
- поддержка загрузки файлов.

Для определения необходимого для работы вики-движка, необходимо рассмотреть популярные движки, в особенности их требования для установки и развёртывания вики-системы и их особенности.

В таблице 2.1, 2.2 и 2.3 приведены данные по популярным вики-движках [7], широко использующихся в вики-системах различной степени направленности. У каждого движка перечислены его требования к установке и особенности, по которым и будет определён самый подходящий к требованиям движок.

Таблица 2.1 – Характеристики вики-систем

Название	**Система**	**Лицензия**	**Написан на языке**	**Используемая система хранения данных**
Confluence	Microsoft Windows, Linux	проприетарная	Java, Java EE	DB2, MS SQL Server, MySQL, Oracle, или PostgreSQL
DokuWiki	Linux	GNU GPL v2	PHP	Файловая система
eXo Platform	Unix, Linux, Hosted	LGPL	Java	PostgreSQL, MySQL, Oracle, Apache Derby, HSQLDB
Foswiki	Unix, Linux	GNU GPL	Perl	Flat-file, RCS

IBM Connections	Linux, Unix	проприетарная	Java/Java EE	DB2, MS SQL Server или Oracle
Название	**Система**	**Лицензия**	**Написан на языке**	**Используемая система хранения данных**
MediaWiki	Linux, Unix, Windows	GNU GPL v2	PHP	MySQL, PostgreSQL, SQLite, и другие базы данных
Nuclino	кроссплатформа	проприетарная	Javascript	OrientDB
PhpWiki	Linux, Unix, Windows	GNU GPL	PHP	Berkeley DB, MySQL, PostgreSQL, Microsoft SQL Server, Oracle 8, Firebird
Tiki Wiki CMS Groupware	Любая ОС с J2VM	LGPL	PHP	MySQL

Таблица 2.2 – Особенности вики-движков

Вики-движок	**Загрузка файлов**	**Защита от спама**	**Контроль доступа к статьям**	**Использование HTML в коде статьи**	**Кастомизируемый пользовательский интерфейс**
Confluence	Да	Да, с помощью CAPTCHA	Да	опционально	Да, шаблоны и темы + CSS
DokuWiki	Да	Да, чёрный список	Да, опционально	Да, опционально	Да, шаблоны, CSS, PHP с помощью PHP API
eXo Platform	Да	Нет	Да	Да	Да, CSS
Foswiki	Да	Да	Да	Да	Да, шаблоны, темы, пользовательский CSS
IBM Connections	Да	Да, в приватном режиме	Нет	Да	Да, шаблоны, CSS
MediaWiki	Да	Да, блокирование IP, адресов, CAPTCHA	Частично	Да	Частично, шаблоны
PhpWiki	Да	Да, с помощью модуля	Да	С плагином	Не документировано
Tiki Wiki CMS Groupware	Да	Да, CAPTCHA для регистрации	Да	Да	Темы, пользовательский CSS

Таблица 2.3 – Дополнительные особенности вики-движков

Вики-движок	**WYSIWYG редактор**	**Новостная лента**	**Импорт, экспорт**	**Расширяемость**	**Альтернативные вики-синтаксисы**
Confluence	Да	Да, RSS	Нет	Нет	Нет
DokuWiki	Да, плагин	Да, RSS/Atom	Да, XML, plain text, LaTeX	Да, плагины	Да
eXo Platform	Да	Нет	Нет	Да, макросы	Нет
Foswiki	Да, предустановленый плагин	Да, RSS/Atom, с поисковой строкой	Нет	Да, API для плагинов	Да
IBM Connections	Да	Да, RSS/Atom	Нет	Да	Да
MediaWiki	Да, с расширениями	Да, RSS/Atom	Да, web API, оболочка	Да	Да
PhpWiki	Нет	Да, RSS/Atom	Нет	Да, плагины	Нет
Tiki Wiki CMS	Да	Да, RSS/Atom/PDF	Нет	Плагины, модули	Нет

Groupware					

Исходя из выдвинутых требований и благодаря своей лёгкости развёртки, открытой лицензии, относительно высокой доступности, количества функций, доступных с начала работы после установки, вики-движок MediaWiki является самым подходящим движком для использования для развёртки вики-системы для работы с мобильным приложением.

2.2 Основные особенности вики-движка MediaWiki

Популярность движка MediaWiki обоснована гибкостью развёрнутой вики-системы, относительная лёгкость установки на основных видах операционных систем, возможностью глубокой конфигурации, относительно как контента, так и внешнего вида (если используется стандартная браузерная версия вики-системы с использованием пользовательских стилей, оформления и тем), дополнительные возможности импорта и экспорта статей и загруженных файлов.

С технической точки зрения вики-движок MediaWiki реализован с упором на масштабируемость, что позволяет системе оперировать с большим количеством информации, хранящейся с использованием MySQL, при этом выдерживая миллионы обращений в день [6].

Самые известные вики-системы, реализованные с помощью движка MediaWiki – Википедия, Викимедия, Wikia.

Благодаря лёгкому синтаксису и API для модулей, вики-движок MediaWiki получил за время существования поддержку огромного количества плагинов и модификаций, добавляющих в вики-систему много дополнительных возможностей, которые не предоставляются вики-системой с момента непосредственной установки после конфигурации. Однако со временем базовая конфигурация вики-движка стала включать в себя всё больше функций, которые раньше были доступны только с установкой дополнительных плагинов или модификаций [5, 6, 8]. К примеру, поддержка TeX и LaTeX для формул значительно увеличивает значимость вики-системы. MediaWiki первоначально был написан для свободной энциклопедии Википедии, но со временем

значительно преобразился и представляет собой мощную базу для построения вики-проектов. MediaWiki является свободной программой и распространяется на условиях Общественной лицензии GNU. Использование в крупнейшей библиотеке сети Интернет зарекомендовало движок и сейчас он используется в довольно широком кругу разнообразных проектов – от личных, до корпоративных и государственных.

MediaWiki предоставляет интерфейс работы с базой страниц, разграничение прав доступа к администрированию системы, возможность обработки текста как в собственном формате вики-текст, так и в форматах HTML и TeX (для формул), возможность загрузки изображений и других файлов, а также другие возможности.

MediaWiki предусмотрен специальный интерфейс прикладного программирования, обеспечивающий прямой высокоуровневый доступ к информации из баз данных. Клиентские программы могут использовать API для авторизации, получения данных и отправки изменений. Именно через этот интерфейс работают скрипты на веб-странице при обычной работе с Википедией. Доступ к API может быть получен любым образом, в том числе и через front-end и back-end программы. Главным удобством API является то, что не имеет никакого значения язык программирования, на котором будет написано клиентское приложение, поскольку все запросы обрабатываются по протоколу HTTP, а ответ получается в удобном для разработчика формате: XML, сериализованном PHP, YAML или JSON. Именно такой ответ будет использовать мобильное приложение [5].

Вики-движок скрепляет все статьи в единое целое благодаря системе ссылок, которые интуитивно определяются на уровне разметки, которая здесь тоже специальная.

Вики-разметка – язык разметки, который используется для оформления текста на веб-сайтах (как правило, относящихся к классу вики-проектов) и позволяет упростить доступ к возможностям языка HTML. Страницы, оформленные с применением вики-разметки, предварительно преобразуются в

HTML для просмотра в веб-браузере, преобразованием же занимается как раз вики-движок.

Вики-разметка создавалась с видением разметки содержимого внутри одной статьи, поэтому у неё лимитированный относительно HTML функционал. Однако, в этом есть и преимущество – размеченные таким образом страниц занимают меньше места в хранилище, легче версионируются (по большей части, визуально легко заметить различия) и позволяют использовать перекрёстные ссылки [8].

Для каждого вики-движка используется своя разметка, хоть они и не отличаются друг от друга на первый взгляд, на самом деле для каждого движка происходит «оптимизация» разметки для более точного и быстрого распознавания разметки движком.

2.3 MediaWiki API

MediaWiki API предоставляет удобный доступ к функциям, данным и метаданным вики-системы по HTTP, с помощью обращения к api.php.

Получаемые данные можно выводить в различных форматах, указав необходимый формат в параметрах запроса. Среди поддерживаемых форматов присутствуют [5]:

- JSON;
- PHP;
- XML.

До июня 2014 года вики-движок использовал и такие форматы, как YAML, plaintext, dump, WDDX и dbg, но из-за непопулярности среди разработчиков вики-движка и использующих API в своих приложениях, от этих форматов отказались в пользу поддержки более популярных форматов.

Конечной точкой (endpoint) служит api.php, который в случае развёртывания вики-системы находится по URL: http://unrael.ru/wiki/api.php

Все запросы передаются в качестве параметров в URL конечной точки, специфицируя информацию, получаемую в ответе на запрос. К примеру, запрос на получение информации с главной страницы вики-системы, будет выглядеть следующим образом:

http://unrael.ru/wiki/api.php?action=query&titles=Заглавная_страница&prop=revisions&utf8=1&rvprop=content&format=json

Параметр action определяет действие, которое должно будет совершено при обработке данного обращения к вики-системе. В данном случае, этот параметр action=query, что означает запрос некоторой информации.

Ранее такие запросы отправлялись сразу на конечную точку query.php, но от такого решения было принято отказаться в пользу унифицированной поддержки API.

Оставшаяся часть URL содержит параметры для действия «query».

В конструкции “titles=Заглавная_страница” сообщается о том, что пользователем запрашивается информация о вики-странице под названием "Заглавная страница". Если требуется запросить несколько страниц, они указываются в одном запросе для оптимизации ресурсов сети и сервера: titles=PageA|PageB|PageC.

Параметр prop=revisions сообщает веб-службе API о необходимости выдачи информации об определённой версии страницы. Если информация о ревизии не была указана, API возвращает информацию о последней ревизии запрошенной страницы на текущий момент.

Для корректного отображения кириллицы возможна передача параметра “utf8=1” для изменения кодировки возвращаемой информации в соответствующую.

Параметр rvprop сообщает веб-службе API возврате последней ревизии страницы. Существует возможность получения последней версии содержимого страницы и имени пользователя, совершившего правку, с помощью rvprop=content|user.

Параметр format сообщает движку, в каком формате требуется вывод ответа. На данный момент времени самыми используемыми форматами являются JSON, сериализованный PHP и XML. В указанном выше запросе конструкция “format=json” указывает на использовании JSON как формата ожидаемого ответа.

С помощью таких запросов, MediaWiki предоставляет возможность работать с контентом вики-системы из любого программного обеспечения, поддерживающего отправку запросов через HTTP [19, 21]. Гибкий механизм запросов, множество форматов возвращаемой информации позволяют максимально использовать MediaWiki API даже на современных мобильных устройствах, как в браузере, с помощью обычных запросов с использованием адресной строки, так и в полноразмерном нативном приложении, использующего преимущества системы для генерации, отправки запроса и парсинга полученной информации с помощью библиотек, непосредственно входящих в систему.

Так же Wikimedia предоставляет REST API как надстройку над существующим API, но реализовано решение не полностью, поэтому доступно только в бета-версии [6].

2.4 Использование MediaWiki API в приложении

Самое основное использование MediaWiki API в программном обеспечении состоит в генерации и отправке запросов к конечной точке и обработке полученной информации. Поскольку MediaWiki API использует доступ через HTTP, запрашивать информацию можно с помощью уже реализованных решений и/или модулей для работы с HTTP-запросами.

Полученный ответ необходимо пропустить через парсер, для получения структурированного объекта с содержимым, полученным из ответа необходимого типа [2, 22].

Позже, полученную с помощью парсера информацию, можно выводить на экран, применяя необходимую разметку со стороны мобильной системы,

эквивалентную визуализации вики-разметки в браузерной версии вики-системы.

С использованием инструментов разработки для соответствующей мобильной операционной системы становится возможным использование разметки текста, исходя из стандартизованных системных стилей, а также с помощью создания собственных стилей оформления для необходимой разметки [16]. С помощью кастомизируемых стилей оформления становится возможным построение структуры статьи, наиболее приближенной к оформлению этой же статьи в основной браузерной версии, генерируемой вики-движком.

При запросе к MediaWiki API разработчик получает не только текст статьи, но и сопроводительную служебную информацию, такую как текущий номер ревизии, который служит одним из необходимых параметров для синхронизации содержимого статьи между версией, хранимой на мобильном устройстве, и версией, находящейся непосредственно на сервере с развёрнутой вики-системой.

Глава 3 Проектирование и разработка мобильного приложения для работы с вики-системой

3.1 Доступ к статьям вики-системы

Чтобы позволить системе функционировать, как было задумано, необходимо обеспечить первоначальное подключение к вики-системе. Без получения данных с сервера приложение не будет в состоянии работать с вики-системой и сможет функционировать только с сохранёнными статьями.

Существует два режима доступа к информации, доступной на вики-системе:

- стандартный доступ к серверу, на котором развёрнута вики-система, с помощью подключения к сети Интернет. В этом режиме пользователь получает самую актуальную ревизию интересующей его статьи, но требует поддерживаемого подключения.
- автономный доступ к статьям, сохранённым в хранилище мобильного устройства посредством кеширования статей во время стандартного доступа. Пользователь получает доступ к интересующей его статье при любом состоянии сети, однако не может получить новую ревизию статьи без синхронизации.

Для обеспечения доступа к статье в автономном режиме требуется хотя бы раз подключиться к вики-системе на сервере напрямую, обратиться к интересующей статье, совершить действие, за которым следует автоматическое кеширование статьи (изменение статьи, добавление в список отслеживаемых, непосредственное использование функции кеширования в мобильном приложении) и осуществить загрузку содержимого статьи в хранилище.

На рисунке 3.1 изображена схема первоначального доступа к статье, находящейся на развёрнутой вики-системе.

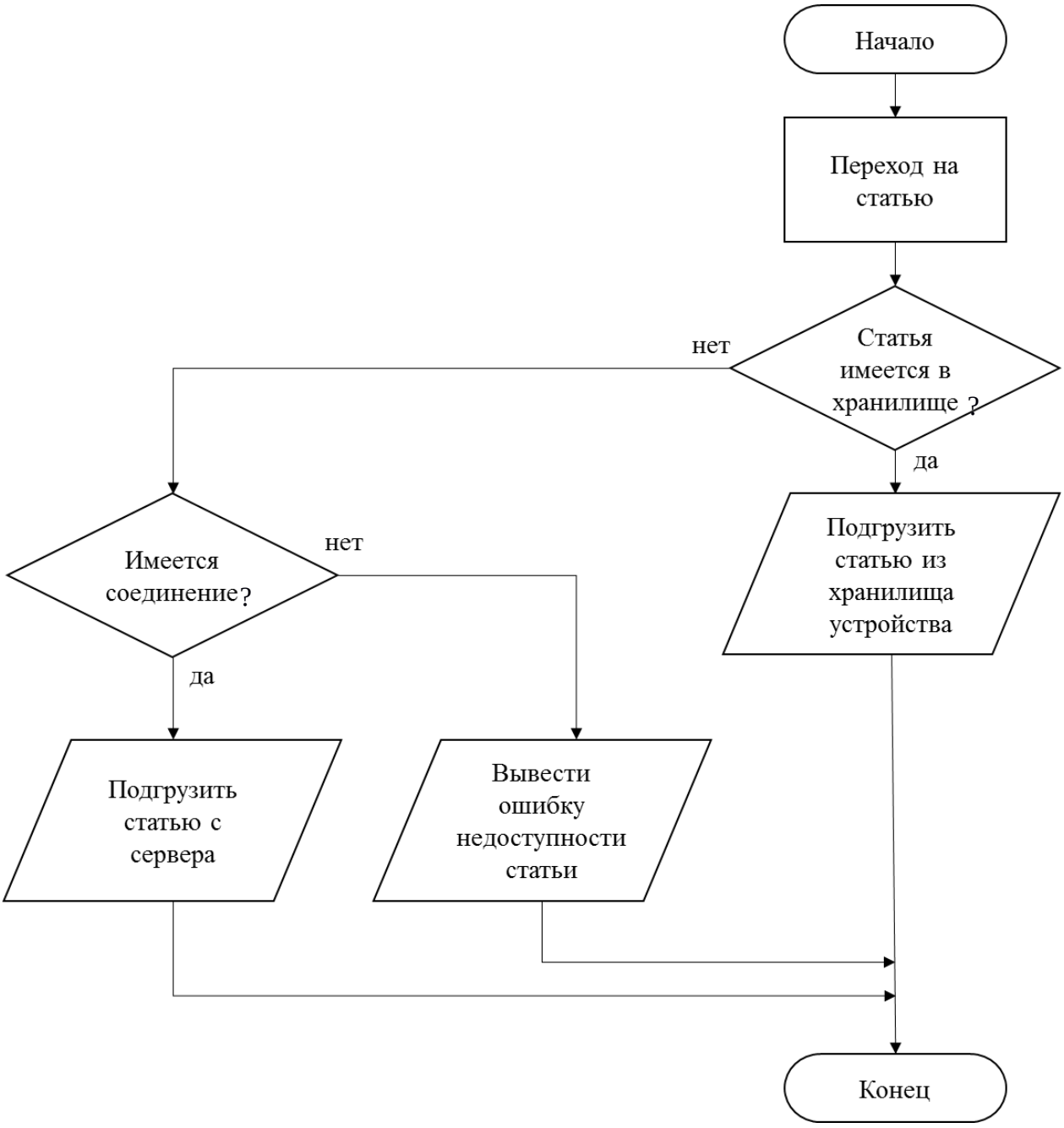


Рисунок 3.1 – Схема первоначальной подгрузки статьи

При переходе на статью в первую очередь осуществляется проверка наличия статьи в хранилище с последующей возможностью подключения к серверу для первоначальной загрузки статьи, так как на данный момент содержимое статьи может быть не кэшировано на внутреннем хранилище мобильного приложения [13]. Если соединение с сервером, на котором развёрнута вики-система, успешно, осуществляется подгрузка статьи

непосредственно с сервера с помощью MediaWiki API по названию статьи; в обратном случае пользователю выводится сообщение о невозможности подключения к серверу.

На рисунке 3.2 представлена блок-схема алгоритма подгрузки уже сохранённой статьи в хранилище устройства, с условием возможности подгрузки статьи с сервера с развёрнутой вики-системой.

Условием для вступления этого алгоритма в действие является условие о наличии статьи в хранилище из схемы на рисунке 3.1. Если соединение с сервером не установлено, подгрузка будет осуществляться непосредственно из внутреннего хранилища мобильного устройства. Если соединение было установлено, то программа сравнивает ревизии статьи, находящейся на сервере, и статьи, сохранённой в хранилище устройства. Если ревизии совпадают, то подгрузка статьи с сервера необязательна, и для экономии трафика статья подгружается с хранилища устройства.

Если ревизии различаются, то программа проверяет на наличие изменений в сохранённой на устройстве статье. Если изменений не обнаружено (обозначается флагом в сохранённой статье), то производится подгрузка статьи с сервера с развёрнутой вики-системой; в противном случае пользователю выдаёт предупреждение о конфликте ревизий, которое предлагает пользователю вручную внести необходимые изменения в сохранённой на устройстве статье, чтобы устранить состоявшийся конфликт.

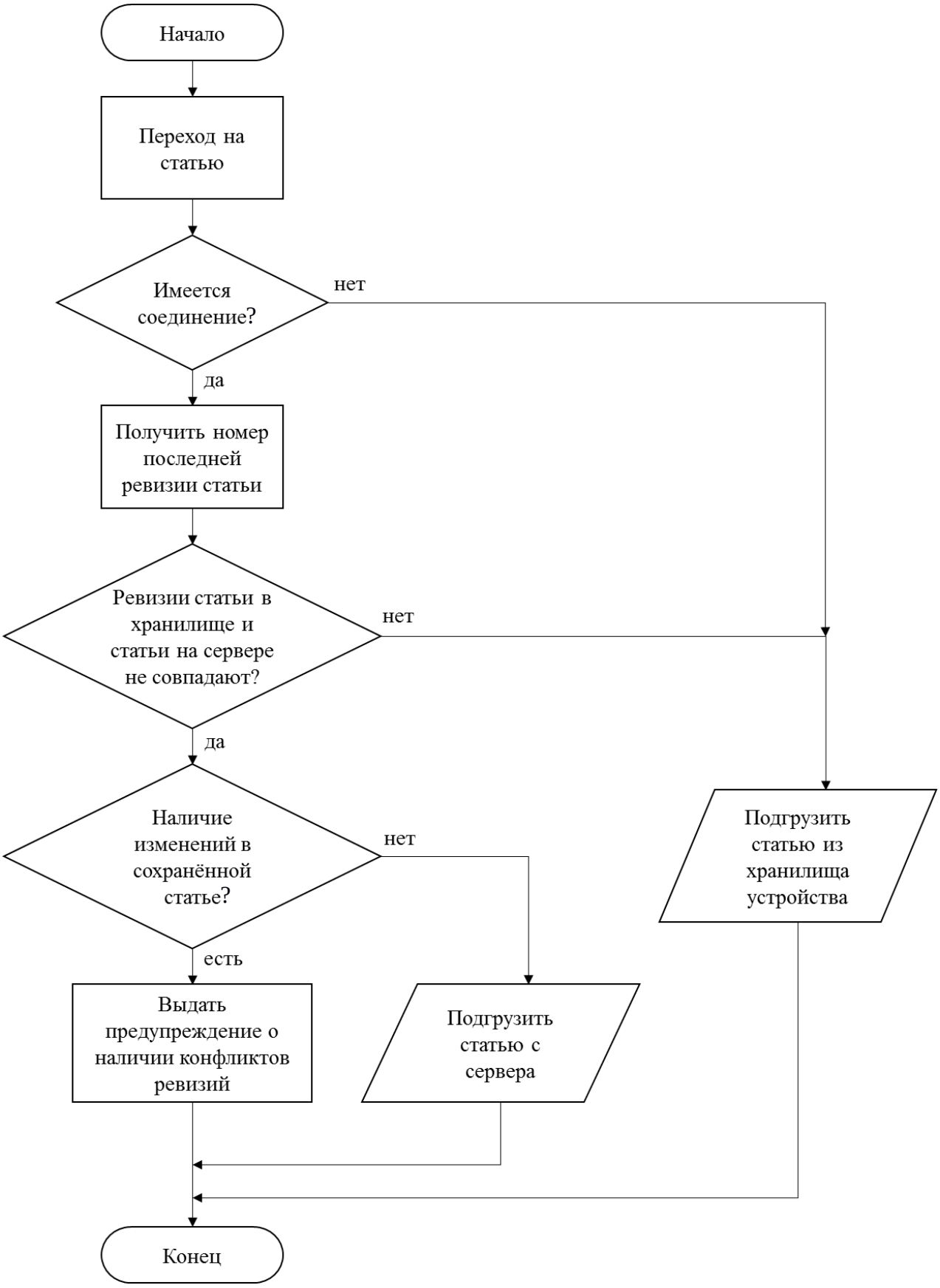


Рисунок 3.2 – Схема подгрузки содержимого сохранённой статьи

При возникновении конфликтов ревизий, пользователь получает уведомление, при взаимодействии с которым происходит навигация на страницу статьи, на которой отображается информация о конфликте содержимого между двумя ревизиями. При этом принятие решений об устранении конфликтов остаётся за пользователем, который может либо принять за эталонную одну из существующих ревизий (сохранённую статью в хранилище мобильного устройства и статью, находящуюся на сервере), либо совершить редактирование сохранённой статьи таки образом, чтобы получить новую ревизию с необходимыми изменениями.

3.2 Подгрузка статьи с помощью MediaWiki API

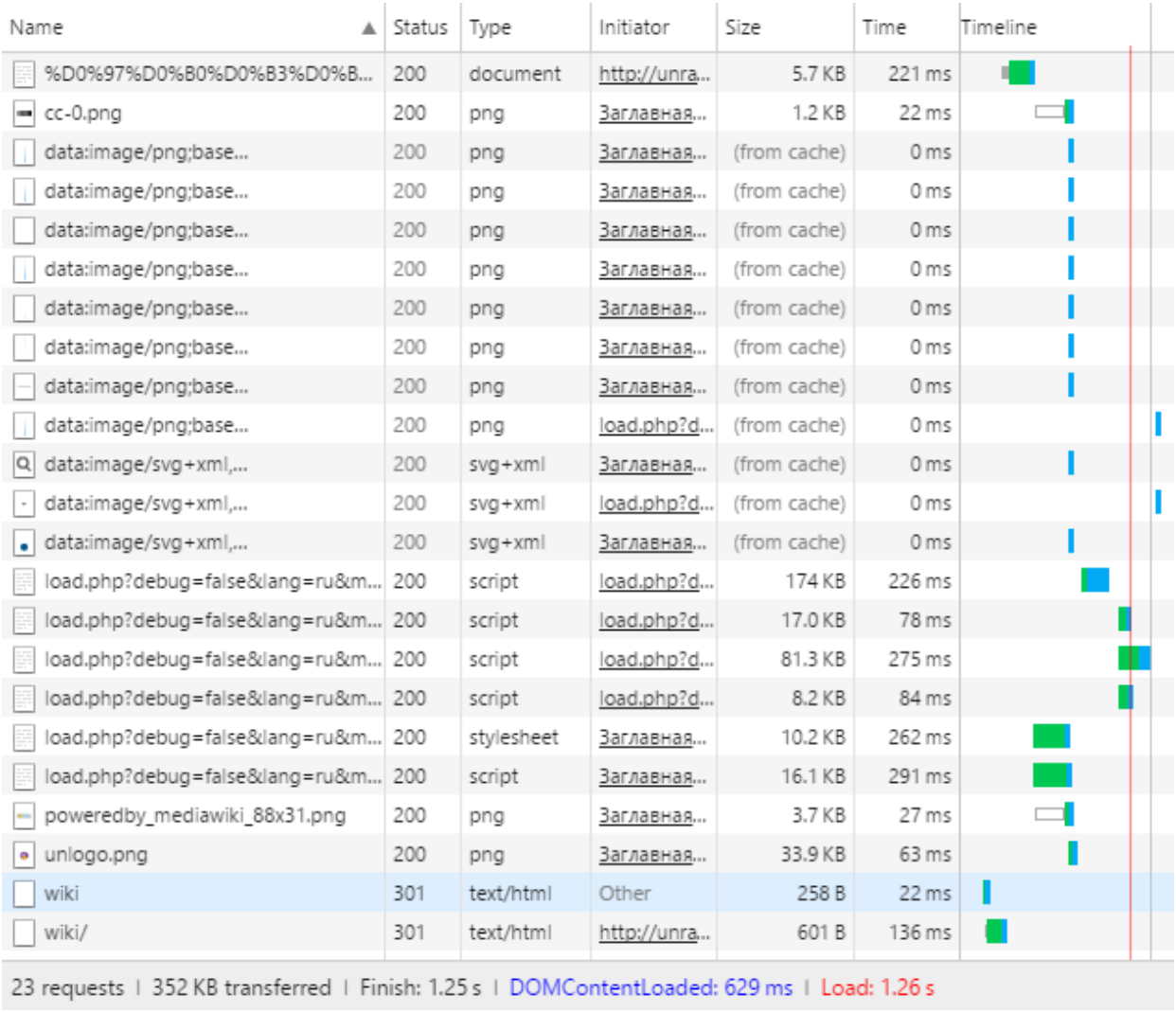
Подгрузка статьи из вики-системы в мобильном приложении осуществляется с помощью HTTP API, предоставляемого вики-движком MediaWiki. Поскольку в ответе на запрос содержится строго запрашиваемая информация без сопроводительного материала, пользователем не подгружаются дополнительные скрипты, таблицы стилей, сопровождающие изображения, необходимые для веб-интерфейса вики-системы, пользователь тратит меньше трафика для получения необходимой информации в качестве статьи.

Теоретически, уменьшение размера загружаемых данных должно снижать нагрузку на устанавливаемое соединение и затрачивать меньшее количество времени от перехода на статью до её отображения на конечном устройстве. Однако стоит принимать во внимание скорость мобильного соединения, которое зависит от выбранного пользователем способа подключения и покрытия сети, как беспроводной (Wi-Fi), так и сотовой связи (GRPS, EDGE, 3G).

Для наглядной демонстрации преимущества подгрузки статей через MediaWiki API над стандартным браузерным решением, генерируемым вики-движком MediaWiki, был использован браузер на базе движка Webkit, с помощью которого были совершены запросы к вики-системе. Сетевое

взаимодействие с вики-системой фиксировалось браузером с помощью инструментов разработчика; список полученных файлов с затраченными на них трафиком и временем обновляется в режиме реального времени для подробной статистики.

На рисунке 3.3 показан список файлов, которые загружает браузер при обращении к заглавной странице вики-системы, используя стандартный вход в браузерное решение, генерируемое вики-движком. В основном в списке присутствуют файлы, необходимые для обеспечения работы веб-интерфейса вики-системы. Браузер, разработанный на базе движка Webkit, сделал 23 запроса к серверу, загрузил 352 килобайта данных, в том числе изображений в формате PNG с поддержкой прозрачности, что довольно увеличивает размер файла, затратив на передачу данных 1,25 секунды.



Name	Status	Type	Initiator	Size	Time	Timeline
%D0%97%D0%B0%D0%B3%D0%B...	200	document	http://unra...	5.7 KB	221 ms	
cc-0.png	200	png	Заглавная...	1.2 KB	22 ms	
data:image/png;base...	200	png	Заглавная...	(from cache)	0 ms	
data:image/png;base...	200	png	Заглавная...	(from cache)	0 ms	
data:image/png;base...	200	png	Заглавная...	(from cache)	0 ms	
data:image/png;base...	200	png	Заглавная...	(from cache)	0 ms	
data:image/png;base...	200	png	Заглавная...	(from cache)	0 ms	
data:image/png;base...	200	png	Заглавная...	(from cache)	0 ms	
data:image/png;base...	200	png	Заглавная...	(from cache)	0 ms	
data:image/png;base...	200	png	load.php?d...	(from cache)	0 ms	
data:image/svg+xml,...	200	svg+xml	Заглавная...	(from cache)	0 ms	
data:image/svg+xml,...	200	svg+xml	load.php?d...	(from cache)	0 ms	
data:image/svg+xml,...	200	svg+xml	Заглавная...	(from cache)	0 ms	
load.php?debug=false&lang=ru&m...	200	script	load.php?d...	174 KB	226 ms	
load.php?debug=false&lang=ru&m...	200	script	load.php?d...	17.0 KB	78 ms	
load.php?debug=false&lang=ru&m...	200	script	load.php?d...	81.3 KB	275 ms	
load.php?debug=false&lang=ru&m...	200	script	load.php?d...	8.2 KB	84 ms	
load.php?debug=false&lang=ru&m...	200	stylesheet	Заглавная...	10.2 KB	262 ms	
load.php?debug=false&lang=ru&m...	200	script	Заглавная...	16.1 KB	291 ms	
poweredby_mediawiki_88x31.png	200	png	Заглавная...	3.7 KB	27 ms	
unlogo.png	200	png	Заглавная...	33.9 KB	63 ms	
wiki	301	text/html	Other	258 B	22 ms	
wiki/	301	text/html	http://unra...	601 B	136 ms	

23 requests | 352 KB transferred | Finish: 1.25 s | DOMContentLoaded: 629 ms | Load: 1.26 s

Рисунок 3.3 – Загружаемые файлы, необходимые для функционирования веб-интерфейса

На рисунке 3.4 показан ответ на запрос, разобранный в главе 2.3, к MediaWiki API. В ответе на запрос к API находится только содержимое запрашиваемой статьи, в данном случае заглавной страницы, браузер, разработанный на базе движка Webkit, делает всего один запрос и загружает 1,3 килобайта данных за 297 миллисекунд.

Для мобильного устройства, которое может использовать соединение посредством мобильной связи (в том числе GPRS и EDGE), с низкой скоростью и высокой стоимостью за мегабайт, использование второго метода получения содержимого статей является более выгодным, как в финансовом, так и во временном плане.

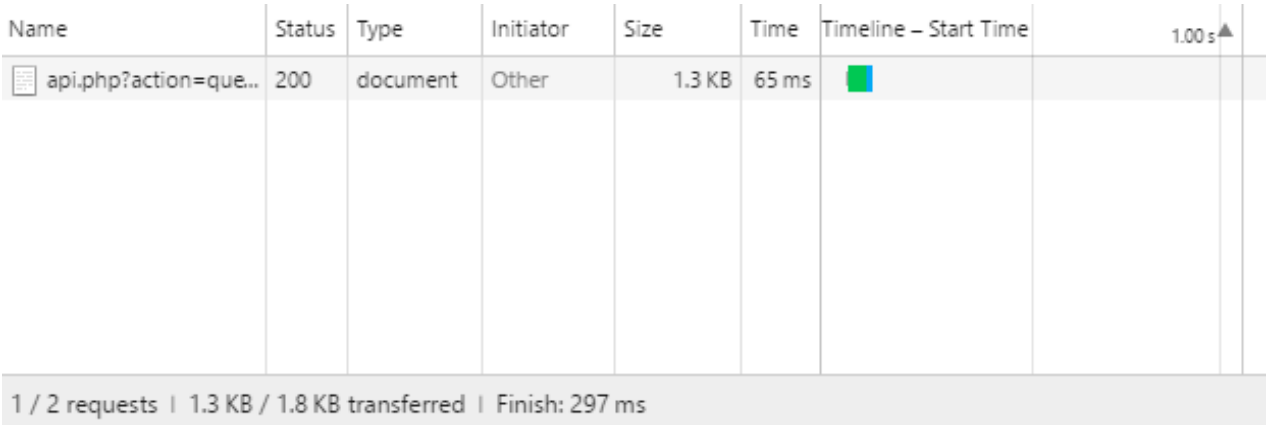
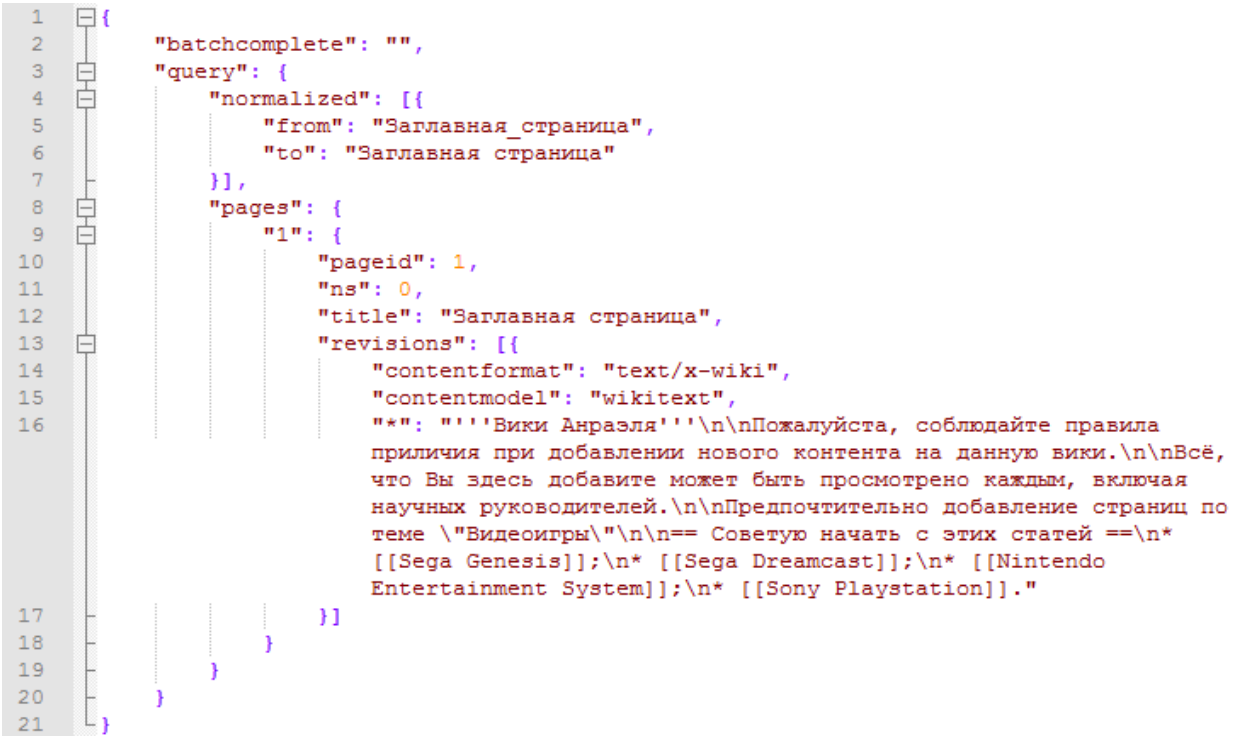


Рисунок 3.4 – Полученный ответ на запрос к MediaWiki API

Полученный ответ приходит в формате JSON, который структурирован для удобства чтения как человеком, так и машиной [9]. На рисунке 3.5 представлено содержимое ответа на запрос (2.1), форматированный для удобства чтения.



```
{
    "batchcomplete": "",
    "query": {
        "normalized": [{
            "from": "Заглавная_страница",
            "to": "Заглавная страница"
        }],
        "pages": {
            "1": {
                "pageid": 1,
                "ns": 0,
                "title": "Заглавная страница",
                "revisions": [{
                    "contentformat": "text/x-wiki",
                    "contentmodel": "wikitext",
                    "*": "'''Вики Анраэля'''\n\nПожалуйста, соблюдайте правила
                    приличия при добавлении нового контента на данную вики.\n\nВсё,
                    что Вы здесь добавите может быть просмотрено каждым, включая
                    научных руководителей.\n\nПредпочтительно добавление страниц по
                    теме \"Видеоигры\"\n\n== Советую начать с этих статей ==\n*
                    [[Sega Genesis]];\n* [[Sega Dreamcast]];\n* [[Nintendo
                    Entertainment System]];\n* [[Sony Playstation]]."
                }]
            }
        }
    }
}
```


Рисунок 3.5 – Ответ от MediaWiki API в формате JSON

Существующие библиотеки и классы в Android API позволяют работать с файлами формата JSON «из коробки» – при использовании имеющихся классов в приложении, разработчик получает необходимый функционал для парсинга JSON-данных для дальнейшего использования полученных данных внутри приложения [17, 23]. Подобные решения существуют и для других форматов, которые позволяет возвращать MediaWiki API, к примеру, XML.

3.3 Разработка мобильного приложения с помощью Android SDK

3.3.1 Выбор версии Android API

Android SDK предоставляет разработчику большие возможности для работы с мобильными устройствами на базе операционной системы Android OS посредством использования библиотек, поддерживаемых всеми устройствами данной операционной системы, позволяя охватывать большую часть рынка мобильных устройств, несмотря на огромную фрагментацию рынка в плане производителей и поддерживаемых систем, включая разные версии SDK.

В настоящее время разработчиками операционной системы Android рекомендовано использование Android SDK версии не ниже 19-ой (Android 4.4 Kitkat), но с развитием операционной системы ожидается очередное повышение рекомендуемой версии API. Однако, устаревшие функции и библиотеки не выводятся из использования при выходе новых версий; в этом случае для устаревших функции заканчивается поддержка, но в выходящий пакеты они всё ещё входят. Благодаря этому приложения, разработанные с использованием более ранних версий Android SDK, способны запускаться и корректно работать на современных версиях операционной системы Android OS.

На рисунке 3.6 показано распределение установленных на мобильных устройствах всех версий операционной системы Android OS [4].

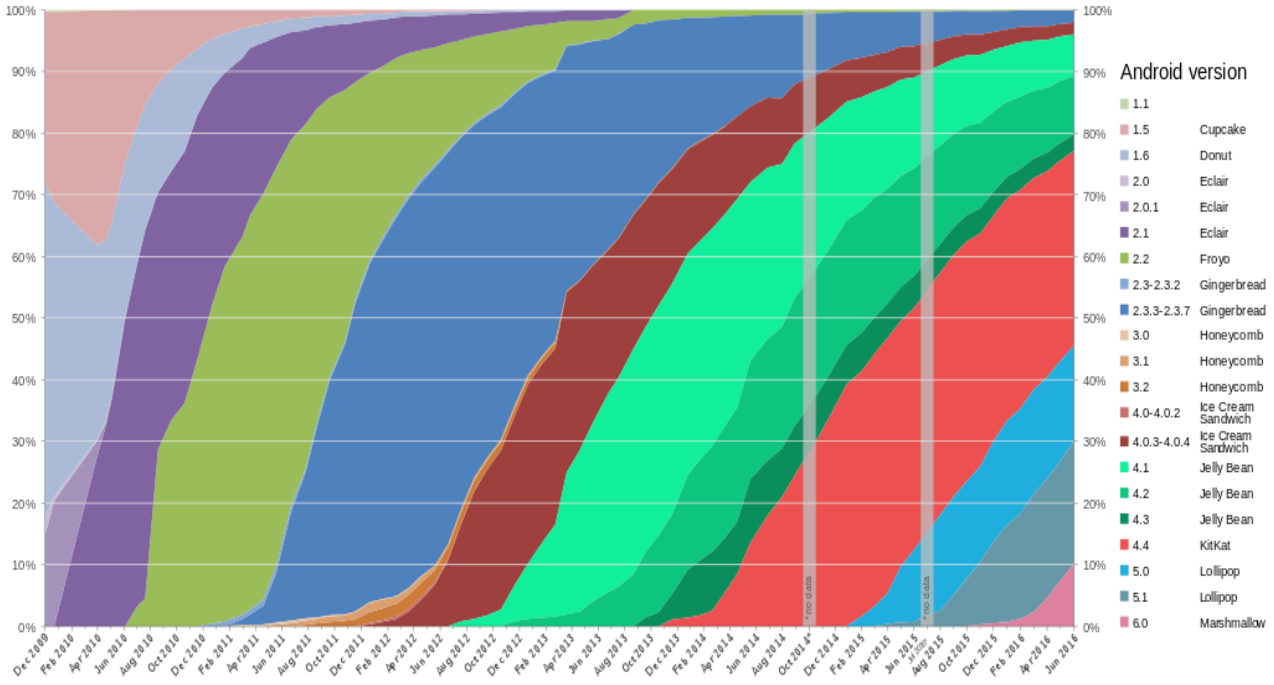


Рисунок 3.6 – Глобальное использование версий операционной системы Android

Приложение разрабатывалось с использованием Android API 19-ой версии для охвата наибольшей части мобильных устройств под управлением Android OS на сегодняшний день. Данная версия предоставляет возможность использовать функции, доступные абсолютному большинству мобильных устройств, что не оказывает негативного влияния на разработку вследствие актуальности.

3.3.2 Проектирование основной части мобильного приложения

Для выполнения поставленной задачи приложение должно обладать необходимыми разрешениями для проведения соответствующих операций с устройством. Следуя политике приложений Android OS, любое реализованное на данной платформе приложение при установке должно запросить разрешения на виды операций, проводимые приложением с мобильным устройством во время его работы. Весь список запрашиваемых разрешений при установке выводится на экран для пользователя, чтобы тот мог решить, стоит ли устанавливать данное мобильное приложение на своё устройство.

В случае приложения, которое разрабатывается исходя из данной поставленной задачи, будут запрашиваться разрешения на работу с сетевыми подключениями (обеспечивается доступ к беспроводным сетям для связи с сервером с развёрнутой вики-системой и синхронизацией данных между вики-системой и хранилищем на мобильном устройстве) и на доступ к файловой системе (для хранения настроек приложения, для доступа к хранилищу, в котором и будет производиться сохранение статей для последующего доступа к ним в режиме без прямого подключения).

Все основные настройки, свойства и запросы на разрешения мобильного приложения, разработанного с помощью Android API, описаны в файле AndroidManifest.xml и предоставлены в формате XML. Необходимые мобильному приложению разрешения записываются разработчиком непосредственно в этот файл. На рисунке 3.7 перечислены описанные в файле AndroidManifest.xml разрешения.

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

Рисунок 3.7 – Разрешения для разрабатываемого мобильного приложения

После первой сборки, на этапе установки собранного приложения, система выведет диалоговое окно с перечислением требуемых приложением разрешений, как представлено на рисунке 3.8.

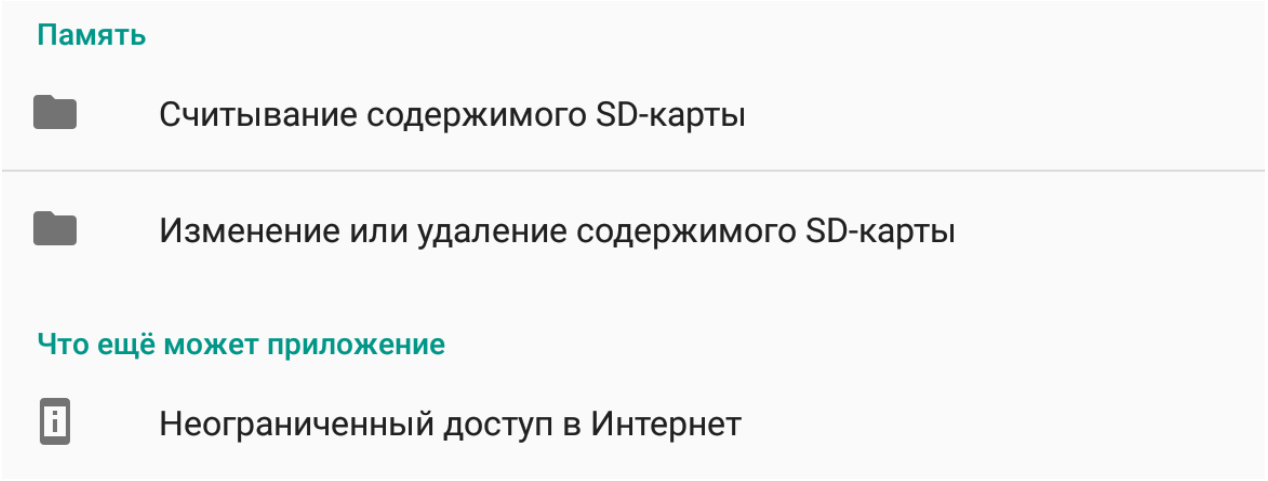


Рисунок 3.8 – Запрос разрешений при установке приложений

Одним из базовых элементов, необходимых для функционирования приложения, является активность (Activity), которая представляет собой часть пользовательского интерфейса, которая может быть связана с непосредственной логикой. Для активности создаётся разметка, для удобства настройки которой современные среды разработки содержат графические инструменты, генерирующие необходимый XML для внешнего вида Activity, исходя из разметки, созданной пользователем с помощью предоставленных модулей, включающих элементы управления, таблицы дополнительной разметки, виджеты и т.д. Графический интерфейс для создания дизайна Activity, предоставленный в Android Studio, IDE на базе IntelliJ IDEA, показан на рисунке 3.9.

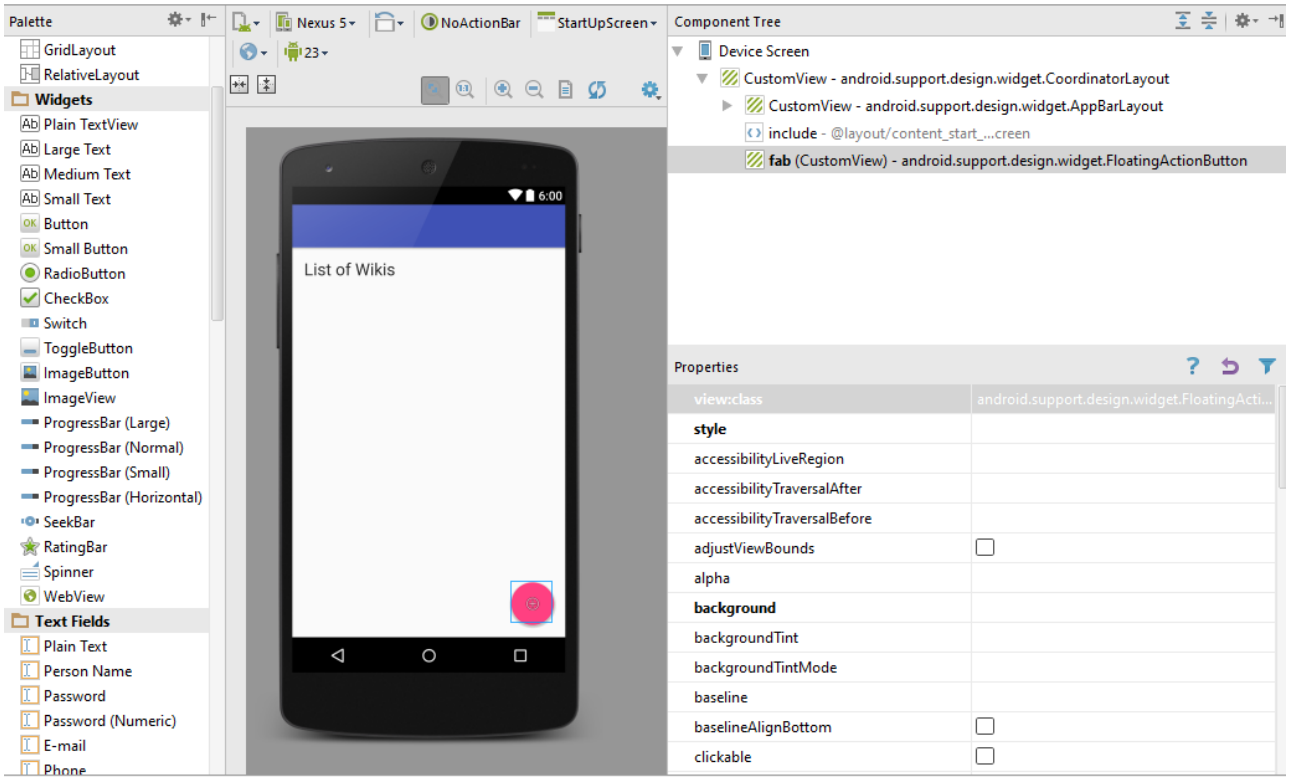


Рисунок 3.9 – Графический дизайнерский интерфейс Android Studio

Каждая активность отвечает за один «экран», который проходит пользователь в приложении один за другим, переходя к необходимому в момент времени функционалу.

Активность прописывается в AndroidManifest.xml вместе со своими параметрами, для возможной связи между другими активностями.

Для приложения необходимы следующие активности:

- активность настроек;
- активность подключения к вики-системе;
- активность просмотра статьи;
- активность списка сохранённых статей.

Активность может иметь параметр начальной активности; в этом случае выбранная активность будет запускаться при запуске приложения. В данном случае, активность подключения к вики-системе будет являться начальной. На рисунке 3.10 изображена активность WikiAdditionPage, с помощью которой пользователь может обратиться к необходимой вики-системе по URL.

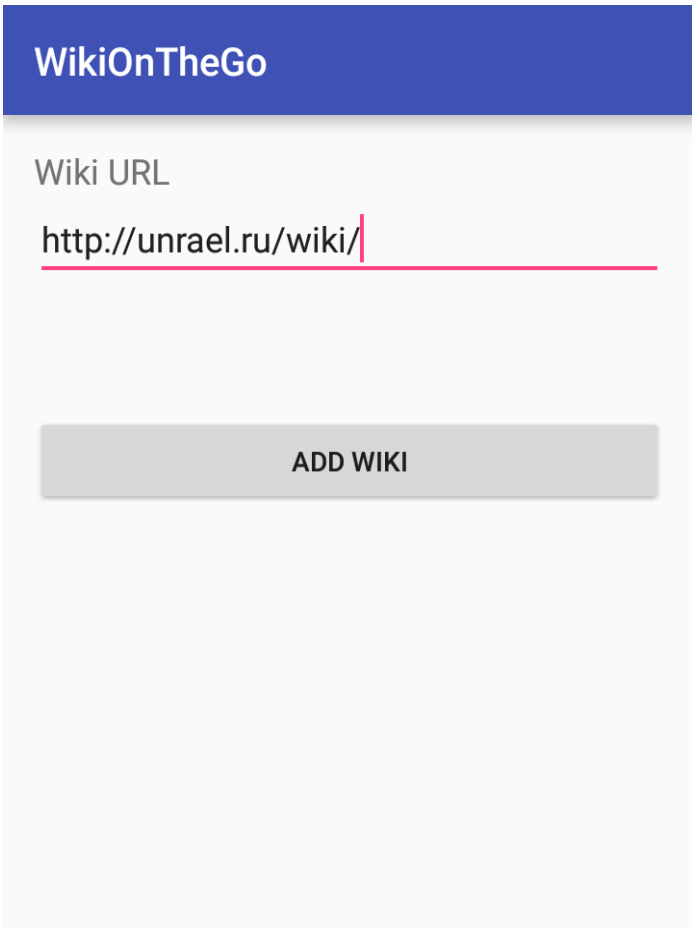


Рисунок 3.10 – Активность WikiAdditionPage в работающем приложении

Активность WikiAdditionPage содержит в себе элемент относительной разметки, на которой расположены текстовая строка, подпись к этой строке и кнопка. В коде самой активности происходит проверка текста внутри текстовой строки как адреса, с помощью встроенной функции Android API, по нажатию кнопки:

- если введённый текст в текстовом поле не является валидным URL, приложение сообщает пользователю с помощью всплывающей подсказки, что введённый им текст не является правильным URL-адресом;
- если введённый текст в текстовом поле является валидным URL, текст из текстового поля передаётся в отдельный поток для совершения подключения к вики-системе, возможно находящейся по указанному адресу.

На рисунке 3.11 представлен код, отвечающий за реализацию приведённого выше алгоритма.

```
public void onClick(View view) {
    String url = wikiURL.getText().toString();
    if(URLUtil.isValidUrl(url)) {
        Context context = getApplicationContext();
        RequestTask isReachable = new RequestTask(url, context, WikiAdditionPage.this);
        isReachable.execute();
    } else {
        Context context = getApplicationContext();
        CharSequence text = "URL is NOT valid";
        int duration = Toast.LENGTH_SHORT;

        Toast toast = Toast.makeText(context, text, duration);
        toast.show();
    }
}
```

Рисунок 3.11 – Проверка адреса, введённого в текстовую строку на активности WikiAdditionPage

Непосредственное обращение к вики-системе происходит уже из другого потока, ибо в Android API запрещается проводить потенциально долгосрочные операции в потоке, который работает с пользовательским интерфейсом, иначе появляется риск заставить приложение не отвечать системе и привести приложение к преждевременному закрытию.

На рисунке 3.12 представлен код, отвечающий за тестовое асинхронное обращение к возможно развёрнутой по данному адресу вики-системе.

```
public class RequestTask extends AsyncTask<String, String, String> {
    String uri;
    Context context;
    Activity activity;
    int code = 0;

    public RequestTask(String uri, Context cntxt, Activity activity) {
        this.uri = uri;
        this.context = cntxt;
        this.activity = activity;
    }

    @Override
    protected String doInBackground(String... params) {
        HttpURLConnection urlConnection = null;
        try {
            URL url = new URL(this.uri + "api.php");
            urlConnection = (HttpURLConnection) url.openConnection();
            urlConnection.connect();
            InputStream in = new BufferedInputStream(
                    urlConnection.getInputStream());
            code = urlConnection.getResponseCode();
            if(urlConnection.getResponseCode() == 200) {
                return "Wiki found";
            }
        } catch (IOException e) {
            Toast toast = Toast.makeText(this.context, "Exception: " + e.getMessage(), Toast.LENGTH_SHORT);
            toast.show();
        } finally {
            urlConnection.disconnect();
        }
        return "No wiki found";
    }

    @Override
    protected void onPostExecute(String result) {
        Toast toast = Toast.makeText(this.context, result, Toast.LENGTH_SHORT);
        toast.show();

        if(code == 200) {
            Intent i = new Intent(context, PageActivity.class);
            i.putExtra("url", uri + "api.php?action=query&prop=revisions&titles=Main_Page&format=json");
            activity.startActivity(i);
        }
    }
}
```

Рисунок 3.12 – Класс RequestTask, отвечающий за тестовое соединение

В параллельном потоке будет осуществлён запрос к API потенциально развёрнутой вики-системы по адресу, который был передан из активности:

- если по адресу, предоставленному из активности, нет вики-системы (невозможно обратиться к API), приложение сообщает пользователю с помощью всплывающей подсказки, что по введённому пользователем адресу невозможно найти развёрнутую вики-систему.

- Если по адресу, предоставленному из активности, находится вики-система и имеется доступ к API, то генерируется запрос заглавной страницы вики-системы и передаётся в активность просмотра статьи.

3.3.3 Отображение статьи в приложении

Активность PageActivity представляет область для отображения содержимого статьи, полученной с помощью запросов к API, прошедшего через парсер вики-текста. Полученный после преобразований текст статьи помещается в элемент WebView, расположенный на PageActivity, и отображается пользователю на экране. На рисунке 3.13 представлено приложение с открытой PageActivity.

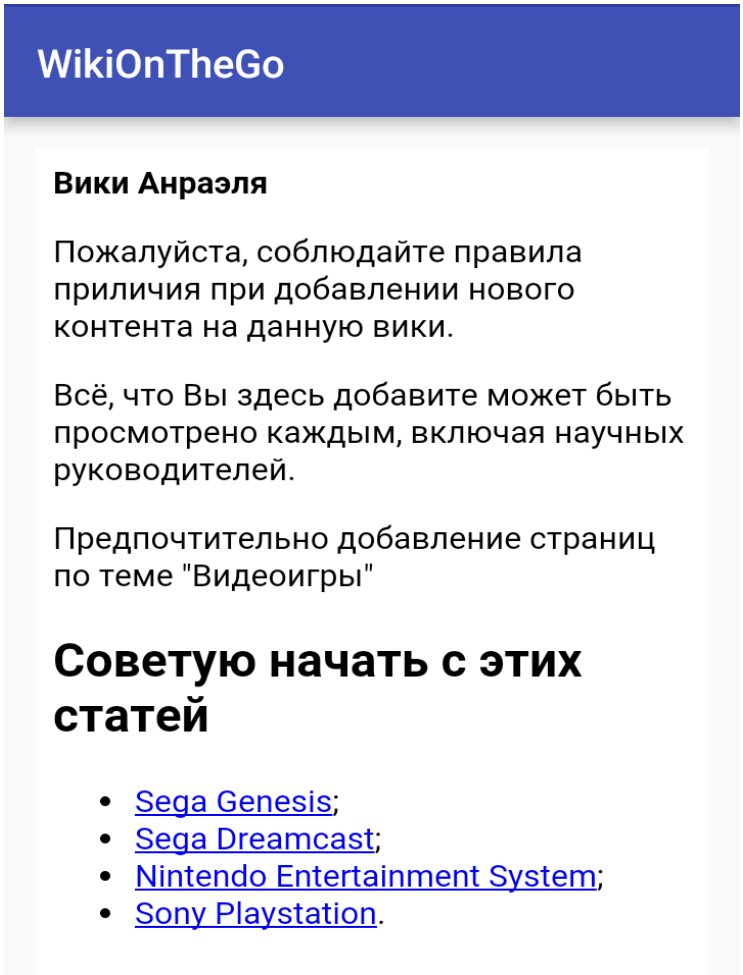


Рисунок 3.13 – Просмотр заглавной страницы вики с помощью PageActivity

Возможно отображение статьи в форме ответа от API вики-системы, как показано на рисунке 3.14.

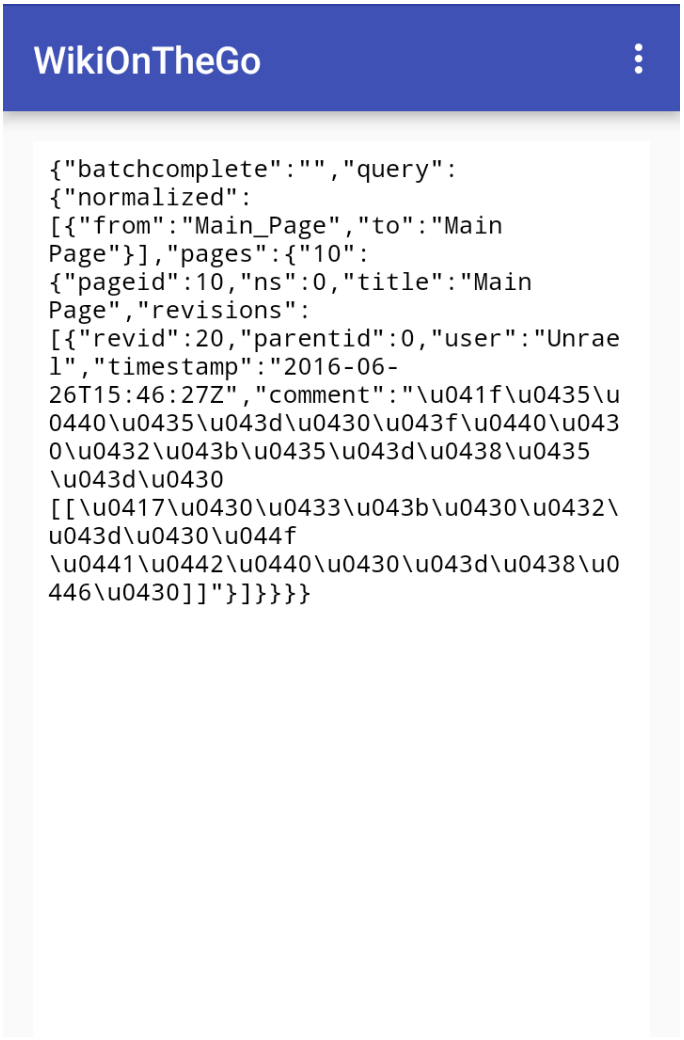


Рисунок 3.14 – Просмотр перенаправления на заглавную статью из PageActivity

Активность SavedPagesActivity позволяет пользователю просмотреть список сохранённых на устройстве статей, с возможным переходом на статьи, включая переходы в режиме без прямого подключения, когда подгрузка статьи осуществляется напрямую из хранилища устройства.

На рисунке 3.15 показана активность SavedPagesActivity в работающем приложении со списком сохранённых статей

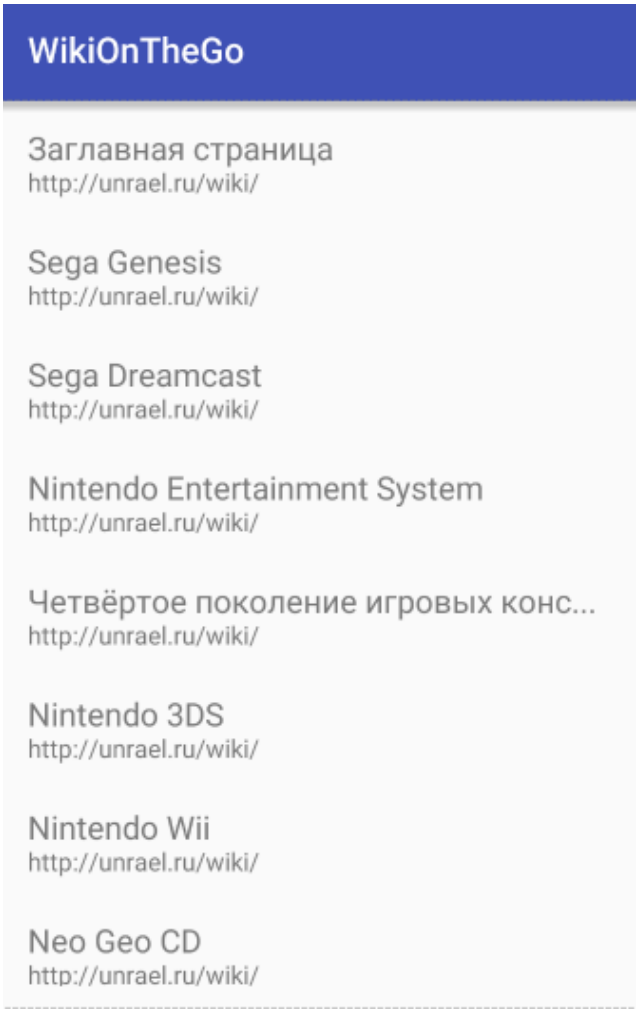


Рисунок 3.15 – Список сохранённых на устройстве статей, приведённый в SavedPagesActivity

При переходе на одну из приведённых на рисунке 3.15 статей, срабатывает код, отвечающий за алгоритм с рисунка 3.2, где приложение пытается подгрузить статью, которая уже существует в хранилище устройства, при этом обращая внимание на наличие или отсутствие прямого доступа к серверу с развёрнутой на нём вики-системой.

-

Заключение

В ходе исследовательской работы были получены необходимые требования к системе, состоящей из двух основных частей: сервера с вики-системой и мобильного устройства с разработанным мобильным приложением. Исходя из этих требований, были выбраны непосредственные компоненты для использования в системе: вики-движок MediaWiki, необходимый для развёртывания вики-системы, и мобильное устройство на базе мобильной операционной системы Android OS.

В ходе практической части исследования был разработан алгоритм, позволяющий пользователю мобильного приложения подгружать статьи, первоначально хранящиеся на сервере с развёрнутой вики-системой, как в режиме прямого подключения к вики-системе (непосредственный запрос к API с последующим получением ответа и обработкой с помощью парсера), так и в автономном режиме (загрузка сохранённой ревизии статьи из хранилища мобильного устройства, при условии, что статья уже была кэширована на целевом устройстве).

При загрузке содержимого статьи с помощью API, количество затраченного трафика снижается за счёт дополнительных скриптов, таблиц стилей и сопроводительных изображений.

Полученное содержимое статей в формате JSON при кешировании сохраняется в хранилище устройства и, при последующей подгрузке статьи из хранилища, обрабатывается с помощью парсера.

Была реализована система синхронизации содержимого статей с мобильного приложения на вики-систему, с помощью системы ревизий, поддерживаемой вики-движком MediaWiki. В результате при появлении любых конфликтных ситуаций, пользователь получает возможность внесения изменений в ручном режиме.

В результате работы была разработана система из вики-системы, развёрнутой с помощью движка MediaWiki на сервере, отвечающем

требованиям движка, и из мобильного приложения, разработанного с помощью Android API и работающего на мобильном устройстве на базе операционной системы Android OS. Система спроектирована таким образом, что статьи могут быть сохранены на устройстве пользователя и содержимое этих статей будет доступно, даже если вики-система недоступна или устройство не имеет подключения к сети Интернет на момент загрузки статьи.

Список используемой литературы

1. Лафоре Р. Структуры данных и алгоритмы Java; Пер. с англ. – Е. Матвеев – СПб. : Питер, 2016. – 704 с.

2. Харди Б. Программирование под Android; Пер. с англ. – Е. Матвеев – СПб. : Питер, 2014. – 592 с.

3. Шестаков В.К. Извлечение онтологий из wiki-систем / В.К. Шестаков. - Научно-технический вестник Санкт-Петербургского государственного университета информационных технологий, механики и оптики, 2012, № 1.

4. Android Developers [Электронный ресурс] // Google – Электрон. дан. – [Б. м.], 2016. – URL - https://developer.android.com/index.html (дата обращения 02.05.2016).

5. MediaWiki [Электронный ресурс] // Википедия – Электрон. дан. – [Б. м.], 2016. – URL: https://ru.wikipedia.org/wiki/MediaWiki (дата обращения: 12.04.2016).

6. What is MediaWiki? [Электронный ресурс] // Wikimedia Foundation – Электрон. дан. – [Б. м.], 2016. – URL: https://www.mediawiki.org/wiki/Manual:What_is_MediaWiki%3F (дата обращения 12.04.2016).

7. Wiki [Электронный ресурс] // Wikipedia, the free encyclopedia – Электрон. дан. – [Б. м.], 2016. – URL: https://en.wikipedia.org/wiki/Wiki (дата обращения: 07.04.2016).

8. Barrett D. J. MediaWiki. – O'Reilly Media, 2008.

9. Bray T. The javascript object notation (json) data interchange format. – 2014.

10. Choate M. S. Professional Wikis – Wrox, 2007.

11. Cunningham W. Wiki History – WikiWikiWeb, 2008.

12. Darcey L. Android Application Development in 24 Hours – Sams, 2010.

13. Friesen J. Android Recipes: A Problem-Solution Approach – Apress, 2011.

14. Griffiths Dawn, Griffiths David. Head First Android Development: A Brain-Friendly Guide – O'Reilly, 2015.

15. Koren Y. Working with MediaWiki – 2nd edition. WikiWorks Press, 2014.

16. Mednieks Z., Dornin L, Meike G. B., Nakamura M. Programming Android – O'Reilly, 2011.

17. Oaks S. Java Performance: The Definitive Guide – O'Reilly, 2014.

18. Orloff J., Rahman M. MediaWiki 1.1 Beginner's Guide – Packt Publishing, 2010.

19. Peroni S. Semantic Data Interfaces for the Masses // Semantic Web Technologies and Legal Scholarly Publishing. – Springer International Publishing, 2014. – С. 195-256.

20. Rahman M. MediaWiki administrators' tutorial guide: install, manage, and customize your MediaWiki installation. – Packt Publishing Ltd, 2007.

21. Schaffert S. IkeWiki: A Semantic Wiki for Collaborative Knowledge Management // 15th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises. – WETICE, 2006. – С. 388-396.

22. Smyth N. Android Studio Development Essentials – Android 6 Edition – Payload Media, 2015.

23. Wolfson M. Android Developer Tools Essentials – O'Reilly, 2013.

Приложение А Листинг классов приложения

Класс StartUpScreen

```
package com.unrael.wikionthego;

import android.content.Intent;
import android.os.Bundle;
import android.support.design.widget.FloatingActionButton;
import android.support.design.widget.Snackbar;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.View;
import android.view.Menu;
import android.view.MenuItem;

public class StartUpScreen extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_start_up_screen);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        FloatingActionButton fab = (FloatingActionButton)
findViewById(R.id.fab);
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(StartUpScreen.this,
SavedPagesActivity.class);
                startActivity(intent);
            }
        });
    }
```

```
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.menu_start_up_screen,
menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
                int id = item.getItemId();

        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }
}
```

Класс WikiAdditionPage

```
package com.unrael.wikionthego;

import android.content.Context;
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.webkit.URLUtil;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
```

```
import java.net.URI;

public class WikiAdditionPage extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Intent intent = getIntent();
        setContentView(R.layout.activity_wiki_addition_page);

        Button addWikiButton = (Button)
findViewById(R.id.addWikiButton);
        final EditText wikiURL = (EditText)
findViewById(R.id.wikiURL);
        addWikiButton.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String url = wikiURL.getText().toString();
                if(URLUtil.isValidUrl(url)) {
                    Context context = getApplicationContext();
                    RequestTask isReachable = new RequestTask(url,
context, WikiAdditionPage.this);
                    isReachable.execute();
                } else {
                    Context context = getApplicationContext();
                    CharSequence text = "URL is NOT valid";
                    int duration = Toast.LENGTH_SHORT;

                    Toast toast = Toast.makeText(context, text,
duration);
                    toast.show();
                }
            }
        });
    }
```

```
}
```

Класс RequestTask

```
package com.unrael.wikionthego;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.os.AsyncTask;
import android.widget.Toast;

import java.io.BufferedInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.URL;

public class RequestTask extends AsyncTask<String, String, String>
{
    String uri;
    Context context;
    Activity activity;
    int code = 0;

    public RequestTask(String uri, Context cntxt, Activity
activity) {
        this.uri = uri;
        this.context = cntxt;
        this.activity = activity;
    }

    @Override
    protected String doInBackground(String... params) {
        HttpURLConnection urlConnection = null;
```

```
        try {
            URL url = new URL(this.uri + "api.php");
            urlConnection = (HttpURLConnection)
url.openConnection();
            urlConnection.connect();
            InputStream in = new BufferedInputStream(
                    urlConnection.getInputStream());
            code = urlConnection.getResponseCode();
            if(urlConnection.getResponseCode() == 200) {
                return "Wiki found";
            }
        } catch (IOException e) {
            Toast toast = Toast.makeText(this.context, "Exception:
" + e.getMessage(), Toast.LENGTH_SHORT);
            toast.show();
        } finally {
            urlConnection.disconnect();
        }
        return "No wiki found";
    }

    @Override
    protected void onPostExecute(String result) {
        Toast toast = Toast.makeText(this.context, result,
Toast.LENGTH_SHORT);
        toast.show();
        if(code == 200) {
            Intent i = new Intent(context, PageActivity.class);
            i.putExtra("url", uri +
"api.php?action=query&prop=revisions&titles=Main_Page&format=json"
);
            activity.startActivity(i);
        }
    }
}
```