**1** ОБЗОР ЛИТЕРАТУРЫ

## **1.1** Обзор существующих аналогов

На этапе проектирования системы были тщательно изучены существующие аналоги. Одним из наиболее приближенных примеров является приложение «Daily Budget» (рисунок 1.1).



Рисунок 1.1 – Главная страница приложения «Daily Budget»

Данный проект – это приложение для ведения бюджета. В нем реализована функциональность введения доходов и расходов. Все доходы и расходы разделяются на группы (рисунок 1.2), что удобно для группировки и представления данных. Это также удобно для отображения графиков ежемесячных затрат для каждой конкретной группы.

Также в приложении предусмотрено указание постоянного ежемесячного дохода, а также постоянных затрат, чтобы не вводить данную информацию ежемесячно вручную. Исходя из ежемесячного дохода и желаемой суммы для сохранения, высчитывается ежедневный бюджет, который доступен пользователю для достижения поставленной цели. Это сделано для того, чтобы наглядно на графике можно было убедиться, вкладывается ли пользователь в положенный ему ежедневный бюджет (рисунок 1.3).

Также, к основным недостаткам приложения относятся:

* разработано только под операционную систему (ОС) iOS;
* нет синхронизации данных для работы с разных устройств;
* все данные хранятся у пользователя в телефоне, что не безопасно;
* ограниченный набор графиков;
* жесткая привязка к работе только с одной валютой.



Рисунок 1.2 – Группы затрат приложения «Daily Budget»



Рисунок 1.3 – График попадания в бюджет приложения «Daily Budget»

Еще один аналог - приложение «ViZi Бюджет» (рисунок 1.2). Это приложения для ведения бюджета предоставляет больший набор функций в сравнении с предыдущим аналогом. Есть возможность синхронизации данных, введения собственных категорий затрат, а также возможность иметь несколько счетов. Приложение имеет бесплатную и платную версии, и представлено для двух мобильных платформ: iOS и Android.



Рисунок 1.4 –Приложение «ViZi Бюджет»

Данное приложение также не лишено недостатков:

* сложный пользовательский интерфейс;
* отсутствие веб-версии приложения;
* слишком большой набор функций для новичка.

## **1.2** Аналитический обзор

Веб-приложение – это клиент-серверное приложение, в котором клиентом выступает браузер, а сервером – веб-сервер. Логика веб-приложения распределена между сервером и клиентом, хранение данных осуществляется, преимущественно, на сервере, обмен информацией происходит по сети. Одним из преимуществ такого подхода является тот факт, что клиенты не зависят от конкретной операционной системы пользователя, поэтому веб-приложения являются кроссплатформенными сервисами. Книга Сэмма Руби «Гибкая разработка веб-приложений»[1] содержит информацию о том, как создать простое веб-приложение с нуля. В книгу включены описание простейшего рабочего процесса (с использованием текстового редактора и системы контроля версий Git), основы технологий клиентской стороны (HTML, CSS, jQuery, JavaScript), основы серверных технологий (HTTP, базы данных), основы облачного развертывания и несколько примеров правильной практики написания кода (функции, MVC). С её помощью можно изучить фундаментальные основы языка, научиться программировать, используя объекты и массивы, а также ментальные модели, которые соответствуют этому типу разработки программного обеспечения (ПО).

Мобильное приложение – программное обеспечение, предназначенное для работы на смартфонах, планшетах и других мобильных устройствах [2]. Мобильные приложения ушли далеко вперёд в связи с развитием сотовой связи и беспроводных технологий (3G, 4G или LTE, Wi-Fi, WiMax).

Многие приложения могут быть предустановлены на устройство в процессе производства, загружены пользователем с помощью различных платформ для распространения ПО или существовать в формате веб-приложений.

Основные операционные системы, на базе которых строятся мобильные приложения – iOS, Android и WindowsMobile.

Существуют различные каналы распространения приложений: специализированный онлайн магазин – AppStore, AndroidMarket, MicrosoftStore, самостоятельный поиск или скачивание в Интернете.

## **1.3** Архитектура клиент-сервер

Важным вопросом в разработке веб-приложений является их архитектура. Наиболее эффективную работу приложений обеспечивает архитектура «клиент-сервер» [3].

Особенностью архитектуры является то, что само веб-приложение находится и выполняется на сервере, клиент при этом получает только результаты работы. Работа приложения основывается на получении запросов от пользователя (клиента), их обработке и выдачи результата. Вся передача запросов и результатов их обработки происходит через Интернет.

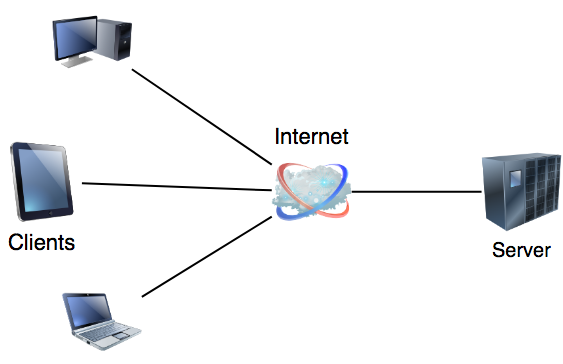


Рисунок 1.5 - Архитектура клиент-сервер

Приемом данных от клиента и их передачей на сервер, а также отображением результатов запросов обычно занимается специальное приложение – браузер (Microsoft Edge, Firefox, Safari, Google Chrome). Одной из функций браузера является отображение данных, полученных из Интернета, в виде страницы, описанной на языке HTML, следовательно, результат, передаваемый сервером клиенту, должен быть представлен на этом языке.

На стороне сервера веб-приложение выполняется специальным программным обеспечением (веб-сервером). Основные задачи, которые он решает – принимает запросы клиентов, обрабатывает их, формирует ответ в виде страницы, описанной на языке HTML, и передает его клиенту.

В процессе обработки запроса пользователя веб-приложение компонует ответ на основе исполнения программного кода, работающего на стороне сервера, веб-формы, страницы HTML, другого содержимого, включая графические файлы.

В результате, как уже было сказано, формируется HTML-страница, которая и отправляется клиенту. Получается, что результат работы веб-приложения идентичен результату запроса к традиционному веб-сайту, однако, в отличие от него, веб-приложение генерирует HTML-код в зависимости от запроса пользователя, а не просто передает его клиенту в том виде, в котором этот код хранится в файле на стороне сервера. То есть веб-приложение динамически формирует ответ с помощью исполняемого кода – так называемой исполняемой части. За счет наличия исполняемой части, веб-приложения способны выполнять практически те же операции, что и обычные Windows-приложения, с тем лишь ограничением, что код исполняется на сервере, в качестве интерфейса системы выступает браузер, а в качестве среды, посредством которой происходит обмен данными, – Интернет.

К наиболее типичным операциям веб-приложения относятся:

* прием данных от пользователя и сохранение их на сервере;
* аутентификация пользователя и отображение интерфейса системы, соответствующего данному пользователю;
* отображение постоянно изменяющейся информации;
* выполнение различных действий по запросу пользователя: извлечение данных из базы данных (БД), добавление, удаление, изменение данных в БД;
* проведение сложных вычислений.

К основным достоинствам архитектуры «клиент-сервер» можно отнести следующие:

* все данные хранятся на сервере, который защищён лучше большинства клиентов;
* использовать ресурсы сервера могут клиенты с разными аппаратными платформами и операционными системами;
* возможность распределить функции вычислительной системы между несколькими компьютерами в сети, что позволяет упростить обслуживание системы;
* на сервере проще организовать контроль прав и полномочий, чтобы разрешать доступ к данным только клиентам с соответствующими правами;
* позволяет объединить различные клиенты.

Среди недостатков можно выделить необходимость квалифицированного администратора данной системы. В случае использования централизованной системы, неработоспособность основного сервера может сделать неработоспособным всё приложение. Немаловажным фактором является высокая стоимость оборудования.

## **1.4** REST-сервис

REST – это набор архитектурных принципов и стиль проектирования приложений, ориентированный на создание сетевых систем, в основе которых лежат механизмы для описания и обращения к ресурсам [4].

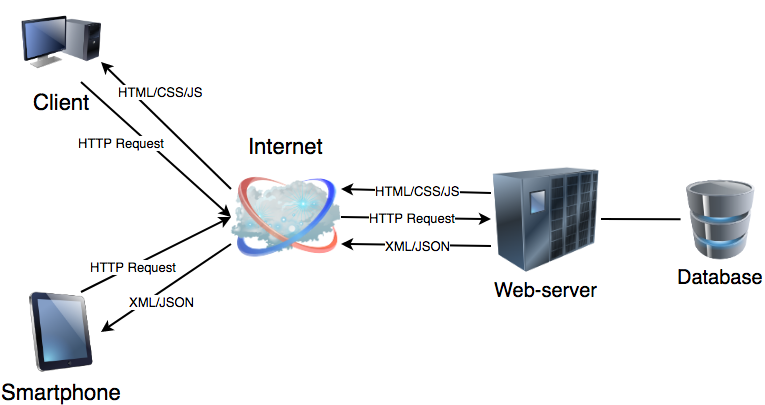


Рисунок 1.6 - REST-сервис

Примером такой системы служит World Wide Web – распределённая система, предоставляющая доступ к связанным между собой документам, расположенным на различных компьютерах, подключённых к сети Интернет [5].

REST определяет строгое разделение ответственности между компонентами клиент-серверной системы, облегчающее реализацию необходимых актеров. Еще одна цель REST – упрощение семантики взаимодействия компонентов сетевых систем для улучшения масштабируемости и повышения производительности. В REST заложен принцип автономности запросов, который означает, что запросы, обрабатываемые сервером или клиентом, обязательно должны включать всю информацию, необходимую для их понимания.

Для обмена данными стандартных медиа-типов в системах REST используется минимальное количество запросов. REST-системы используют URI (универсальные идентификаторы ресурсов) для поиска и получения доступа к представлениям необходимых ресурсов. В течение последних нескольких лет разработчики создавали REST-сервисы для своих приложений, используя самые разнообразные технологии. Также архитектура REST отличается своей простотой, требуя от приложений обеспечить только возможность приема сообщений с HTTP- заголовками.

## **1.5** Apache Cordova

Apache Cordova — это платформа разработки мобильных приложений с открытым исходным кодом. Она позволяет использовать стандартные веб-технологии, такие как HTML5, CSS3 и JavaScript для кроссплатформенной разработки, избегая родного языка разработки для каждой из мобильных платформ. Приложения выполняются внутри обертки нацеленной на каждую платформу и полагаются на стандартные API (программный интерфейс приложения) для доступа к датчикам устройства, данным и состоянию сети [6].

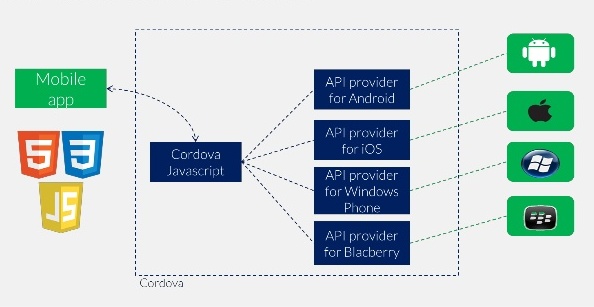


Рисунок 1.7 - Программный интерфейс приложения Cordova

Основные причины использования Apache Cordova:

* нужно расширить приложение на более чем одну платформу, без необходимости повторно реализовать его для каждого языка разработки платформ и набора инструментов;
* нужно развернуть веб-приложение, которое упаковано для распространения в различных магазинах приложений;
* необходимо смешивание компонентов собственного приложения с WebView (специальный браузер окно).

Само приложение реализовано как веб-страницы, по умолчанию локальный файл под названием index.html, который ссылается на любой CSS, JavaScript, изображения, файлы мультимедиа или другие ресурсы необходимы для его запуска. Приложение выполняет как WebView в пределах оболочки приложения, которую вы распространяете в магазины приложений.

Таким образом, с помощью данной платформы есть возможность писать слабонагруженные приложения, который будут работать на любой мобильной платформе. При этом затраты на разработку будут минимальны, поскольку код будет написан один раз, а запускаться будет как в браузере, так и на любом мобильном устройстве.