МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»
Кафедра IT-инжиниринг

ДОПУЩЕН К ЗАЩИТЕ
Заведующий кафедрой
PhD, доцент
_____ Т.С. Картбаев
«____» _____ 2019 г.

ДИПЛОМНЫЙ ПРОЕКТ

На тему: Разработка мобильного приложения «Журнал посещаемости»

Специальность: 5В070400 – «Вычислительная техника и программное обеспечение»

Выполнил: Рамазан М.Б. Группа: ВТз-17-1
Научный руководитель: к.т.н., доцент Тусупова Б.Б.

Консультанты:
по экономической части: к.э.н., профессор _____ Ж.Г. Аренбаева
«05» 06 2019 г.
по безопасности
жизнедеятельности: д.т.н., ст. преп. _____ Ш.Ш. Бекбасаров
«14» 05 2019 г.
по применению
вычислительной техники: ст. преп. _____ М.Н. Майкотов
«14» 05 2019 г.

Нормоконтролер: ст. преп. _____ А.А. Айтказина
«15» 05 2019 г.

Рецензент: к.т.н., доцент _____ Д.М. Ескендирова
«____» _____ 2019 г.

Алматы 2019

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»
Институт систем управления и информационных технологий
Кафедра IT-инжиниринг

Специальность 5В070400 – «Вычислительная техника и программное обеспечение»

ЗАДАНИЕ
на выполнение дипломного проекта

Студенту Рамазан Мақсат Болатұлы

Тема проекта: Разработка мобильного приложения «Журнал посещаемости»

Утверждена приказом по университету № 152 от «27» декабря 2018 г.
Срок сдачи законченного проекта «24» мая 2019 г.
Исходные данные к проекту (требуемые параметры результатов исследования (проектирования) и исходные данные объекта): Руководство системы менеджмента качества на предприятии; международные стандарты ИСО-9001, данные преддипломной практики.
Перечень вопросов, подлежащих разработке в дипломном проекте, или краткое содержание дипломного проекта:
- аналитическая часть;
- проектная часть;
- экспериментальная часть;
- экономическая часть;
- безопасность жизнедеятельности;
- приложение А. Техническое задание;
- приложение Б. Листинг программы;
- приложение В. Акт внедрения.

Перечень графического материала (с точным указанием обязательных чертежей): представлены 15 таблиц, 20 иллюстрации.

Основная рекомендуемая литература:

— Коровкин С. Д., Левенец И. А., Ратманова И. Д., Старых В. А., Щавелёв Л. В. Решение проблемы комплексного оперативного анализа информации хранилищ данных // СУБД. - 1997. - № 5-6. - С. 47-51.
— Распределенная обработка данных: курс лекций / Сост. Найханова Л.В. – Улан-Удэ, Издательство ВСГТУ, 2001. – 122 с.

— Гофман В.Э. Хомененко А.Д. Delphi 7 – СПб.: БХВ – Санкт-Петербург, 2005 – 1152с.

Консультации по проекту с указанием относящихся к ним разделов проекта

Раздел	Консультант	Сроки	Подпись
Экономическая часть	Аренбаева Ж.Г.	05.05.19	
Безопасность жизнедеятельности	Бекбасаров Ш.Ш.	14.05.19	
Программное обеспечение	Майкотов М.Н.	14.05.19	
Нормоконтролер	Айтказина А.А.	01.05.19 - 15.05	

ГРАФИК
подготовки дипломной проекта

Наименование разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю	Примечание
Аналитическая часть	14.01.2019 - 12.03.2019	выполнено
Проектная часть	15.03.2019 - 13.05.2019	выполнено
Экспериментальная часть	13.05.2019 - 17.05.2019	выполнено

Дата выдачи задания «26» 10 2019 г.

Заведующий кафедрой _____ Т.С. Картбаев

Научный руководитель проекта _____ Б.Б. Тусупова

Задание принял к
исполнению студент _____ М. Б. Рамазан

АҢДАТПА

Мақсаты: Мұғалімдердің жұмысын атқаратын мобильды қосымша құрастыру.

Міндеттері:

1. Мұғалімдердін оқушыларды бақылау жүйесің мобильды қосымшаға аудару.

2. Веб қолданбаның әкімшілік жүйесін құрастырып оны мобильды қосымшаны бақылауға реттеу.

Дипломдық жоба 3 бөлімнен және қосымшадан, 76 беттен, 22 суреттен, 7 кестеден, 18 әдебиет көздерінен тұрады.

Бірінші бөлімінде теориялық мәліметтер жиналған және зерттелген. Практикаға байланысты қажет ақпараттары көңіл бөлінді.

Екінші бөлімінде жиналған теориялық мәліметтерді ескере отырып, дипломдық жобаның практика түрінде жүзеге асыратын құрал жобаланды. Бұл құралдың қалай жұмыс істейтіні туралы, оның артықшылықтары мен кемшіліктері талданды.

Үшінші бөлімде қауіпсіздік және еңбекті қорғауға байланысты қажет факторлар нақтыланды және енгізілді.

Қосымшада бағдарламаны жазудағы кодтар көрсетіліп кетті. Олардың әрқайсысына анықтама берілді.

АННОТАЦИЯ

Цель: спроектировать мобильное приложение для фиксации посещаемости студентов преподавателями.

Задачи:

1. Вывести из процесса наблюдения за посещаемостью студентов человеческий фактор.

2. Создать административную панель управления для мобильного приложения

Дипломный проект состоит из 3 частей и приложений, 76 страниц, 22 рисунка, 7 таблиц, 18 источников литературы.

В первой части были собраны и изучены теоретические данные. Соответствующая информация была предоставлена для практики.

Во второй части учитывая собранный теоретический материал, разработано средство в виде практической реализации дипломной работы. Было проанализировано, как работает это средство его преимущества и недостатки.

В третьем разделе описываются и внесены факторы безопасности и охраны труда.

В приложении показано коды, которые написаны в программе.

ABSTRACT

Purpose: design a mobile application for fixing student attendance by teachers.

Objective:

1. To remove the human factor from the process of monitoring student attendance.

2. Create an administrative control panel for a mobile application

The diploma project consists of 3 parts and applications, 76 pages, 7 tables, 22 sources of literature.

In the first part, theoretical data were collected and studied. Relevant information was provided for practice.

In the second part, taking into account the collected theoretical material, a tool has been developed in the form of practical implementation of the thesis. It was analyzed how this tool works its advantages and disadvantages.

The third section describes and introduces factors of safety and labor protection.

The appendix shows the codes that are written in the program

Содержание

Введение

Данный дипломный проект посвящен актуальной проблеме фиксации посещаемости студентами занятий в режиме on-line посредством использования современных гаджетов в виде мобильного приложения. В мире имеется достаточное количество аналогов, но построить по-настоящему масштабную и адаптируемую систему еще никому не удавалось в мире открытых проектов, возможно и существуют аналоги, удовлетворяющие интересам отдельных лиц и организации, но, к сожалению, эти проекты недоступны широким массам в виду конфиденциальности проектов, и возможного хранения коммерческих тайн в коде программ.

Актуальность данного проекта в своевременном реагировании на события, происходящие в каком-либо сообществе, примером может служить отслеживание студента на посещения, и оперативные меры пресечения негативных действии, чтобы в будущем избежать крупных проблем, как для студента, так и для учебного заведения. На данный момент большинство конфликтов на образовательной почве в основном происходит, в связи с тем, что информация доносится до ответственных лиц несвоевременно.

На сегодняшний день в Казахстане и СНГ аналога предлагаемого в данного приложения не существует, именно поэтому вышесказанное показывает актуальность и необходимость темы дипломного проекта.

Реализованное приложение имеет называние «S-Assistant» и уже работает в тестовом режиме. В приложении имеются эндпоинты серверного приложения, взаимодействующее с базами данных, и все данные в приложении отображаются оттуда, для динамичности контента в мобильном приложении.

Необходимость разветвления компонентов целого приложения на мобильное приложение, серверное приложение и базы данных имеет в основе цель оптимизации всего программного кода, оптимизируя каждый запрос на несколько миллисекунд, в общей перспективе, где мобильное приложение имеет около пятидесяти запросов это сыграет очень важную роль.

В данной дипломном проекте имеется три главы из основы. Первая глава описывает краткий анализ среды разработки, параметры состоянии информационной системы, а также описывается приоритетность намеченных задач, где расписан подробный план по реализации мобильного приложения. Вторая глава об обосновании выбора системы управления базами данных и описания его преимуществ перед другими базами данных для мобильного приложения. Третья глава содержит описания о среде разработки и технической документации мобильного приложения. Создание прототипа и ввода в эксплуатацию мобильного приложения.

1 Анализ современного состояния вопроса

1.1 Постановка задачи дипломного проекта

Необходимо спроектировать и программно реализовать мобильное приложение в среде разработки «Android Studio» используемая операционная система Android. Приложение должно будет иметь функционал преподавателя вносить занятия, отмечать студентов, а также все эти действия проходят через интерфейс мобильного приложения в базу данных PostgreSQL, которая находится на сервере. Задачи, решаемые мобильным приложением:

- фиксация посещаемости студентом занятии;
- своевременное реагирование на пропуски студентов;
- системный подход к методологии преподавания;
- исключение человеческого фактора.

Требование к ПО:

При разработке мобильного приложения необходимо использовать:

- фреймворк «Laravel» серверного языка программирования PHP для написания бэкенд части мобильного приложения, административной панели управления данным приложением;
- среду разработки «PHPStorm», для удобства отслеживания программного кода, для ведения системы контроля версии;
- язык программирования Java для мобильного приложения, со специально разработанными пакетами и библиотеками для удобства работы по протоколу HTTP;
- среду разработки «Android Studio», это самый оптимальный выбор для разработки мобильного приложения, по причине централизованности всех инструментов необходимых для разработки мобильного приложения;
- использовать систему контроля версии с репозиторием расположенным на Github, намного удобнее хранить программный код в репозитории с возможностью хранения каждой версии на случай, если приложению понадобится откатиться на версию раньше, на предмет выявления серьезных угроз безопасности, логических ошибок в архитектуре приложения, мелких недоработок и ошибок.

Предпочтительное программное обеспечение, именуемое в дальнейшем средой разработки мобильного приложения является Android Studio и операционная система Android с открытым режимом разработчика.

Язык программирования, который был использован для реализаций ПО «S-Assistant» Java 14 LTS, что обозначает долгую техническую поддержку создателями этого языка данной версии языка.

Функциональное предназначение:

Функциональное предназначение приложения, помощь в работе учителей. Так как система реализуется для закрытой целевой аудитории все участники данной системы создаются вручную в административной панели данного приложения, построенное на серверном языке программирования

PHP, а именно реализация на его фреймворке Laravel. В приложении реализована авторизация пользователя через логин, а также возможность создать предметы, и студентов касательно категории путем добавления информации через мобильное приложение. Для пользователя есть свой профиль, где указано имя и номер сотового для связи. Авторизованный пользователь имеет доступ ко всему функционалу, которые находятся в мобильном приложении, а также имеется функционал удаления записей по неактуальности.

Требования к информационному обеспечению:
В качестве СУБД необходимо использовать PostgreSQL, его отличие от остальных систем управления базами данных, то, что скорость выполнения выборок по базе данных намного быстрее, существуют сложные операторы, реализованные из нескольких простых операторов, этого можно добиться и в других системах, но лучше использовать проверенный временем инструмент, чем создавать новый инструмент, на проверку работоспособности которого уйдет больше времени. Его главный недостаток — это неспособность реагировать на большое количество запросов единовременно, на этот случай данная СУБД снабжается собственным программным обеспечением под названием «PGBouncer» которая служит своего рода буфером между «S-Assistant» из которого идет запрос на какие-либо данные и базой данных.

1.1.1 Хранение информации

Информация в мобильном приложении должна храниться в специальном хранилище, доступном только этому приложению. При каждом подключении устройства к сети Интернет приложение отправляет данные на сервер для актуализации их в базе данных. Веб-приложение может располагаться на том же сервере, что и сервис для работы с API и с базой данных.

1.1.2 Язык интерфейса

Интерфейс приложения должен быть реализован на английском языке. Для того чтобы охватить наибольшую аудиторию, по причине распространенности языка.

1.1.3 Механизм работы

При отсутствии подключения к сети Интернет Мобильное приложение должно продолжать работать при отсутствии подключения к сети Интернет, при этом пользователю должен быть доступен весь функционал. Изменения сохраняются локально в мобильном устройстве. Веб-приложение при отсутствии доступа к сети Интернет не должно работать.

1.1.4 Обратная связь

При работе с приложением пользователь должен получать обратную связь от своих действий. В приложении это можно реализовать следующими способами: − внешний вид кнопок изменяется при нажатии на них (меняют оттенок); − поля, заполняемые пользователем, выделяются цветом; − для соответствия рекомендациям для приложений Android от Google, нужно придерживаться принципов Material Design [4].

1.1.5 Ориентация экрана

При работе на мобильных устройствах ориентация экрана может быть портретной или альбомной. В веб-приложении верстка может учитывать смену ориентации экрана, размещать блок информации по центру или изменять вид меню навигации.

1.1.6 Работа с API

Для синхронизации изменений в приложении нужно обмениваться информацией с сервером – конечным хранилищем данных. На сервере будет храниться информация о задачах, их статусах, группах задач, зарегистрированных пользователях. Получая данную информацию с сервера, приложение сохраняет ее в свое локальное хранилище, при необходимости обновляя путем соответствующего запроса к серверу. При работе с API нужно использовать ключ, который мобильное приложение будет добавлять в запросы, чтобы сервер мог идентифицировать пользователя.

1.1.7 Требования к дизайну

− соответствие принципам Material Design мобильных приложений Google;

− дизайн приложения должен быть адаптирован для корректного отображения при различных разрешениях экрана;

− дизайн веб-приложения должен одинаково отображаться в разных популярных браузерах и соответствовать принципам Material Design.

1.1.8 Операционная система и устройства, обеспечивающие работу приложения

Мобильное приложение должно работать на устройствах под управлением операционной системы Android версии 5.0 и выше. Веб-приложение является частью сервиса на сервере приложений. Для работы сервиса требуется сервер со следующими установленными программными

компонентами: − PHP – PostgreSQL. Системные требования сервера должны соответствовать требованиям 10 указанных программных компонентов.

1.2 Используемые языки и программное обеспечение

При разработке мобильного приложения, веб-приложения и сервера рекомендуется использовать следующие технологии и языки программирования:

- Java;
- JavaScript;
- PHP;
- Laravel;
- Android SDK;
- JWT;
- PostgreSQL;
- Nginx.

Язык Java – это объектно-ориентированный язык программирования. Приложения Java транслируются в специальный байт-код, поэтому они могут работать на любой виртуальной Java-машине вне зависимости от компьютерной архитектуры. Виртуальная машина Java (JVM) – программа, обрабатывающая байтовый код и передающая инструкции оборудованию как интерпретатор. Язык Java был выбран, т.к. платформа Android подразумевает разработку приложений на этом языке. В Android используется собственная версия JVM, которая активно поддерживается и является основной средой выполнения приложений для данной платформы.

Язык JavaScript (JS) – прототипно-ориентированный сценарный язык программирования. Является реализацией языка ECMAScript [5]. Наиболее широкое применение язык нашел в браузерах как язык сценариев для придания интерактивности веб-страницам. Именно по причине его большой популярности среди браузеров было решено использовать его для реализации веб-приложения. Основные архитектурные черты: динамическая типизация, слабая типизация, автоматическое управление памятью, прототипное программирование, функции как объекты первого класса. У языка большое сообщество разработчиков, которые поддерживают в актуальном состоянии множество библиотек и программных платформ (framework) для реализации типовых и узкоспециализированных задач.

Android – операционная система для смартфонов, планшетов, электронных книг, цифровых проигрывателей, наручных часов, игровых приставок, нетбуков, очков Google, телевизоров, мультимедийных систем в автомобилях и других устройств. В качестве ядра операционной системы 11 используется Linux [6] и реализации виртуальной машины Java от Google [7]. Android позволяет создавать Java-приложения, управляющие устройством через разработанные Google библиотеки. Данная платформа была выбрана как самая популярная и широко распространенная платформа для мобильных

устройств. Для разработки используется среда Android Studio, основанная на IntelliJ IDEA от JetBrains [8]. Процесс создания Android приложения не требует дополнительных устройств, кроме, Android устройства (можно использовать эмулятор). Android Studio характеризуется:

- гибкой системой сборки Gradle;
- расширенной поддержкой сервисов Google и различных типов устройств;
- богатым функционалом редактором экранов приложений с поддержкой редактирования тем интерфейса;
- возможностью подписания приложений;
- встроенной поддержкой облачной платформы Google и возможности простой интеграции с Google Cloud Messaging и App Engine [9].

PHP – известен, как серверный язык программирования. Это означает, что он работает на веб сервере. Большинство языков веб-программирования являются серверными языками, но некоторые, например, JavaScript, работают на стороне клиента, это означает, что они работают в веб-браузере.

Серверные языки дают вам больше гибкости, так как вы можете то, что трудно осуществить с помощью JavaScript — например, работа с файлами, базами данных, или работа с изображениями. Нужно сказать, что JavaScript распространился очень быстро в наши дни.

Выполнение кода со стороны сервера является более безопасным способом, чем на стороне клиента, как это делает JavaScript. Поскольку код JavaScript отправляется в веб-браузер, для посетителей сайта легко его просмотреть и редактировать. Даже на одной странице сайта можно с легкостью совмещать PHP и JavaScript. Код находящийся на стороне сервера остаётся в веб-сервере и недоступен для посетителей сайта. PHP это инструмент, который находится на веб-сервере и там выполняет PHP скрипты. PHP — программное обеспечение с открытым исходным кодом

PHP представляет собой программное обеспечение с открытым исходным кодом. Это означает, что любой пользователь может получить доступ и работать на PHP. Это помогает быть уверенным, что PHP будет работать в течение длительного времени. PHP можно свободно скачать и использовать. Это является причиной того, что многие хостинг-провайдеры широко используют PHP. Вы обнаружите, что подавляющее большинство веб-хостингом поддерживают работу PHP. PHP ориентирован на разработку веб-приложений, В то время как многие языки программирования могут быть использованы для создания веб-приложений, PHP является одним из языков, специально разработанных для использования в Интернете. PHP имеет множество полезных веб-функций таких, как:

- считывание и обработка веб-форм и куки-файлов;
- функции создания и работы с графикой;
- установка связи с популярными базами данных такими, как MySQL и PostgreSQL;
- функции для работы с HTML.

Laravel – бесплатный веб-фреймворк на основе PHP с открытым кодом, предназначенный для разработки веб приложении и сложных порталов с использованием архитектурной модели MVC (Model View Controller – модель-представление-контроллер). Laravel выпущен под лицензией MIT. Ключевые особенности, лежащие в основе архитектуры Laravel:

- пакеты – позволяют создавать и подключать модули в формате Composer к приложению на Laravel. Многие дополнительные возможности уже доступны в виде таких модулей;

- Eloquent ORM – реализация шаблона проектирования ActiveRecord на PHP. Позволяет строго определить отношения между объектами базы данных. Стандартный для Laravel построитель запросов Fluent поддерживается ядром Eloquent;

- логика приложения – логика разрабатываемого приложения, объявленная либо при помощи контроллеров, либо маршрутов (функций-замыканий). Синтаксис объявлений похож на синтаксис, используемый в каркасе Sinatra;

- обратная маршрутизация связывает между собой генерируемые приложением ссылки и маршруты, позволяя изменять последние с автоматическим обновлением связанных ссылок. При создании ссылок с помощью именованных маршрутов Laravel автоматически генерирует конечные URL;

- REST-контроллеры – дополнительный слой для разделения логики обработки GET- и POST-запросов HTTP;

- автозагрузка классов – механизм автоматической загрузки классов PHP позволяющий без необходимости подключать файлы их определений в include. Загрузка по требованию предотвращает загрузку ненужных компонентов; загружаются только те из них, которые действительно используются;

- составители представлений – блоки кода, которые выполняются при генерации представления (шаблона);

- инверсия управления – позволяет получать экземпляры объектов по принципу обратного управления. Также может использоваться для создания и получения объектов-одиночек (singleton);

- миграции – система управления версиями для баз данных. Позволяет связывать изменения в коде приложения с изменениями, которые требуется внести в структуру базы данных, что упрощает развёртывание и обновление приложения;

- модульное тестирование (юнит-тесты) – играет очень большую роль в Laravel, который сам по себе содержит большое число тестов для предотвращения регрессий (ошибок вследствие обновления кода или исправления других ошибок).

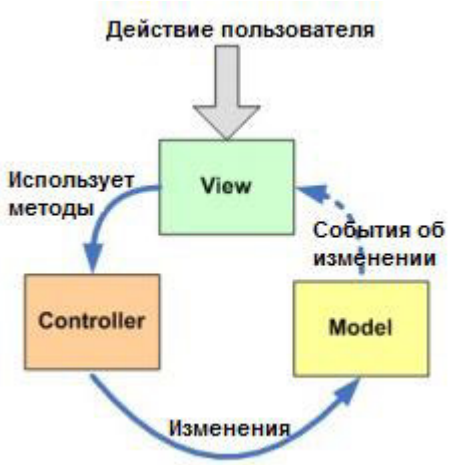


Рисунок 1.1 – Паттерн проектирования MVC

Nginx – веб-сервер и почтовый прокси-сервер, работающий на Unix-подобных операционных системах, nginx позиционируется производителем как простой, быстрый и надёжный сервер, не перегруженный функциями.
Применение nginx целесообразно, прежде всего, для статических веб-сайтов и как обратного прокси-сервера перед динамическими сайтами.
HTTP-сервер:

- обслуживание неизменяемых запросов, индексных файлов, автоматическое создание списка файлов, кэш дескрипторов открытых файлов;
- акселерированное проксирование без системы кэширования, простое распределение нагрузки и отказоустойчивость;
- поддержка и программирование хэш-индексов и систем кеширования при акселерированном проксировании и FastCGI сценариев;
- акселерированная поддержка FastCGI и memcached серверов, простое распределение нагрузки и отказоустойчивость;
- модульность, фильтры, в том числе сжатие (gzip), byte-ranges (докачка), chunked ответы, HTTP-аутентификация, SSI-фильтр;
- несколько подзапросов на одной странице, обрабатываемые в SSI-фильтре через прокси или FastCGI, выполняются параллельно;
- поддержка SSL;
- поддержка PSGI, WSGI;
- экспериментальная поддержка встроенного Perl.

В Nginx рабочие процессы обслуживают одновременно множество соединений, мультиплексируя их вызовами операционной системы select, epoll (Linux) и kqueue (FreeBSD). Рабочие процессы выполняют цикл обработки событий от дескрипторов. Полученные от клиента данные разбираются с помощью конечного автомата. Разобранный запрос последовательно обрабатывается цепочкой модулей, задаваемой конфигурацией из корня веб сервера. Ответ клиенту формируется в буферах, которые хранят данные либо в памяти, либо указывают на отрезок файла.

Буферы объединяются в цепочки, определяющие последовательность, в которой данные будут переданы клиенту. Если операционная система поддерживает эффективные операции ввода-вывода, такие как writev и sendfile, то nginx применяет их по возможности.

JWT – объект JSON, который определен в RFC 7519 как безопасный способ обмена информацией между двумя сторонами открытый стандарт, который определяет компактный и автономный способ безопасной передачи информации.

JWT определяет особую структуру информации, которая отправляется по сети. Она представлена в двух формах – сериализованной и десериализованной. Первая используется непосредственно для передачи данных с запросами и ответами. С другой стороны, чтобы читать и записывать информацию в токен, нужна его десериализация. В несериализованном виде JWT состоит из заголовка и полезной нагрузки, которые являются обычными JSON-объектами.

Заголовок в основном используется для описания криптографических функций алгоритмов шифрования, которые применяются для подписи и/или шифрования токена. Здесь также можно указать дополнительные свойства, например тип содержимого, хотя это редко требуется. Процесс сериализации JWT состоит из кодирования заголовка, полезной нагрузки и подписи, если она есть, с помощью алгоритма base64url. Это простая вариация base64, которая использует URL-безопасный символ _ вместо небезопасных + и /. Дело в том, что некоторые инструменты обработки данных распознают управляющие символы в строке, поэтому их быть не должно.

В отличие от серверных сессий, данные клиентских сеансов хранятся на машине пользователя и отправляются с каждым запросом. Один из способов реализации этого механизма – использование файлов cookie. В них можно хранить не только идентификатор сеанса, но и сами данные. Куки-файлы очень удобно использовать, поскольку они автоматически обрабатываются веб-браузерами. Однако это не единственный способ передавать информацию сеанса. Это также можно делать через HTTP-заголовки, чтобы избежать проблем с CORS, или с помощью URL-параметров.

На рисунке 1.1 изображен процесс сериализации неподписанного токена:

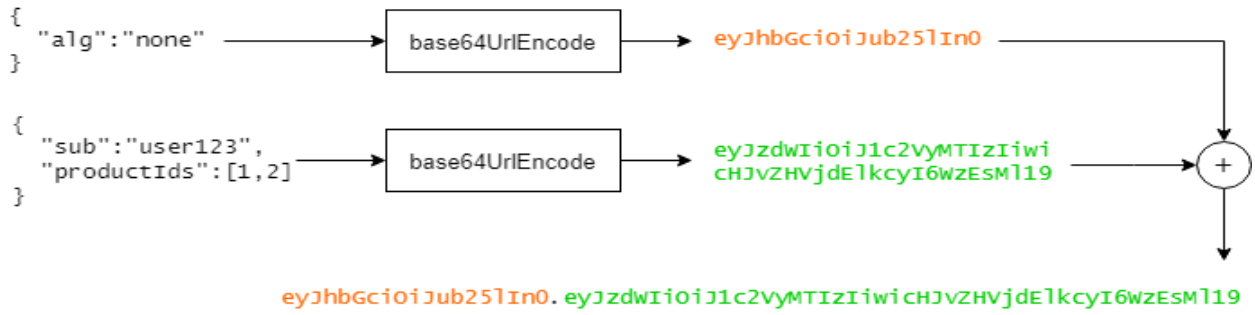


Рисунок 1.2 – Алгоритм шифрования Base64

С сеансами на стороне клиента снимается проблема с масштабируемостью веб-сайтов и мобильных приложении. Однако появляются уязвимости безопасности. Нельзя гарантировать, что клиент не изменяет данные сессии. Например, если идентификатор пользователя хранится в файлах cookie, его легко изменить. Это позволит получить доступ к чужой учетной записи. Чтобы предотвратить подобные действия, нужно обернуть данные в защищенный от несанкционированного воздействия пакет.

Именно для этого и нужен JWT. Благодаря тому, что JSON токены имеют подпись, сервер может проверять подлинность данных сеанса и доверять им. При необходимости их можно даже зашифровать, чтобы защититься от чтения и изменения.

Впрочем, у сеансов на стороне клиента также есть свои минусы. Например, приложение может требовать большого объема пользовательских данных, и их придется отправлять туда-обратно для каждого запроса. Это может даже перекрыть все преимущества простоты и масштабируемости. Таким образом, в реальном мире приложения зачастую совмещают клиентские и серверные сеансы.

1.3 Анализ существующих систем

Рассмотрим уже существующие способы реализации ведения учебного процесса в современном мире. В наши дни очень легко можно создать сайт, мобильное приложение, чтобы с легкостью можно было отследить активность студента. В отличие от традиционных журналов данный подход современен, но подходы решения проблем одинаковы. Такое происходит, потому что информационную систему строят разрозненно, и каждый сервис в таких приложениях живет своей жизнью, актуальность же такого рода приложении в единстве взаимодействующих сервисов, и в своевременном реагировании на негативные события. Аналогов проектируемого сервиса очень много, так как актуальность визуальности отчетов о посещениях нужно для корректного ведения и методологии преподавателя.

Kaspersky Safe Kids – многофункциональное приложение, решающее множество задач. Оно дает возможность родителям ограничить времяпрепровождение ребенка за планшетом или телефоном, блокирует нежелательные приложения и сайты. Также родители могут получить информацию о друзьях своего ребенка из социальных сетей и быть в курсе того, где ребенок находится.

Особенности приложения Kaspersky:

- детский режим – специальная функция Интернета, защита от нежелательных сайтов;

- поиск детей – контроль за местонахождением ребенка, возможность задавать место, где ребенку следует находиться, уведомления о его перемещении;

- правильное развитие – лимит на проведение времени в Интернете;

- здоровое общение – отслеживание публикаций на Facebook, звонков и сообщений;

- взаимопонимание – анализ поисковых запросов ребенка. Так родители будут знать о проблемах детей и смогут в нужный момент помочь им.

Приложение «Родитель» создано для того, чтобы родители контролировали посещаемость и оценки своих детей. Приложение действует для тех родителей, чьи дети обучаются в школах.

После установки приложения на телефоны родителей станут приходить соответствующие уведомления. В рамках проекта созданы личные кабинеты, где хранится информация о посещаемости и успеваемости. Оттуда данные дублируются на телефон родителей. Больше о проекте – на его сайте. Учителя ставят оценки в специальный электронный журнал. Посещаемость отслеживается благодаря установленным турникетам, через которые проходит ребенок.

Вся информация становится доступна после запуска приложения «Родитель» на устройстве Android или iPhone. Данное приложение, представляет собой -электронный дневник ученика, с дополнительными функциями для родителей. Удобная и привлекающая к себе внимание функция push-уведомлений. Как только ребенок прошел через турникет, на мобильное устройство приходит оповещение. Мгновенное уведомление об оценках, домашних заданиях, изменении в расписании – все это будет отображаться на экране телефона родителей. Проект «SmileS.Школьная карта» разработал интуитивно понятный интерфейс, разобраться в котором – дело двух минут.

2 Разработка информационного обеспечения системы

2.1 Функциональная структура информационной системы

При проектировании информационной системы данной тематики, была разработана функциональная структура систем, которая показывает список задач, которые будут выполнять следующие подсистемы:

Подсистема «Администратор»:

- имеет все разрешения.

Подсистема «Редактор»:

- проверяет информацию на достоверность;
- сравнивает сведения преподавателей;
- ведет поиск ошибочных фиксации посещаемости.

Подсистема «Преподаватель»:

- добавляет информацию о посещаемости;
- просматривает имеющиеся отчеты;
- добавляет и удаляет расписание предметов и студентов.

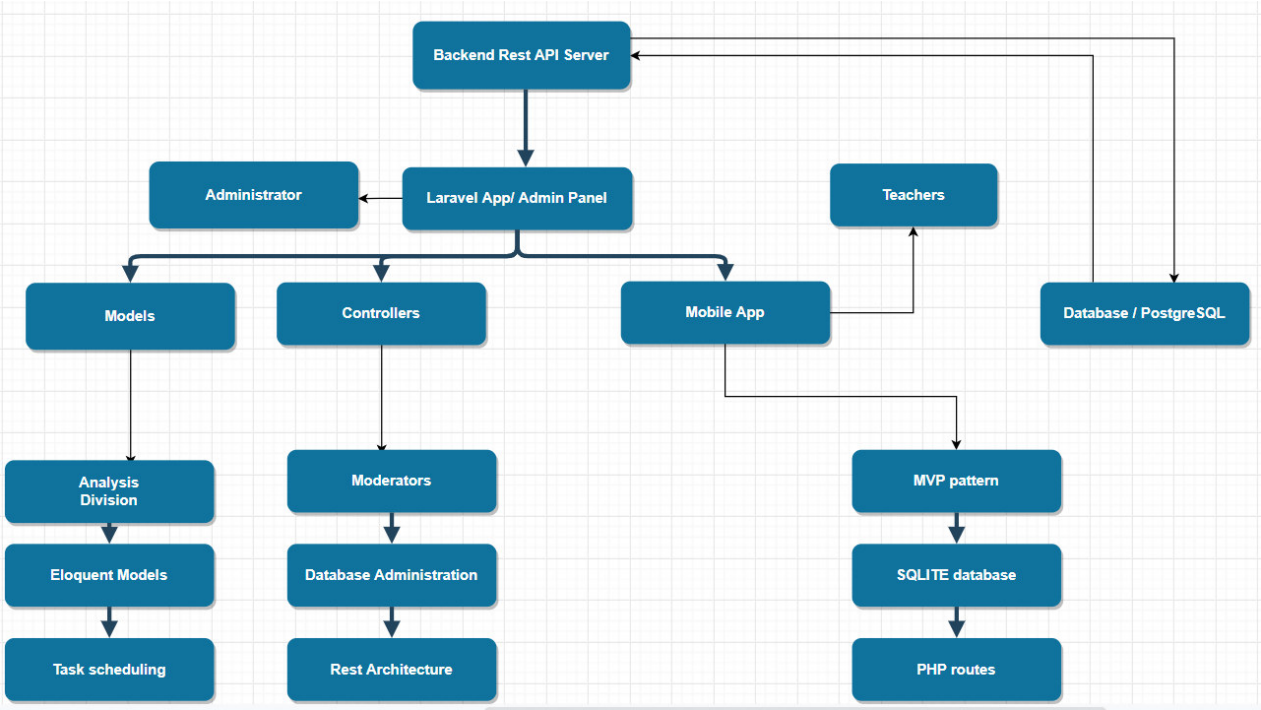


Рисунок 2.1 – Функциональная структура

2.2 Обоснование выбора СУБД

На данный момент имеются очень много видов СУБД, но мало из них имеют такие же качества как у PostgreSQL. Мой выбор СУБД пал именно на PostgreSQL, так как у него очень много достоинств и гибкости в использовании. PostgreSQL создана на основе некоммерческой СУБД Postgres,

разработанной как open-source проект в Калифорнийском университете в Беркли. К разработке Postgres, начавшейся в 1986 году имел непосредственное отношение Майкл Стоунбрейкер, руководитель более раннего проекта Ingres, на тот момент уже приобретенного компанией Computer Associates. Стоунбрейкер и его студенты разрабатывали новую СУБД в течение восьми лет с 1986 по 1994 год. Стоунбрейкер использовал полученный опыт в создании коммерческой СУБД Illustra, продвигаемой его собственной одноименной компанией, а его студенты разработали новую версию Postgres — Postgres95, в которой язык запросов POSTQUEL — наследие Ingres был заменен на SQL, так появилась ныне популярная реляционная СУБД PostgreSQL.

PostgreSQL предоставляет множество различных возможностей, достаточно надежна и имеет хорошие характеристики по производительности. Она работает практически на всех UNIX-платформах, включая UNIX-подобные системы, такие как FreeBSD и Linux. Ее можно применять на Windows NT Server и Windows 2000 Server, а для разработки годятся даже такие системы Microsoft для рабочих станций, как ME. Кроме того, PostgreSQL свободно распространяется и имеет открытый исходный код. [9]
Основные преимущества СУБД PostgreSQL: [8]

PostgreSQL выгодно отличается от многих других СУБД. Она обладает практически всеми возможностями, которые есть в других базах данных (коммерческих или Open Source), а также некоторыми дополнительными.
Приведем перечень функциональных возможностей PostgreSQL:

- транзакции;
- вложенные запросы;
- представления;
- ссылочная целостность - внешние ключи;
- сложные блокировки;
- типы, определяемые пользователем;
- наследственность;
- правила;
- проверка совместимости версий.

Одной из сильных сторон PostgreSQL является ее архитектура. Как и многие коммерческие СУБД, PostgreSQL может применяться в среде клиент-сервер, что дает массу преимуществ, как пользователям, так и разработчикам. Основа PostgreSQL составляет серверный процесс базы данных. Он выполняется на одном сервере. (в этой СУБД еще не реализована технология высокой готовности, как в некоторых других коммерческих системах уровня предприятия, которые могут распределять нагрузку между несколькими серверами, добиваясь таким образом дополнительной масштабируемости и устойчивости к внешним воздействиям.) Доступ из приложений к данным базы осуществляется посредством процесса базы данных. Клиентские программы не могут получить доступ к данным самостоятельно, даже если они работают на том же компьютере, на котором выполняется серверный

процесс. Такое разделение клиентов и сервера позволяет построить распределенную систему.

Можно отделить клиентов от сервера посредством сети и разрабатывать клиентские приложения в среде, удобной для пользователя. Например, можно реализовать базу данных под UNIX и создать клиентские приложения, которые будут работать в системе Microsoft Windows.

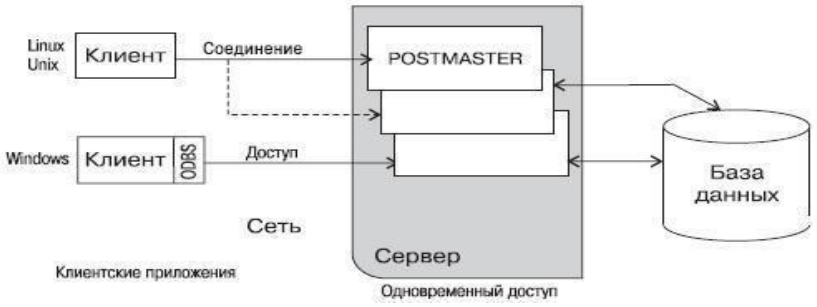


Рисунок 2.2 – Схема процесса работы «Инфографика процессов»

Функции в PostgreSQL являются блоками кода, исполняемыми на сервере, а не на клиенте БД. Хотя они могут писаться на чистом SQL, реализация дополнительной логики, например, условных переходов и циклов, выходит за рамки собственно SQL и требует использования некоторых языковых расширений. Несколько клиентов подсоединяются к серверу по сети. PostgreSQL ориентирован на протокол TCP/IP - это может быть локальная сеть или Интернет. Каждый клиент соединяется с основным серверным процессом базы данных (на схеме – Postmaster), который создает новый серверный процесс специально для обслуживания запросов на доступ к данным конкретного клиента.

Функции могут писаться с использованием различных языков программирования. PostgreSQL допускает использование функций, возвращающих набор записей, который далее можно использовать также, как и результат выполнения обычного запроса. Функции могут выполняться как с правами их создателя, так и с правами текущего пользователя. Иногда функции отождествляются с хранимыми процедурами, однако между этими понятиями есть различие.

Триггеры в PostgreSQL определяются как функции, инициируемые DML-операциями. Например, операция INSERT может запускать триггер, проверяющий добавленную запись на соответствия определённым условиям. При написании функций для триггеров могут использоваться различные языки программирования. Триггеры ассоциируются с таблицами. Множественные триггеры выполняются в алфавитном порядке.

Механизм правил в PostgreSQL представляет собой механизм создания пользовательских обработчиков не только DML-операций, но и операции выборки. Основное отличие от механизма триггеров заключается в том, что правила срабатывают на этапе разбора запроса, до выбора оптимального

плана выполнения и самого процесса выполнения. Правила позволяют переопределять поведение системы при выполнении SQL-операции к таблице. Индексы в PostgreSQL следующих типов: B-дерево, хэш, R-дерево, GiST, GIN. При необходимости можно создавать новые типы индексов, хотя это далеко не тривиальный процесс.

Многоверсионность поддерживается в PostgreSQL — возможна одновременная модификация БД несколькими пользователями с помощью механизма

Расширение PostgreSQL для собственных нужд возможно практически в любом аспекте. Есть возможность добавлять собственные преобразования типов, типы данных, домены (пользовательские типы с изначально наложенными ограничениями), функции (включая агрегатные), индексы, операторы (включая переопределение уже существующих) и процедурные языки.

Наследование в PostgreSQL реализовано на уровне таблиц. Таблицы могут наследовать характеристики и наборы полей от других таблиц (родительских). При этом данные, добавленные в порождённую таблицу, автоматически будут участвовать (если это не указано отдельно) в запросах к родительской таблице.

2.3 Описание базы данных системы

В приложении будет реализована база данных третьего уровня нормализации с помощью вспомогательных таблиц содержащих историю посещении, длительность, расписания.

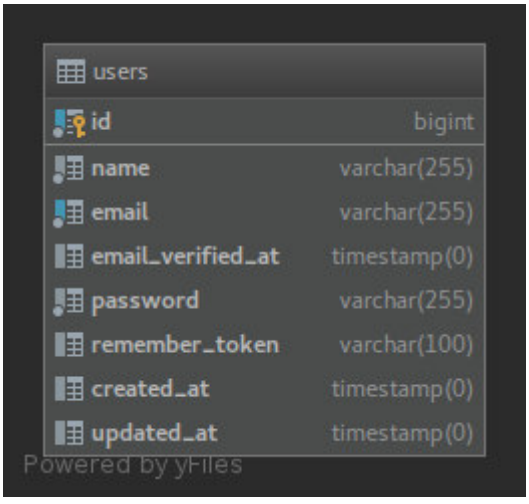


Рисунок 2.3 – Структура таблицы студентов

В таблице студентов хранятся его данные, необходимые для его идентификации преподавателем. Таблица носит исключительно информативный характер, содержащая лишь информацию о студенте,

необходимая для нормального функционирования приложения. Следующая таблица немаловажная по приоритетности таблица преподавателей изображена на рисунке 2.3:

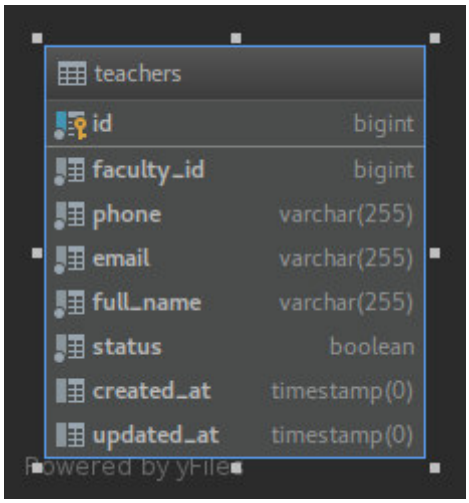


Рисунок 2.4 – Структура таблицы преподавателей

Преподаватели являются главным действующим лицом в мобильном приложении, ниже перечислен функционал доступный им:

- создавать расписание занятии;
- фиксировать посещение занятия студентом;
- создавать пометки;
- создавать учетную запись студента;
- редактировать учетную запись студента;
- редактировать созданные им расписания занятии.

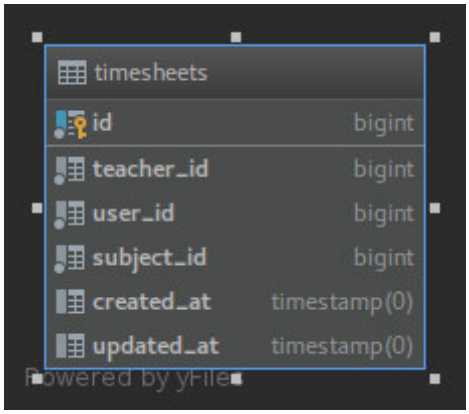


Рисунок 2.5 – Структура таблицы фиксации посещаемости

С помощью таблицы изображенной на рисунке 2.4 становится возможным отслеживать активность студента с помощью фильтров на бэкенд части, где расположена административная панель управления данным

мобильным приложением. Так как мой сервис поделен на две части, то есть мобильное приложение, где работает преподаватель, и административная панель управления на сервере было уместно создать и пользователя в рамках данного веб приложения, другими словами человека, администрирующего контент данного мобильного приложения на сервере, таким образом, была создана таблица пользователей административной панели, изображенная на рисунке 2.5.

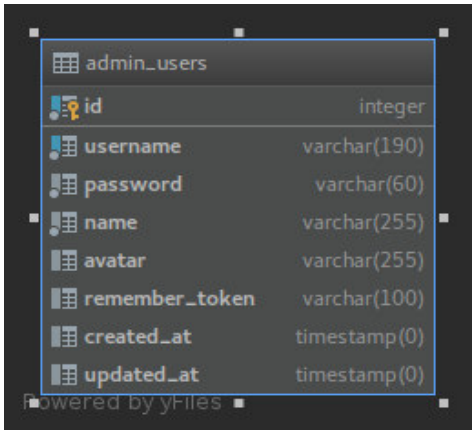


Рисунок 2.6 – Структура таблицы пользователей администраторов

Также для удобства отслеживания любого действия имеется таблица логирования каждого шага, каждого запроса, произошедшего на сервере, такая таблица необходима для прозрачности действии, чтобы не происходила фальсификация материалов, структура данной таблицы описана в рисунке 2.6.

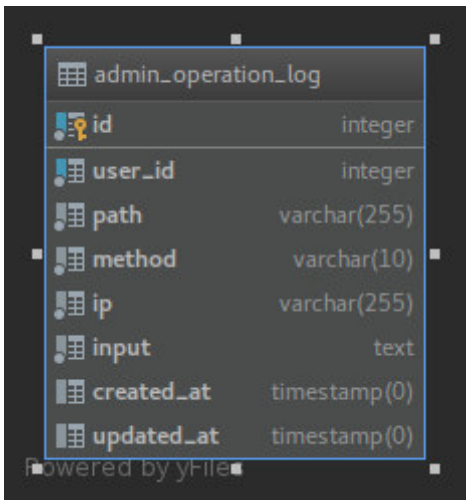


Рисунок 2.7 – Структура таблицы логирования

Для большой наглядности, было решено визуализировать видение базы данных, создать UML диаграмму для отображения связей и показать все

таблицы в графическом режиме схематично, диаграмма изображена на рисунке 2.7.

База данных состоит из двух составных информационных частей, таблицы административной панели, таблицы для мобильного приложения.

Для того чтобы создать базу данных в инструменте разработки серверных приложении фреймворка Laravel есть инструмент, называемый миграции, с помощью миграции, в пару строк консольных команд мы создаем базу данных.

```php
class CreateUsersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create( table: 'users', function (Blueprint $table) {
            $table->bigIncrements( column: 'id');
            $table->string( column: 'name');
            $table->string( column: 'email')->unique();
            $table->timestamp( column: 'email_verified_at')->nullable();
            $table->string( column: 'password');
            $table->rememberToken();
            $table->timestamps();
        });
    }
```

Рисунок 2.8 – Пример миграции

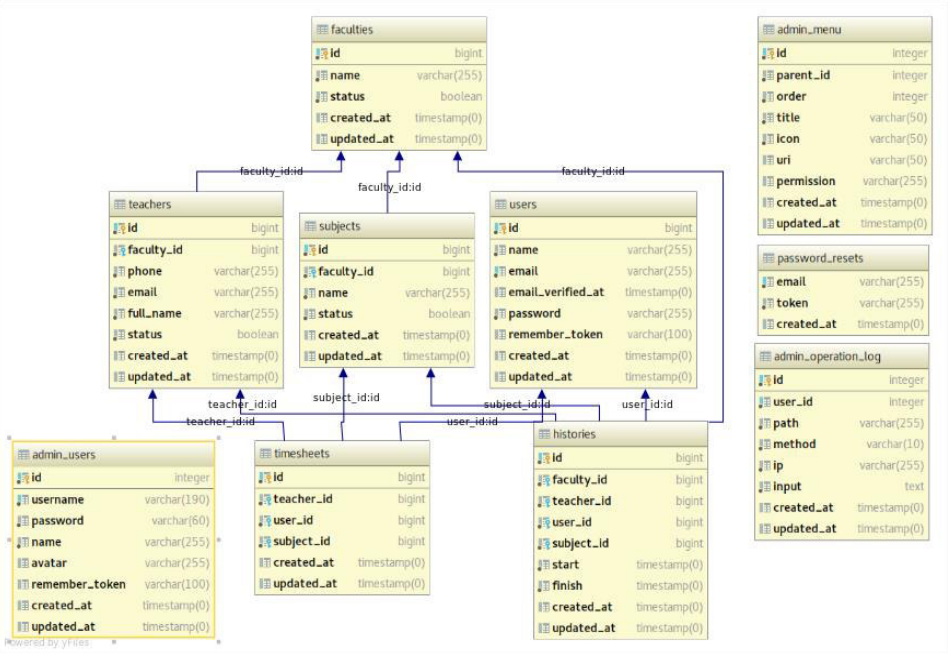


Рисунок 2.9 – Схематичная структура базы данных

2.4 Реализация мобильного приложения

2.4.1 Выбор поддерживаемых версии платформы Android

Для того чтобы охватить большое количество мобильных устройств, было решено провести анализ того, в каком процентном соотношении. В таблице 2.1 приведена история версий платформы Android.

Таблица 2.1 – История версий платформы Android

Кодовое имя	Версия	Дата выпуска
Froyo	2.2 – 2.2.3	May 20, 2010
Gingerbread	2.3 – 2.3.7	December 6, 2010
Honeycomb	3.0 – 3.2.6	February 22, 2011
Ice Cream Sandwich	4.0 – 4.0.4	October 18, 2011
Jelly Bean	4.1 – 4.3.1	July 9, 2012
KitKat	4.4 – 4.4W2	October 31, 2013
Lollipop	5.0 – 5.1.1	November 12, 2014
Marshmallow	6.0 – 6.0.1	October 5, 2015
Nougat	7.0 – 7.1	October 6, 2016
Oreo	8.0 – 8.1	August 18, 2017
Pie	9.0	September 12, 2018
Q	10.0	March 13, 2019

Для анализа использовались данные об относительном количестве устройств, работающих под управлением различных версий платформы Android. Данные собирались в течение семи дней до 7 мая 2019 года.

Некоторые устаревшие версии операционной системы начали увеличивать свою долю, хотя до этого, например, Android 2.3 Gingerbread удерживал свои показатели достаточно долго или снижал их. Последний выпуск Pie занял за восемь месяцев своего существования десятую часть Android-рынка.

Последний раз Google публиковала актуальное распределение 26 октября 2018 года, хотя изначально апдейт статистики выходил каждый месяц.

По информации Android Police, задержка длиной в несколько месяцев связана с потерей источника, который записывал диагностические данные об устройствах, обращающихся к сервисам Google Play. Теперь Google нашла другой способ собирать статистику.

Как и следовало ожидать самой распространенной оказалась ОС Android 8.0-8.1 Oreo с суммарной долей 28,3%. На втором месте оказалась ОС Android 7.0-7.1 Nougat с 19,2% всех совместимых устройств. Третьей по распространенности является ОС Android 6.0 Marshmallow с 16,9%, а четвертой Lollipop 5.0-5.1 — 14,5% рынка. Остальные версии распределились следующим образом: Android 9.0 Pie — 10.4%, KitKat 4.4 — 6,9%, Jelly Bean 4.1-4.3 — 3,2%, а Ice Cream Sandwich 4.0.3-4.0.4 и Gingerbread 2.3.3-2.3.7 — по 0,3% соответственно. В таблице 2.2 приведена информация о распространении версии Android.

Таблица 2.2 – Статистика версий платформы Android

Версия	Название	Год	Доля
2.3	Gingerbread	2010	0,3 %
4.0	Ice Cream Sandwich	2011	0,3 %
	2013	2011	0,5 %
4.4	KitKat	2013	6,9 %
5.0	Lollipop	2014	3 %
6.0	Marshmallow	2015	16,9 %
7.0	Nougat	2016	11,4 %
7.1		2016	7,8 %
8.0	Oreo	2017	12,9 %
8.1		2017	15,4 %
9.0	Pie	2018	10,4 %
10.0	Q	2019	< 0,1 %

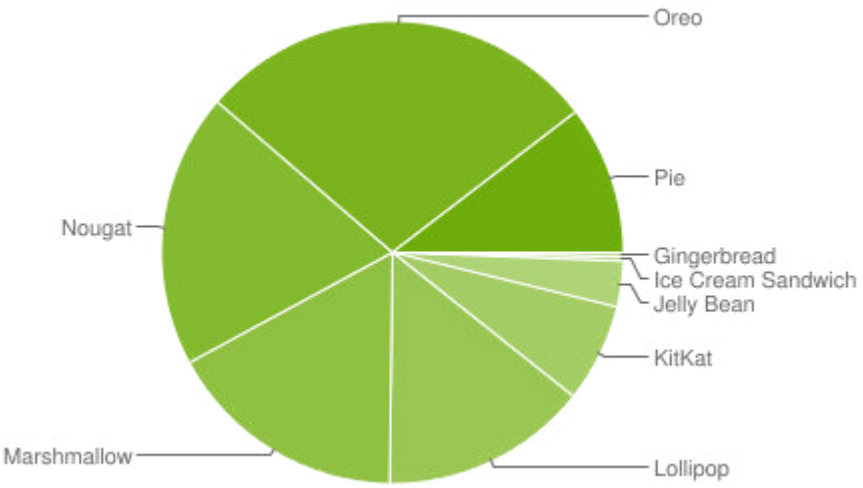


Рисунок 2.10 – Диаграмма относительного количества устройств на платформе Android

В результате проведенного анализа было решено поддерживать новые версии для охвата большого количества платформ Android.

2.4.2 Создание интерфейса

Для разработки мобильного интерфейса в Android используется расширяемый язык разметки (Extensible Markup Language), сокращенно XML. Язык разметки - искусственный язык, который использует набор аннотаций к

тексту, предоставляющий инструкции относительно структуры текста или его отражение.

Расширяемый язык разметки - предложенный консорциумом World Wide Web (W3C) стандарт построения языков разметки иерархически структурированных данных для обмена между различными приложениями, в частности, через Интернет. Существует, также, упрощенная версия языка разметки - SGML. В XML документ состоит из текстовых знаков, и пригоден к чтению человеком.

Стандарт XML определяет набор базовых лексических и синтаксических правил для построения языка описания информации путем применения простых тегов. Иными словами, предложенный стандарт определяет метаязык, на основе которого путем введения ограничений на структуру и содержание документов определяются специфические, предметно-ориентированные языки разметки данных [20]. В Eclipse есть два режима работы с xml файлами: графический и просмотр кода.

Для создания интерфейса следует выбрать вид разметки документа. На данном изображении видно, что на этом экране, как и на большинстве форм приложения, используется «RelativeLayout», в правой части экрана. В ходе разработки были использованы следующие виды разметки:

– RelativeLayout - для каждого элемента настраивается его положение относительно других элементов;

– LinearLayout - отражает View-элементы в виде одной строки (если установлен Horizontal) или одного столбца (если установлен Vertical);

– FrameLayout является самым простым типом разметки.

Обычно это пустое пространство на экране, который можно заполнить только дочерним объектом View или ViewGroup.

DrawerLayout – это компонент необходимый для быстрой реализации возможностей «выезжающего» левого меню. Этого компонента нет в списке стандартных, поэтому его вызов осуществляется посредством использования библиотеки поддержки «android.support.v4».

FrameLayout - это пустой контейнер для отображения различных видов отображения на одном экране. Необходимость этого объекта обусловлена тем, что, когда приложение будет расширять свой функционал чтобы каждый раз не пересоздать экран с левым меню, все отображения имеющих доступ к меню будут помещены в этот объект.

Все остальные экраны приложения тем или иным образом включают в себя компонент «ListView» - это компонент для формирования списков. Рассмотрим основные компоненты и их ключевые характеристики:

– TextView - служит для вывода текста;

– EditText - служит для ввода текста;

– AutoCompleteTextView - текстовое поле, в котором есть встроенные ресурсы для поиска и автоматического заполнения текста (массив данных подключается программно);

– Button – кнопка для выполнения управляющих действий;

– ProgressBar - отражает процесс загрузки;

– ImageView - служит для вывода изображений;

– Spinner - выпадающий список.

Для формирования интерфейса кроме папки «layout» в проекте есть папки с названием «values-xx», где xx - это вспомогательная указание, например код страны или уровень API. В папке values обязательно должен присутствовать файл с названием «strings.xml», в нем сохраняются операционные ресурсы, используемые в приложении. В частности, в приложении есть поддержка русского и английского языков. Языком по умолчанию является английский.

3 Разработка интерфейса и логики мобильного приложения

3.1 Реализация интерфейса

Для отображения всех активностей в приложении было решено реализовать простой экран (Activity) для отображения группы текстовых меток (TextView) в виде списка (ListView). После этого добавить на экран кнопку, которая показывает пользователю новый экран, в котором можно ввести новую задачу в текстовое поле (EditText) и нажать кнопку сохранить. В дальнейшем этот экран стал использоваться так же для изменения уже созданной задачи. После добавления функционала указания группы задачи на экране добавился выпадающий список (Spinner).

Для разработки основного функционала приложения была выбрана архитектура MVC (Model – View – Controller), как наиболее подходящая для этой задачи. В качестве модели (Model) для приложения выступает несколько классов User обозначающий студента, Teacher – учитель, Timesheet – доска расписании, которые реализованы как Singleton. В данных классах хранится информация о списках задач, а именно список текущих задач, список завершенных задач и список удаленных задач, посещение студента, расписание занятии. Для доступа к спискам студентов в этом классе используются внутренние ключи. Эти ключи уникальны в пределах одного экземпляра класса. Соответственно они уникальны для всего приложения, т.к. класс Singleton. Элементами списков являются экземпляры класса UserModel. В этом классе хранится все информация, необходимая для жизненного цикла задачи, а именно: − идентификатор задачи в базе данных сервера; − временная отметка последнего изменения задачи на сервере; − отметка последнего изменения задачи в мобильном приложении; − экземпляр класса NotesStudentData. Класс NotesStudentData представляет собой описание информации, характеризующей задачу, а именно:

− текст;

− название группы, к которой относится занятие;

− приоритет расписания;

− флаг, определяющий задачу как выполненную.

Метод getView в данном классе возвращает представление элемента списка студентов. В зависимости от режима отображения (отображать текущего студента, отображать конкретную группу студентов или отображать завершенные задачи) возвращается соответствующий элемент разметки, с текстовой меткой (TextView) и изображением, выступающим в качестве кнопки (Button).

Для переключения отображения списков, открытия настроек, выхода из приложения было решено использовать меню, которое располагается в ActionBarDrower. Это элемент интерфейса, который вызывается при нажатии на изображение меню в верхнем левом углу приложения или при оттягивании списка меню от левой границы экрана. Заполняется меню при помощи

адаптера, который похож на адаптер для списка задач, но с меньшим количеством методов.

Для того чтобы реализовать интерфейс окна отображения студентов применяется TextView в обертке RecyclerView который может отображать несколько текстовых меток в цикле при подгрузке данных из базы данных RecyclerView динамично изменяет свои параметры чтобы вместить весь пришедший с сервера ответ.

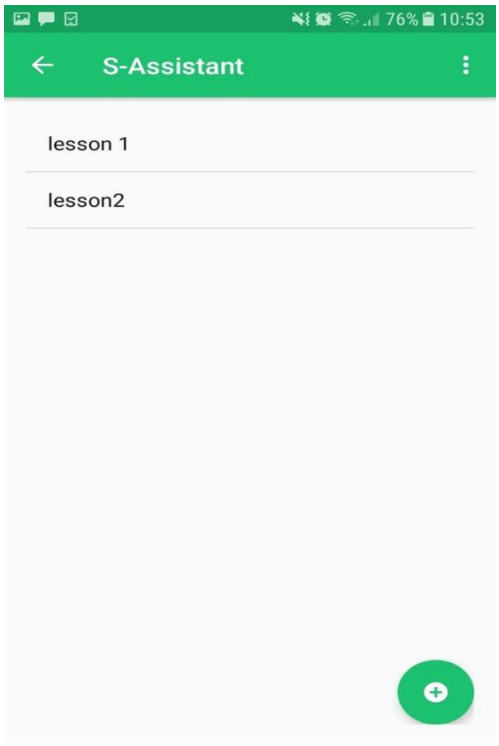


Рисунок 3.1 – Экран для расписания занятии

При запуске приложения на мобильном устройстве происходит проверка авторизации пользователя. Для хранения параметров авторизации используется SharedPreferences. В хранилище записывается token, который приложение получает в результате успешной аутентификации. Для этих целей было сделано несколько экранов (Activity), для входа в сервис, регистрации, восстановления пароля и изменения пароля соответственно. Изображение одного из этих экранов приведено на рисунке 2.10.

Для этих процессов будет использоваться JWT токены. Посредством JWT токена можно гарантировать целостность данных и сессии конкретного пользователя, процесс происходит следующим образом:

Пользователь запускает приложение, перед ним открывается окно авторизации, с паролем и никнеймом, он заполняет эти данные, в этот момент срабатывает EventListener скрипт, ожидающий клика на переход, эти данные отправляются на сервер по маршруту заранее предопределенный для запросов авторизации, пришедшие данные с мобильного приложения, обрабатываются на сервере контроллером авторизации, где пришедшие данные сравниваются с

записями из базы данных, при нахождении совпадения, эти данные связываются указателем с помощью JWT токена, цифровой идентификатор пользователя, именно этот токен отправится в мобильное приложение, в дальнейшем любой запрос пользователя будет отправляться на сервер с этим токеном, на случай если токен не отправится.

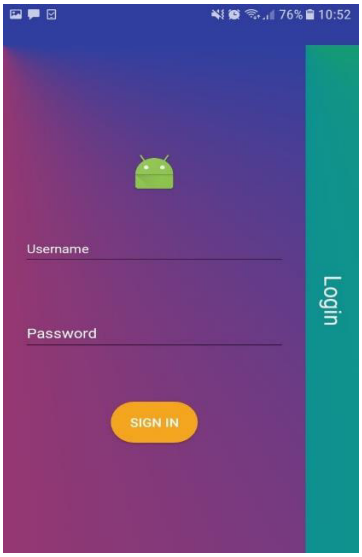


Рисунок 3.2 – Экран авторизации преподавателя

На сервере есть промежуточный уровень называемый Middleware, который принимает все запросы идущие по маршруту до контроллеров, как правило в промежуточных уровнях проверяется, имеет ли право пользователь право на ресурс куда он отправил запрос, в случае успешности запрос идет дальше к контроллеру, который формирует ответ, в случае не успешности запроса то есть если каким то образом , токен не записался в заголовки запроса, пользователю придет ответ, что он не может выполнить данный запрос, прося перейти в окно авторизации.

Схематично паттерн проектирования в мобильном приложении представляет из себя MVP (Model-View-Presenter), это обеспечивает целостность данных, самой актуальной проблемой мобильных приложении, является жизненный цикл активности, ведь каждое Activity при случае смены ориентации экрана или при переходе из активности в другую активность теряет данные, с которыми он работал, эти проблемы решает Presenter сохраняя данные при каждом событий в своих переменных класса. Также логика отправки данных на сервер содержится в Presenter.

Модель — это класс, описывающий сущности, работающие в этом приложении, то есть следующие классы являются моделями:

- teacher model;
- student_model;
- schedule_model;
- setting_model.

Все эти модели описывают свойства сущностей в данной информационной системе

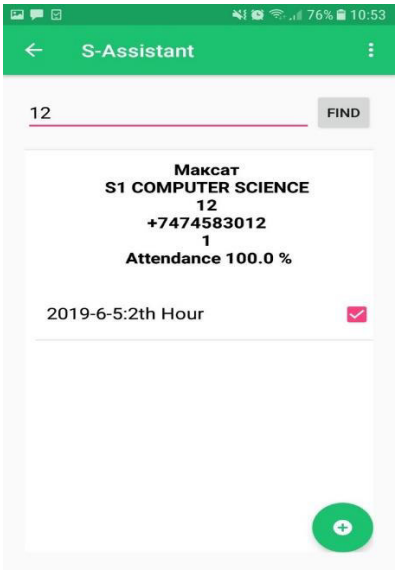


Рисунок 3.3 – Экран для действия со студентом

В окне входа происходит обращение мобильного приложения к серверу, в результате чего на сервере генерируется уникальный токен который присваивается к преподавателю, который активировал запрос, затем весь сеанс, что проводит преподаватель в приложении все запросы на сервер выполняются с помощью этого токена, поэтому токен можно называть цифровым идентификатором преподавателя в сервере.

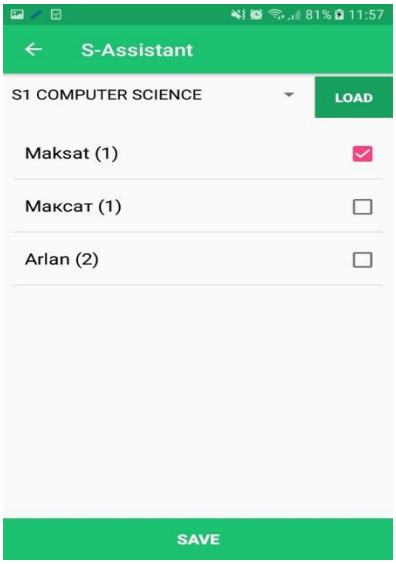


Рисунок 3.4 – Экран фиксации посещаемости

После того как преподаватель создал сущность занятия, он переходит к окну фиксации посещаемости перед ним по нажатию кнопки load загружается

список доступных ему студентов, из которых он может выбрать тех, кто присутствует на данном занятии, и информация отправится на сервер.

На главном экране приложения у преподавателя имеется 5 вкладок, каждый из которых ведет на свой экран активности, а также имеется вкладка настроек, где можно очищать данные.

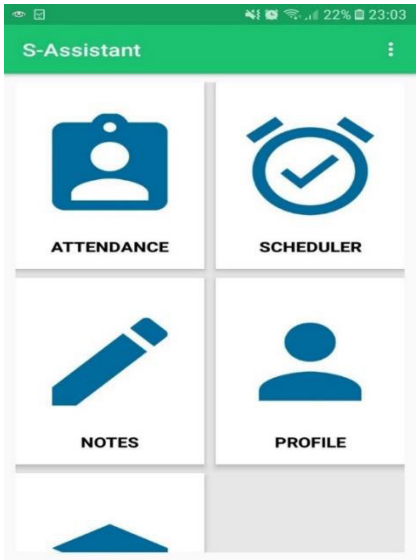


Рисунок 3.5 – Главный экран

Переход между окнами активностей происходит с помощью стрелок назад, на странице студента преподаватель может найти студента по заданному им самим номеру идентификации, также имеет возможность отредактировать информацию студента, может добавить студента, собственноручно заполняя его данные.

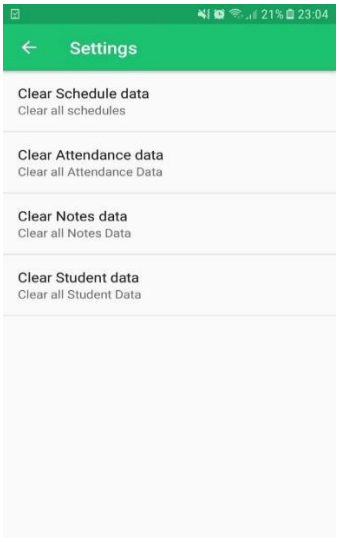


Рисунок 3.6 – Экран настроек

3.2 Описание программного продукта

3.2.1 Общие сведения

Рассмотрим более подробно модули, входящие в состав проекта:
- attendanceActivity.java
- noteActivity.java;
- scheduleActivity.java;
- profileAcitivity.java;
- baseActivity.java;
- databaseHandler.java;
- userContrloller.php;
- teacherController.php;
- timeSheetController.php;
- facultyController.php.

Программа реализована в программных средах разработок «PHP Storm», «Android Studio».

3.3.2 Функциональное назначение

Данный программный продукт служит для ввода и хранения информации о студентах их посещении занятии, сохранении информации о расписаниях занятии, которые проводились преподавателями, корректировка информации о посещении студентов, внесение дополнительной информации или удаление ненужной информации, занесение в архив отработанных данных.

Функционально, приложение состоит из приведенных ниже модулей (активностей). Активность является схематичной формой представления Android приложений. Каждый экран пользовательского интерфейса представлен классом Activity и, по сути, является отдельной формой приложения. Android-приложение способно состоять из нескольких активностей и может переключаться между ними во время выполнения приложения.

Модуль AttendanceActivity.java выводит на экран монитора диалоговое окно для создания заметки, в котором непосредственно заносится информация о посещении занятии студентами, а именно указывается какое было занятие, когда оно проводилось, и посещал ли его студент.

Модуль NoteActivity.java выводит на экран окно допольнительной логики, то есть хранение личной информации преподавателя, по логике действия очень похож на стикеры коими люди пользуются для сохранения какой-либо детальной информации в тезисах авторизации пользователя. Авторизация выполняется путем ввода имени пользователя и пароля. Программа хранит имена пользователей и пароли в закодированном виде в файле.

Модуль ScheduleActivity.java отображает окно создания напоминания, то есть преподаватель может создавать события заранее, с учетом того, чтобы программа напомнила ему о событии, в день его активности. С него можно перейти в окно конструктора посещении, в окно профиля студента, чтобы отследить его посещение.

Модуль ProfileActivity.java выводит на экран окно создания учетной записи студента с личной информацией, которая доступна преподавателю. В этом модуле преподаватель может добавлять новых студентов, редактировать информацию текущих студентов, удалить ненужную информацию, удалить студента, который по каким-либо причинам больше не является студентом, здесь происходит процесс работы с базой данных на сервере, то есть все, что принимает форма данного окна, отправляется на сервер, где с этой информацией работает UserController.php в котором, содержатся скрипты для обработки данных поступающих от мобильного приложения.

Модуль BaseActivity.java содержит всю логику, маршруты, местную базу данных для хранения промежуточных результатов работы, содержит методы авторизации преподавателя, изменения его пароля, выводит на экран окно авторизации. В этом окне совершается вход в приложение, вход защищен в это приложение, поэтому посторонние лица работать с этим приложением не смогут.

Модуль DatabaseHandler.java не содержит информацию касательно отображения данных, здесь происходит работа с данными поступающие от фронтовой системы, описывается логика работы с этими данными, куда что отправить, куда что записать, описывается порядок работы с хранилищами, с которыми оперирует мобильное приложение на телефоне, самое главное описывается логика взаимодействия базы данных на сервере с мобильным приложением, в которых и содержится вся основная информация. Реализован объектно-ориентированный подход к подключению к базе данных, с помощью фильтрации поступающих данных, которые затем отправятся в таблицы на запись в сервер.

Модуль UserController контроллер, расположенный в приложении «Laravel» в котором находится административная панель управления мобильным приложением, где находятся API, располагает основными методами и функциями для работы с данными студентов, в административном панели управления располагается все действия доступные со студентами, то есть удаление, чтение, создание, обновление, такой же ряд действий, совершаются в мобильном приложении преподавателем. Модуль не располагает отображением данных, оно лишь работает с ними, перенаправляя его в требуемые инстанции.

Модуль TeacherController контроллер серверного приложения для формирования API, также не обладает отображением данных, работает исключительно над формированием ответа, для того чтобы отправлять данные для отображения в мобильное приложение. Формирует переменную класса, куда заносится модель TeacherModel, где описываются свойства данной

сущности, все эти поля запрашиваются из базы данных, затем формируется JSON который отправляется в мобильное приложение.

Модуль TimeSheetController, также является контроллером серверного приложения, формирующий таблицу посещений, так как это таблица обладает данными из нескольких простых таблиц для этой таблицы потребуется нормализация, с помощью составных данных формируется таблица, также создается вспомогательная таблица, которая подвязывается к основной таблице, содержит важную информацию такого рода как:

- Информация кем была создана заметка о посещаемости;
- Информация какое занятие проводилось;
- Время и дата проведения данного события;
- Студенты, присутствующие на этом занятии;
- Студенты, которых не было на этом занятии.

Содержит ту информацию, которая именно и нужна для функционирования. Активность TableView содержит список объектов из выбранных категорий с указаниями расстояния до них.

Активность посылает запрос к базе данных о выбранных категориях и на основании полученной информации формирует новый SQL-запрос для получения всех интересующих объектов. При выборе любого объекта запускается новая активность TabWidget, в качестве параметров которой передается идентификатор объекта

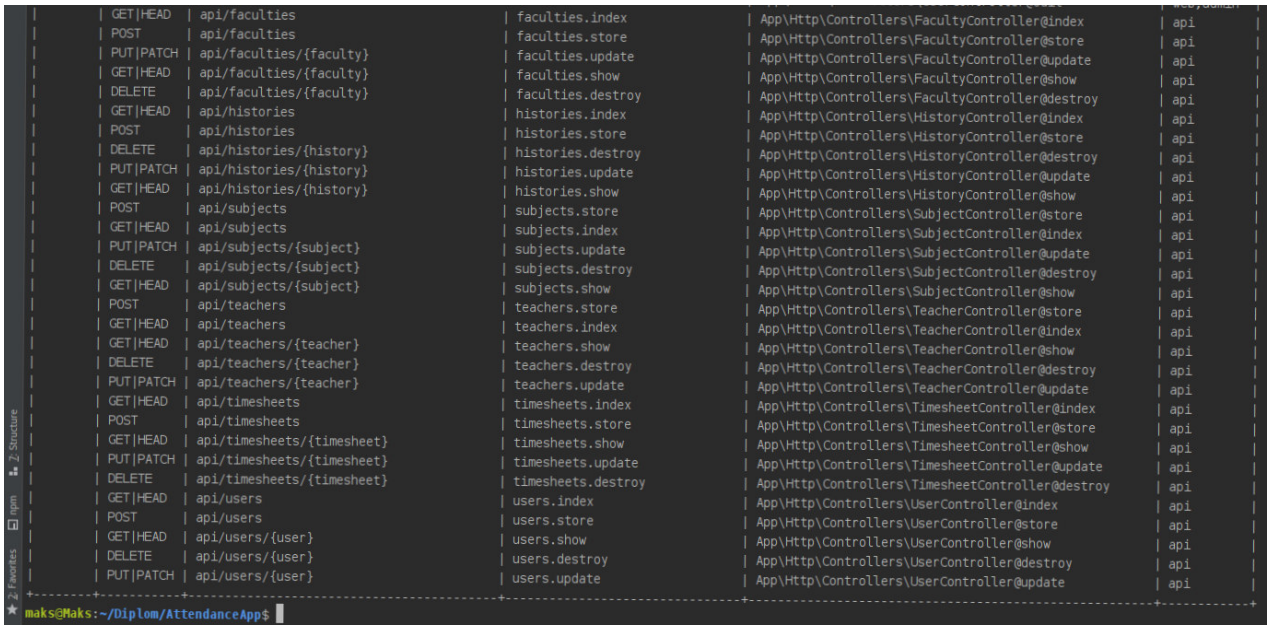


Рисунок 3.7 – Список всех маршрутов мобильного приложения

3.3.3 Описание логической структуры

Логическая структура модуля AttendanceActivity.java с привязкой к строкам текста приведено в таблице 3.1.

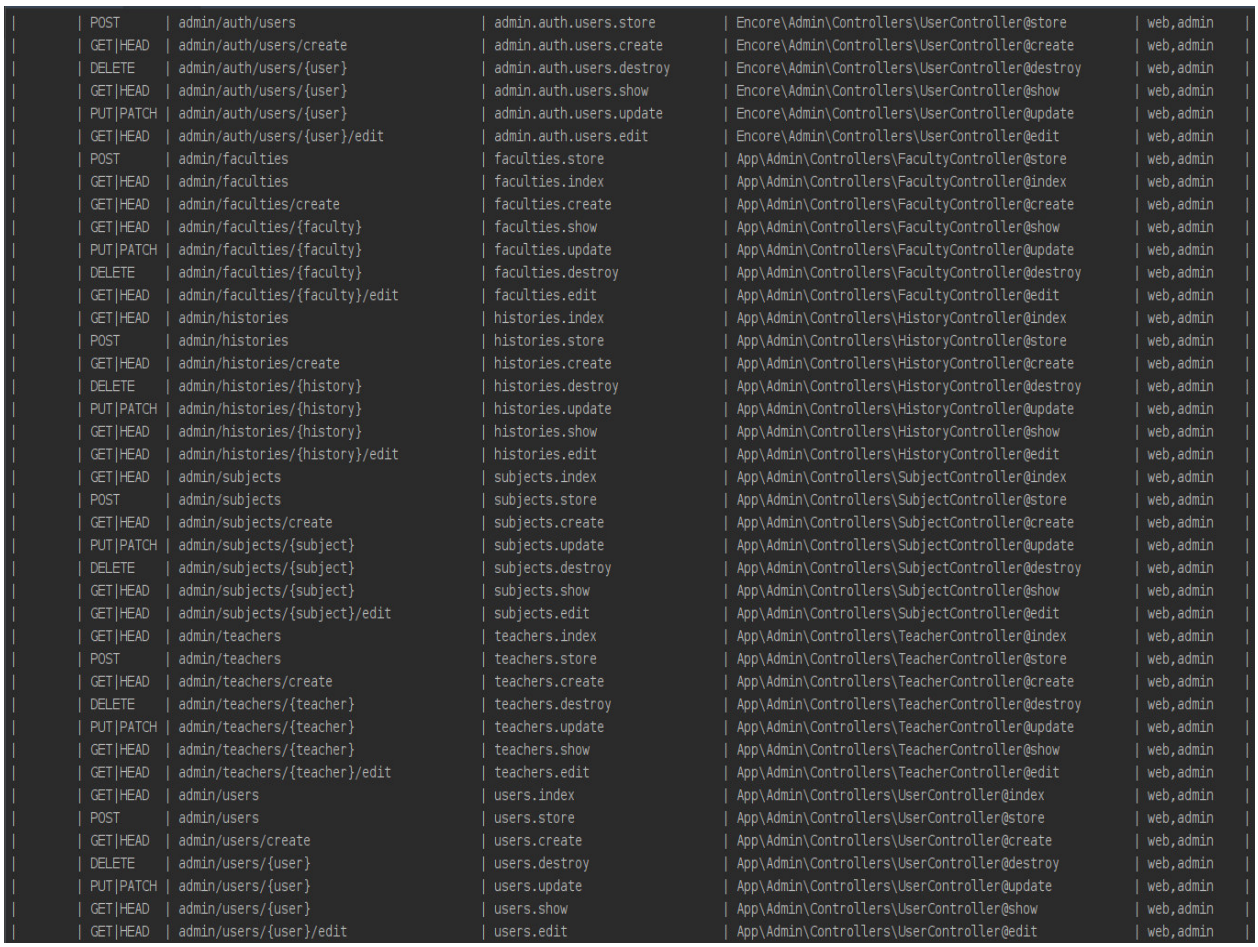
Таблица 3.1 – Логическая структура модуля AttendanceActivity.java

№ строки	Описание
1 – 24	Данные о импорте пакет которые импортированы в этот класс
25 – 28	Метод переопределения методов, статического позднего связывания, для обеспечения синхронности действия класса DatabaseHandler с AttendanceActivity.
29 – 33	Процедура вызова окна создания заметки посещаемости
34 – 38	Процедура вызова окна информации о студентах
39	Конец модуля

Логическая структура модуля NoteActivity.java с привязкой к строкам текста приведено в таблице 3.2.

Таблица 3.2 – Логическая структура модуля NoteActivity.java

№ строки	Описание
1 – 33	Модуль дополнительной логики для создания преподавателем заметок касательно информации о посещаемости
34 – 71	Функция, производящая дешифрование данных методом трисемуса
72 – 75	Процедура выхода из скрипта создания заметок
76 – 79	Процедура перехода на главное окно
80 – 82	Начало процедуры привязки заметки к определенной записи посещаемости.
83 – 93	Чтение данных из базы данных, хранящих информацию студентов
94 – 126	Сравнение информации пользователей и паролей с вводимыми пользователем и действия, производимые в зависимости от правильности введенных данных
127 – 131	Процедура, присоединяющая информацию студента к заметкам.



	POST	admin/auth/users	admin.auth.users.store	Encore\Admin\Controllers\UserController@store	web,admin
	GET|HEAD	admin/auth/users/create	admin.auth.users.create	Encore\Admin\Controllers\UserController@create	web,admin
	DELETE	admin/auth/users/{user}	admin.auth.users.destroy	Encore\Admin\Controllers\UserController@destroy	web,admin
	GET|HEAD	admin/auth/users/{user}	admin.auth.users.show	Encore\Admin\Controllers\UserController@show	web,admin
	PUT|PATCH	admin/auth/users/{user}	admin.auth.users.update	Encore\Admin\Controllers\UserController@update	web,admin
	GET|HEAD	admin/auth/users/{user}/edit	admin.auth.users.edit	Encore\Admin\Controllers\UserController@edit	web,admin
	POST	admin/faculties	faculties.store	App\Admin\Controllers\FacultyController@store	web,admin
	GET|HEAD	admin/faculties	faculties.index	App\Admin\Controllers\FacultyController@index	web,admin
	GET|HEAD	admin/faculties/create	faculties.create	App\Admin\Controllers\FacultyController@create	web,admin
	GET|HEAD	admin/faculties/{faculty}	faculties.show	App\Admin\Controllers\FacultyController@show	web,admin
	PUT|PATCH	admin/faculties/{faculty}	faculties.update	App\Admin\Controllers\FacultyController@update	web,admin
	DELETE	admin/faculties/{faculty}	faculties.destroy	App\Admin\Controllers\FacultyController@destroy	web,admin
	GET|HEAD	admin/faculties/{faculty}/edit	faculties.edit	App\Admin\Controllers\FacultyController@edit	web,admin
	GET|HEAD	admin/histories	histories.index	App\Admin\Controllers\HistoryController@index	web,admin
	POST	admin/histories	histories.store	App\Admin\Controllers\HistoryController@store	web,admin
	GET|HEAD	admin/histories/create	histories.create	App\Admin\Controllers\HistoryController@create	web,admin
	DELETE	admin/histories/{history}	histories.destroy	App\Admin\Controllers\HistoryController@destroy	web,admin
	PUT|PATCH	admin/histories/{history}	histories.update	App\Admin\Controllers\HistoryController@update	web,admin
	GET|HEAD	admin/histories/{history}	histories.show	App\Admin\Controllers\HistoryController@show	web,admin
	GET|HEAD	admin/histories/{history}/edit	histories.edit	App\Admin\Controllers\HistoryController@edit	web,admin
	GET|HEAD	admin/subjects	subjects.index	App\Admin\Controllers\SubjectController@index	web,admin
	POST	admin/subjects	subjects.store	App\Admin\Controllers\SubjectController@store	web,admin
	GET|HEAD	admin/subjects/create	subjects.create	App\Admin\Controllers\SubjectController@create	web,admin
	PUT|PATCH	admin/subjects/{subject}	subjects.update	App\Admin\Controllers\SubjectController@update	web,admin
	DELETE	admin/subjects/{subject}	subjects.destroy	App\Admin\Controllers\SubjectController@destroy	web,admin
	GET|HEAD	admin/subjects/{subject}	subjects.show	App\Admin\Controllers\SubjectController@show	web,admin
	GET|HEAD	admin/subjects/{subject}/edit	subjects.edit	App\Admin\Controllers\SubjectController@edit	web,admin
	GET|HEAD	admin/teachers	teachers.index	App\Admin\Controllers\TeacherController@index	web,admin
	POST	admin/teachers	teachers.store	App\Admin\Controllers\TeacherController@store	web,admin
	GET|HEAD	admin/teachers/create	teachers.create	App\Admin\Controllers\TeacherController@create	web,admin
	DELETE	admin/teachers/{teacher}	teachers.destroy	App\Admin\Controllers\TeacherController@destroy	web,admin
	PUT|PATCH	admin/teachers/{teacher}	teachers.update	App\Admin\Controllers\TeacherController@update	web,admin
	GET|HEAD	admin/teachers/{teacher}	teachers.show	App\Admin\Controllers\TeacherController@show	web,admin
	GET|HEAD	admin/teachers/{teacher}/edit	teachers.edit	App\Admin\Controllers\TeacherController@edit	web,admin
	GET|HEAD	admin/users	users.index	App\Admin\Controllers\UserController@index	web,admin
	POST	admin/users	users.store	App\Admin\Controllers\UserController@store	web,admin
	GET|HEAD	admin/users/create	users.create	App\Admin\Controllers\UserController@create	web,admin
	DELETE	admin/users/{user}	users.destroy	App\Admin\Controllers\UserController@destroy	web,admin
	PUT|PATCH	admin/users/{user}	users.update	App\Admin\Controllers\UserController@update	web,admin
	GET|HEAD	admin/users/{user}	users.show	App\Admin\Controllers\UserController@show	web,admin
	GET|HEAD	admin/users/{user}/edit	users.edit	App\Admin\Controllers\UserController@edit	web,admin

Рисунок 3.8 – Список всех маршрутов административной части приложения

Логическая структура модуля ScheduleActivity.java с привязкой к строкам текста приведено в таблице 3.3.

Таблица 3.3 – Логическая структура модуля ScdeuleActivity.java

№ строки	Описание
1 – 36	Заголовок основного модуля программы и описание используемых системных модулей, используемых компонент, применяемых процедур-обработчиков, констант и переменных
37 – 40	Процедура вызова окна выхода из программы
41 – 47	Процедура вызова окна авторизации пользователя
48 – 51	Процедура вызова окна конструктора заметок посещаемости
52 – 55	Процедура вызова окна создания студента, взаимодействие, привязка к событию
56 – 60	Процедура закрывания программы при закрытии текущего окна
61 – 65	Процедура позднего статического связывания для привязки дополнительной логики из DatabaseHandler
65 – 69	Процедура проверки совершенной работы методами синхронности из DatabaseHandler

Логическая структура модуля ProfileActivity.java с привязкой к строкам текста приведено в таблице 3.4.

Таблица 3.4 – Логическая структура модуля ProfileActivity.java

№ строки	Описание
1 – 89	Заголовок основного модуля программы и описание используемых системных модулей, используемых компонент, применяемых процедур-обработчиков, констант и переменных
90 – 93	Процедура вызова окна "Редактирование информации"
94 – 98	Выход из окна редактирования
99 – 102	Процедура вызова серверных контроллеров для выполнения операции с базой данных
103 – 106	Процедура вызова окна "Профиль студента"
107 – 110	Процедура вызова окна "Удаление информации"
111 – 114	Процедура вызова окна "Статусы студента"

Логическая структура модуля UserController с привязкой к строкам текста приведено в таблице 3.5.

Таблица 3.5 – Логическая структура модуля UserController.java

№ строки	Описание
1 – 46	Заголовок основного модуля программы и описание используемых системных модулей, используемых компонент, применяемых процедур-обработчиков, констант и переменных
47 – 50	Процедура закрытия окна поиска
51 – 55	Процедура выбора исходных для создания SQL-запроса данных из списка, составляемого при переходе на окно поиска из предыдущего окна
56 – 63	Процедуры переключнеия в режимы поиска и сортировки данных
64 – 66	Начало процедуры поиска, описание локальных переменных
67 – 69	Проверка начальных для поиска условий, без которых будет невозможно построение SQL-запроса
70 – 82	Построение SQL-запроса для поиска на Form10
83 – 92	Построение SQL-запроса для поиска на Form13

При загрузке приложения основная активность обращается к базе данных, находящейся на диске. Если на диске база данных не обнаружена, то создается пустая база данных и происходит её обновление через интернет. Таким образом, пользователь может выбрать интересующие его вкладки объектов. При выборе категории в меню загружается активность «Категория», которая обращается к базе данных и, считав возможные категории, отображает их в списке. При выборе категории загружается активность.

С помощью механизма передачи параметров между активностями ей передается информация о категории. Эта активность тоже обращается к базе данных (формируя запрос) и выводит все подкатегории. При выборе любой подкатегории формируется запрос на изменение базы данных. Таким образом, база данных всегда будет знать об интересующих подкатегориях и их не придется вводить после каждого перезапуска приложения. Выход из этих активностей осуществляется с помощью кнопки “Назад”. Активность «Таблица» формирует запрос к базе данных на основании выбранных категорий и выводит список всех объектов в табличной форме.

Логическая структура модуля TimeSheetController с привязкой к строкам текста приведено в таблице 3.6.

Таблица 3.6 – Логическая структура модуля TimeSheetController

№ строки	Описание
1 – 46	Заголовок основного модуля программы и описание используемых системных модулей, используемых компонент, применяемых процедур-обработчиков, констант и переменных
47 – 50	Процедура закрытия окна поиска
51 – 55	Процедура выбора исходных для создания SQL-запроса данных из списка, составляемого при переходе на окно поиска из предыдущего окна
56 – 63	Процедуры переключнеия в режимы поиска и сортировки данных
64 – 66	Начало процедуры поиска, описание локальных переменных
67 – 69	Проверка начальных для поиска условий, без которых будет невозможно построение SQL-запроса
70 – 82	Построение SQL-запроса для поиска на Form10
83 – 92	Построение SQL-запроса для поиска на Form13
93 – 102	Построение SQL-запроса для поиска на Form16
103 – 115	Построение SQL-запроса для поиска на Form37
116 – 129	Построение SQL-запроса для поиска на Form46
130 – 136	Построение SQL-запроса для поиска на Form11
137 – 139	Конец процедуры построения SQL-запросов для поиска
140 – 143	Процедура отображения SQL-запроса для поиска в удобном виде
144 – 150	Процедура отчистки значений фильтра для поиска или сортировки

3.3.4 Используемые технические средства

В ходе разработки ПО были использованы следующие технические средства: персональный компьютер Intel Core i-5 2800 МГц, 8096 Mb ОЗУ, монитор Samsung 795MB, также при работе с данной системой используется лазерный принтер HP LazerJet 1020.

3.3.5 Входные и выходные данные

Входными данными будут являться: имя пользователя и его пароль, а также сведения о посещаемости, студентах, (более подробно состав входных и выходных данных приведен в постановке задачи).

3.4 Взаимодействие с сервером

Для взаимодействия с сетью было решено использовать библиотеку Retrofit, как наиболее популярное и удобное решение для отправки и обработки асинхронных запросов. Для авторизации, аутентификации и синхронизации с сервером было реализовано несколько классов, реализующих интерфейс Callback из библиотеки Retrofit. Это позволило обрабатывать результаты выполнения запросов к серверу. Для отправки запросов синхронизации используется вспомогательный класс DatabaseHandler, который взаимодействует с моделью приложения и реализацией интерфейса REST для синхронизации задач с сервером.

Для сохранения корректного порядка синхронизации используются состояния. Для этого реализован класс SyncState, который содержит все возможные состояния, которые переключаются в течение выполнения запросов синхронизации. Классы для обработки ответов от сервера взаимодействуют с моделью приложения и с базой данных для сохранения пришедших изменений.

Для возможности выполнения синхронизации в фоновом режиме в Android приложении был реализован сервис. Сервис представляет собой фоновый процесс, который является частью приложения и запускается при успешной авторизации при первом запуске приложения и при загрузке мобильного устройства. Сервис с интервалом в 10 секунд синхронизирует данные модели с сервером при помощи обращений к классу Если в момент выполнения сервиса есть подключение к сети Интернет, то синхронизация запускается. Если в данный момент запущен главный экран приложения со списком задач, то сервис сообщает представлению (View) об изменении модели, что позволяет обновлять списки задач после выполнения синхронизации. Для возможности отключения синхронизации в настройках мобильного приложения предусмотрена соответствующая опция.

Для отображения настроек приложения был реализован экран (Activity), расширяющий класс SettingActivity. Это позволило хранить настройки приложения в SharedPreferences. Листинг некоторых классов мобильного приложения представлены в приложении Б. Для разработки интерфейса мобильного приложения было решено придерживаться правил создания интерфейсов Material Design. При этом нужно было поддерживать старые версии платформы, где эти правила поддерживаются не в полной мере. Для этого использовалась библиотека android.support.v7. Эта библиотека

позволила создать ActionBarDrower и прочие элементы интерфейса, характерные для Material Design.

При необходимости получения дальнейшей информации загружается следующая активность, куда передается ID номер выбранного объекта. Активность представляет собой TabView, в одном из табов которого находится карта, а в других – текстовые поля. Активность запрашивает из базы данных информацию об объекте и заполняет текстовые поля.

3.5 Административная панель управления

Административная панель управления построена на базе сервера с помощью приложения.

Функциональное предназначение:

Администрациитивное управление данными будет происходить из браузерного веб-приложения, располагающая всеми инструментами для того чтобы администрировать данные с которыми работает мобильное приложение

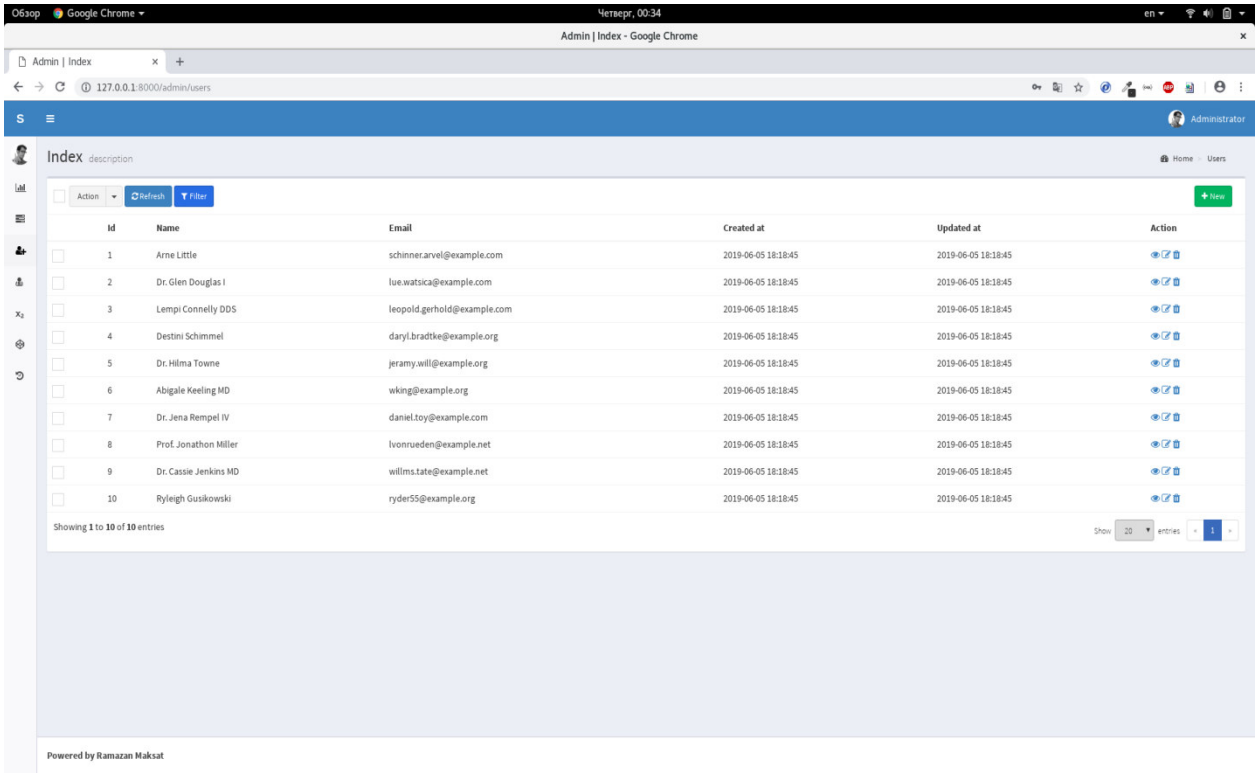


Рисунок 3.8 – Интерфейс административной панели управления

3.6 Тестирование

В процессе разработки приложения производилось поэтапное тестирование с целью выявления программных ошибок и несоответствий ТЗ (техническому заданию). Для этого нами были созданы эмуляторы смартфона и планшета с разными диагоналями экрана для разных версий Android.

Тестируемый программный продукт последовательно запускался на этих эмуляторах, его поведение анализировалось, и при необходимости по результатам анализа вносились изменения в код [12]. Для тестирования отдельных модулей работы с базой данных в текст программы были внесены специальные функции, позволяющие анализировать базу данных и, при подозрении на ошибку, выводящие сообщение в системный журнал. Они также известны как юнит-тесты. Например, при изменениях в базе данных проводилась проверка целостности базы данных (проверка на соответствие ключей – индексам), после чего при необходимости выводилось сообщение в системный log. Были проведены приведенные ниже тесты.

- Каждая активность была подвергнута юнит-тестированию с целью выявления ошибок, вызванных несоответствием ожидаемых и полученных параметров. Для этого для каждой активности был создан специальный юнит-класс, посылающий в активность различные верные и неверные параметры. При аномальном поведении активности или ее сбое, мною анализировалось поведение, и ошибка исправлялась.
- В базу данных намеренно вносились недопустимые данные в соответствующие поля, которые могли быть неверно интерпретированы программой. Затем мною анализировалось поведение активности во время обработки недопустимых данных.
- Приложение было запущено на устройствах, работающих под управлением разных версий Android с целью выявления особенностей работы приложения, запущенного в разных операционных системах.
- После завершения цикла разработки, программный продукт тестировался на реальных устройствах. По результатам тестирования была добавлена виртуальная кнопка «Меню» для устройств, не имеющих аппаратных кнопок.

3.7 Контрольный пример

В этой главе пройдет элементарная демонстрация функционала по приложению, начиная от сборки apk файла заканчивая выходом из приложения.

Перед тем как запустить приложение на смартфоне, необходимо убедиться в том, что в вашем смартфоне включен режим разработчика, естественно, если вы разработчик, для конечных пользователей будет существовать приложение на PlayMarket.

Режим разработчика на моем смартфоне включается нестандартно, необходимо зайти в настройки, затем выбрать сведения о телефоне, где необходимо семь раз кликнуть по своему номеру сборки устройства, смартфон должен вас уведомить о включенном параметре разработчика.

После того как вы включили режим разработчика необходимо выйти назад, и вы увидите вкладку «Параметры разработчика», где вы можете настроить детально и под ваш настрой ваш смартфон.

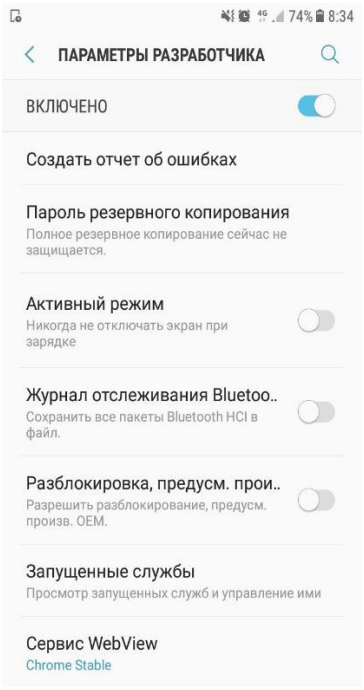


Рисунок 3.9 – Вкладка разработчика

Затем открываем среду разработки «Android Studio», ждем, когда про индексируется проект с мобильным приложением, запускаем его с помощью сборщика «Gradle», когда сборщик все установит, на вашем подсоединенном usb кабелем телефоне откроется первое диалоговое окно мобильного приложения.

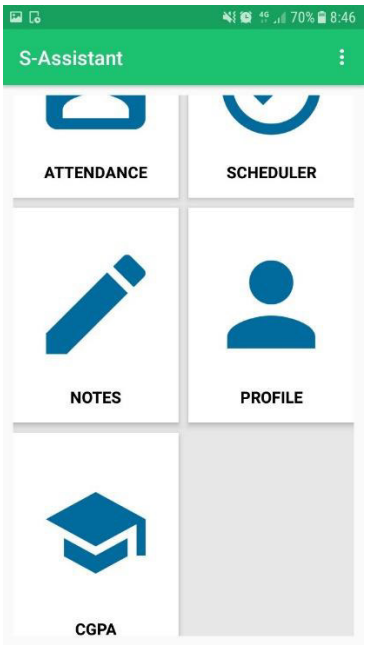


Рисунок 3.10 – Стартовая страница приложения

На стартовой странице вы увидите пять вкладок, пять функционалов данного приложения:

- фрагмент посещении;
- фрагмент напоминании;
- фрагмент добавки виртуальных стиков;
- фрагмент студентов;
- фрагмент GPA калькулятора.

Для начала работы преподавателю необходимо добавить студентов, так как это основная сущность двигающая процесс функциональности в данном приложении.

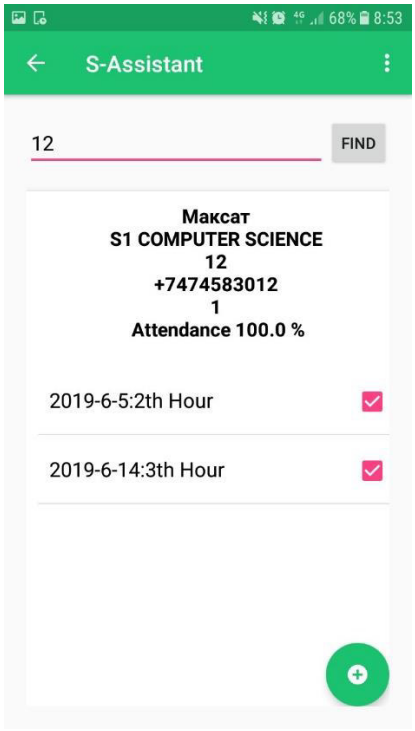


Рисунок 3.11 – Страница действии над студентом

В данном окне, можно заметить кнопку добавления студентов, также реализован поиск студента, где высвечивается искомый студент, если будет необходимо поменять информацию о студенте, это делается при долгом нажатии на вкладку самого студента.

Также имеется функционал, позаимствованный с традиционных методик преподавания, то есть отложенное редактирование занятии, когда необходимо оставить пустым у студента, графу посещаемости, чтобы заполнить его позже. Это дает возможность преподавателю исправить внесенные ошибки. Также помимо функционала студента, не менее важная часть программы, класс программного кода, отвечающий за создание журнала посещении и его логическую связь со студентом, это необходимо, так как база спроектирована до третьего уровня нормализации. Окно фиксации студента на предмет посещаемости изображен на рисунке 3.12.

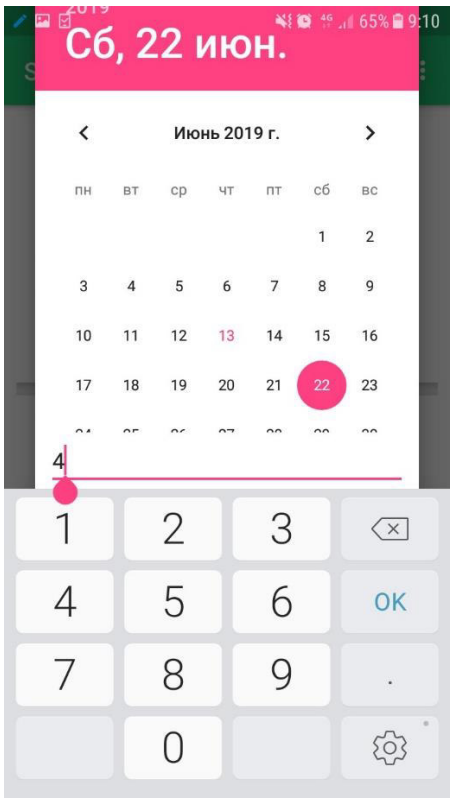


Рисунок 3.12 – Страница добавления заметки о фиксации посещения

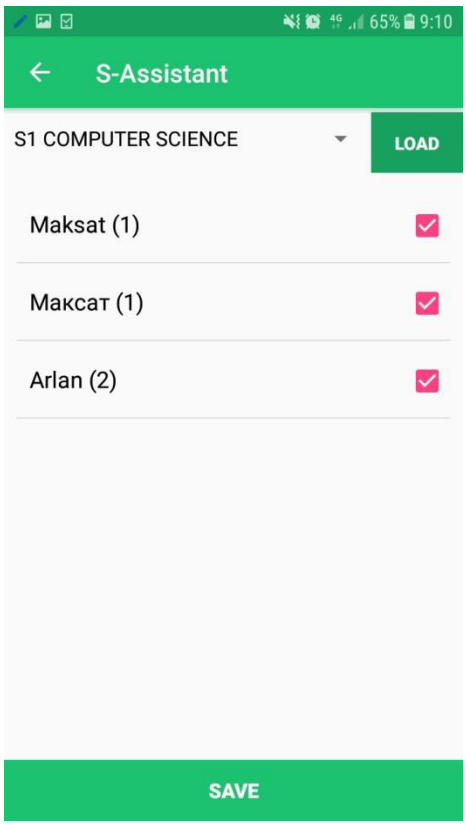


Рисунок 3.13 – Пометка студентов к ранее созданному событию посещения

Далее для того чтобы не забывать про заранее поставленные события в календаре существует подсистема «Scheduler», которая имеет в программном коде скрипт напоминание для преподавателя, на тот случай если преподаватель забудет, приложение напомнит ему о предстоящих событиях push уведомлением. Интерфейс довольно привычно напоминает всем нам знакомый будильник, окно изображено на рисунке 3.14.

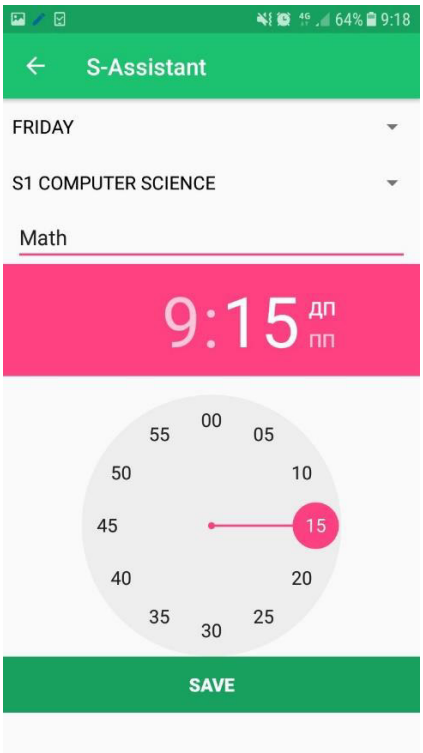


Рисунок 3.14 – Окно напоминания созданных событии

Конечно же, в целях автоматизации создан калькулятор, который считает GPA балл студента, под конец учебного сезона, преподаватели весьма быстро могут пройтись по списку своих студентов, чтобы не терять лишнее время на подсчетах, к тому, что при расчетах велик риск ошибиться, это опять-таки человеческий фактор, который хоть и не полностью исключается, но по автоматизации этой части работы преподавателей довольно много сделано.

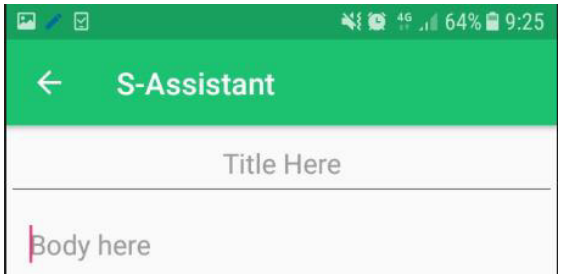


Рисунок 3.15 – Окно создания заметок

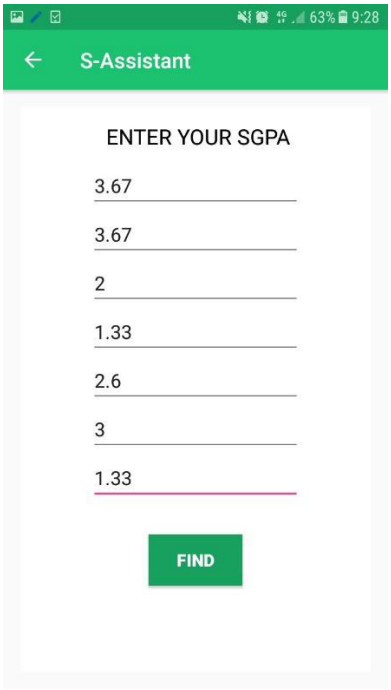


Рисунок 3.16 – Окно калькулятора GPA

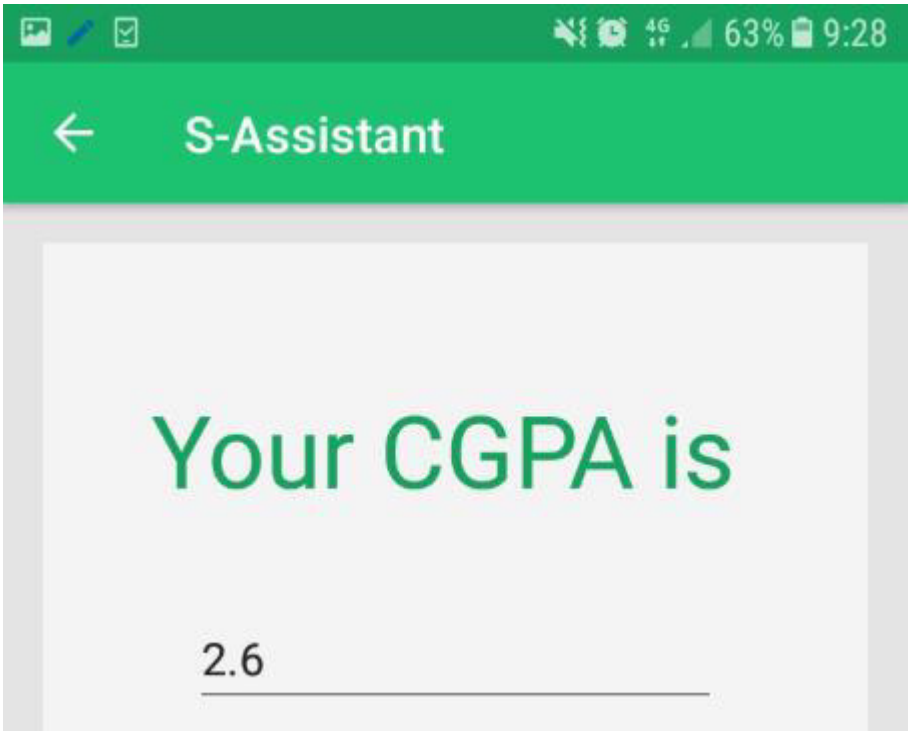


Рисунок 3.17 – Окно результата GPA калькулятора

4 Расчет экономических показателей программного продукта

4.1 Технико-экономическое обоснование

В данном дипломном проекте описываются этапы разработки мобильного приложения для всех учебных заведении, в которых методика преподавания учитывает заполнение журнала посещения.

Главная цель проектируемого мобильного приложения переход от традиционных методов «отслеживания» посещения занятии студентами, исключая человеческий фактор.

Технико-экономическое обоснование данного программного продукта содержит:

- вычисление трудоемкости и затрачиваемых ресурсов на проектирование данного мобильного приложения;
- вычисление затрат на разработку мобильного приложения;
- вычисление рыночной цены мобильного приложения;
- оценка социально – экономических итогов работы;

В стоимость разработки будут включены такие расходы как:

- оплата труда программиста;
- затраты на социальные нужды;
- затраты на профильное оборудование для программиста;
- амортизационные затраты и т.д.

4.2 Трудоемкость разработки мобильного приложения

Для нахождения трудоемкости проектирования мобильного приложения для начала необходимо создать список всех основных этапов проектировки мобильного приложения. Необходимо составить список, придерживаясь логической структуры создаваемого программного продукта, это необходимо для вычисления сроков этапов для эффективной разработки.

Разделение работ поэтапно с учетом трудоемкости на каждый этап приведена в таблице 4.1.

Таблица 4.1 – Распределение работ поэтапно, вычисление трудоемкости

Этапы разработки	Вид работы на данном этапе	Трудоемкость разработки, чел. час.
1 Планирование архитектуры приложения	Анализ и разработка задачи	40
2 Технический анализ по требованиям	Анализ приложения по его функциональности	40

Продолжение таблицы 4.1

Этапы разработки	Вид работы на данном этапе	Трудоемкость разработки, чел. час.
3 Написание технического задания	Написание основного функционала приложения	24
4 Утверждение прототипа	Создание приложения по итогам технического презентация приложения	48
5 Разработка приложения	По утвержденным пунктам технического задания написание программного кода	64
6 Тестирование и отладка	Тестирование готового приложения на предмет выявления ошибок	40
7 Написание технической документации	Написание инструкции по пользованию приложением	10
8 Ввод в эксплуатацию	Развертывание рабочей среды для приложения	24
Итого		290

4.3 Расчет затрат на разработку мобильного приложения

Определение затрат на разработку мобильного приложения включает в себя нижеследующие параметры:

- «Оборудование» - состоит из списка принадлежностей, необходимых для проектирования мобильного приложения. Расчет затрат приводится в таблице 4.2.

- «Материальные затраты» - состоят из различных материалов, необходимых для процесса создания мобильного приложения. Расчет затрат приведена в таблице 4.3.

Таблица 4.2 – Расчет затрат на оборудование

Наименование оборудования	Единица измерения	Количество	Цена за ед., тг	Сумма, тг
Системный блок PULSER Standard 3 Black	Штук	1	155245,00	155245,00
Монитор ASUS MX259H Black-Silver	Штук	2	130136,00	260272,00

Продолжение таблицы 4.2

Наименование оборудования	Единица измерения	Количество	Цена за ед., тг	Сумма, тг
Роутер TP-LINK Archer C20 V4 White-Blue	Штук	1	9490,00	9490,00
Принтер Canon i-SENSYS LBP212dw White-Black	Штук	1	89899,00	89899,00
Итого:				514906,00

Таблица 4.3 – Расчет затрат на материальные ресурсы

Наименование материального ресурса	Единица измерения	Количество	Цена за ед., г	Сумма, тг
Мышь Canyon CNE-CMSW02B Black	Штук	1	1990,00	1990,00
Клавиатура CROWN CMMK-953W Black	Штук	1	3030,00	3030,00
Коврик для мыши Canyon CND-CMP3 Black	Штук	1	1690,00	1690,00
Блок бумаги для заметок 90*90мм, 500 листов	Штук	2	230,00	460,00
Итого				7170,00

Общая сумма затрат на оборудование и материальные ресурсы ($З_м$) определяется по формуле (4.1):

$$З_м = \sum_{i=1}^{n} P_i * Ц_i , \quad (4.1)$$

Где P_i – расход i-го вида материального ресурса, натуральные единицы;

$Ц_i$ – цена за единицу i-го вида материального ресурса, тг;

i – вид материального ресурса;

n – количество видов материальных ресурсов.

$$З_м = 514906,00 + 7170,00 = 522076,00 \text{ (тенге)}.$$

Затраты на программное обеспечение идут бесплатно, так как операционная система «Linux Debian 9.5» распространяется бесплатно, также бесплатно распространяется среда разработки мобильных приложении «Android Studio».

Затраты на интернет от компании «Beeline», по их тарифам предоставляемые веткой компании «Интернет Дома», по тарифу

предоставляющий скорость потока 100 Мбит/сек, составляет за месяц 3900 тенге. Так как разработка планируется на 1 месяц, нам нужно посчитать стоимость интернета за три месяца:

$$3900 * 1 = 3900{,}00 \text{ тг.}$$

4.4 Расчет затрат на электроэнергию

Вычисление расходов на электроэнергию следует учитывать, так как, разработка мобильного приложения будет вестись на устройствах использующие электроэнергию. Расчет расходов на электроэнергию приведены в таблице 4.4.

Таблица 4.4 – Расходы на электроэнергию

Наименование оборудование	Паспортная мощность, кВт	Коэффициент использования мощности	Время работы оборудования для разработки ПП, ч	Цена электроэнергии, $\frac{тг}{кВт \cdot ч}$	Сумма, тенге
Системный блок	0,7	0,7	290	18,32	2603,27
Монитор	0,5	0,7	290	18,32	1859,48
Принтер	0,5	0,9	18	18,32	148,39
Роутер	0,03	0,7	290	18,32	111,56
Освещение	0,4	0,9	180	18,32	1187,13
Итого затраты на электроэнергию					5909,83
НДС 12%					709,17
Общая сумма					6619,00

По тарифу АлматыЭнергоСбыт цена электроэнергии составляет 18,32 $\frac{тг}{кВт \cdot ч}$ (для юридических лиц).

Общая сумма затрат на электроэнергию $З_э$ рассчитывается по формуле (4.2):

$$З_э = \sum_{i=1}^{n} M_i * K_i * T_i * Ц, \quad (4.2)$$

где M_i – паспортная мощность i-го электрооборудования, кВт;

K_i –коэффициент использования мощности i-го электрооборудования (принимается $K_i = 0{,}7 – 0{,}9$);

T_i – время работы i-го оборудования за весь период разработки ПП ч (из таблицы 1);

Ц – цена электроэнергии, тг/кВт×ч;
i – вид электрооборудования;
n – количество электрооборудования.

$З_э$ = 2603,27 + 1859,48 + 148,39 + 111,56 + 1187,13 = 5909,83 тг.

С учетом НДС сумма составит:

$З_э$ = 5909,83 + (5 909,83 * 12%) = 6619,00 тг.

4.5 Затраты на оплату труда

Для разработки программного приложения создана группа из двух человек.

Трудоемкость разработки составляет 290 часа / 8 = 36 рабочих дня.

В «Затраты на оплату труда» – входят расходы на оплату всех работников, занятых разработкой ПП. Затраты на оплату труда рассчитываются по форме, приведенной в таблице 4.5.

Таблица 4.5 – Расходы на оплату труда

Категория работника	Квалификация	Трудоемкость разработки, час	Часовая ставка, тг/ч	Сумма заработной платы за месяц, тг.
Разработчик	Инженер Программист	290	1704,54	494316,6
Продукт менеджер	Инженер Аналитик	160	1136,36	181817,6
Итого				676 134,20

Общая сумма затрат на оплату труда $З_т$ определяется по формуле (4.3):

$$З_{тр} = \sum_{i=1}^{n} ЧС_i \times T_i, \quad (4.3)$$

где $ЧС_i$ – часовая ставка i-го работника, тенге;
Ti – трудоемкость разработки ПП, чел×ч;
i – категория работника;
n – количество работников, занятых разработкой ПП.

Часовая ставка работника может быть рассчитана по формуле (4.4):

$$ЧС_i = \frac{ЗП_i}{ФРВ_i}, \quad (4.4)$$

где ЗПi – месячная заработная плата i-го работника, тг;

ФРВi – месячный фонд рабочего времени i-го работника, час.

$ЧС_{менеджер}$ = 200000,00 / 8 * 22 = 1136,36 (тенге/час).

$ЧС_{разработчик}$ = 300000,00 / 8 * 22 = 1 704,54 (тенге/час).

З тр = (1704,54 * 290) +(1136,36 * 160) = 676 134,20 тенге.

4.6 Расчет затрат по социальному налогу

Социальный налог. В статью «Социальный налог» включена сумма, рассчитываемая как 9,5% от затрат на оплату труда всех работников $З_т$, занятых разработкой ПП. При расчете необходимо учесть, что пенсионные отчисления (10% от $З_т$) не облагаются социальным налогом. Социальный налог можно рассчитать по формуле (4.5):

$$С_н = (ФОТ - ПО) * 0{,}095 \quad (4.5)$$

где ПО – отчисления в пенсионный фонд, они составляют 10% от ФОТ.

ПО= 676 134,20*0,1=67 613,42 тенге

$С_н$=(676 134,20-67 613,42)*0,095=57 809,47 тенге

4.7 Амортизация основных фондов и прочие расходы

Амортизация основных фондов.

В статье «Амортизация основных фондов» рассчитывается сумма амортизационных отчислений от стоимости оборудования и программного обеспечения (ПО), которые использовались в ходе разработки ПП. Амортизационные отчисления рассчитываются по форме, приведенной в таблице 4.6.

Таблица 4.6 – Амортизационные расходы

Наим-е Оборуд-я	Стоимость оборуд-я, тг	Годовая норма аморт-ии, %	Эффективный фонд времени работы оборудования, ч/год	Время работы оборудования для разработки , ч	Сумма, тенге
Монитор ASUS Black-Silver	130136,00	15	2016	290	2 807,99
Роутер TP-LINK V4 White-Blue	9490,00	15	2016	290	204,76
Системный блок PULSER Standard 3 Black	155 245,00	20	2016	290	4 466,37
Принтер Canon i-SENSYS White-Black	89899,00	15	2016	160	1 070,22
Итого амортизационных расходов					8 549,34

Общая сумма амортизационных отчислений определяется по формуле (4.6):

$$З_{АМ} = \sum_{i=1}^{n} \frac{Ф_i \times Н_{Ai} \times Т_{НИРi}}{100 \times Т_{эф}}, \quad (4.6)$$

где Фi – стоимость i-го ОФ, тг;

$Н_{Ai}$ – годовая норма амортизации i-го ОФ, %;

$Т_{НИР}$ –время работы i-го ОФ за весь период разработки ПП, ч;

$Т_{эф}$ – эффективный фонд времени работы i-го ОФ за год, ч/год;

i – вид ОФ;

n – количество ОФ.

$З_{АМ}$ = 130 136,00* 15 * 290 / 100 * 2016 = 2 807,99 (тенге).

$З_{АМ}$ = 9 490,00* 15 * 290 / 100 * 2016 = 204,76 (тенге).
$З_{АМ}$ = 155 245,00 * 20 *290 / 100 * 2016 = 4 466,37 (тенге)
$З_{АМ}$ = 89 899,00 * 15 *160 / 100 * 2016 = 1 070,22 (тенге).

4.8 Полная сметная отчетность по стоимости разработки мобильного приложения

На основе всех представленных расчетов необходимо оформить смету расходов на разработку мобильного приложения согласно форме, которая приведена в таблице (4.8). На рисунке 4.1 продемонстрирована диаграмма рабочих расходов.

Таблица 4.7 – Полная смета расходов на мобильное приложение

№	Наименование статьи затрат	Всего, тг	%
1	Затраты на оборудование	514 906,00	40,38
2	Затраты на материальные средства	7 170,00	0,56
3	Затраты на электроэнергию	6 619,00	0,51
4	Затраты на оплату труда	676 134,20	53,02
5	Затраты на социальные налоги	57 809,47	4,53
6	Амортизация основных фондов	8 549,34	0,67
7	Прочие расходы (интернет)	3 900,00	0,33
ИТОГО:		1 275 088,01	100%

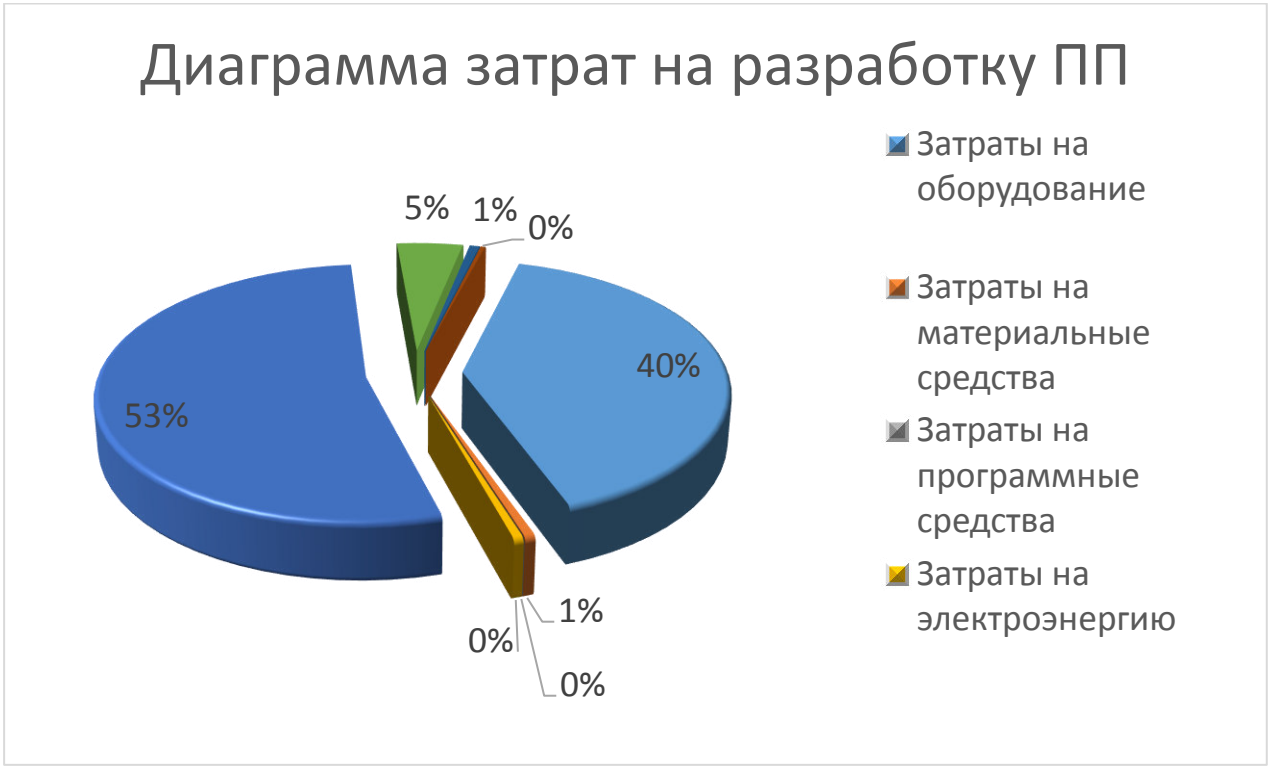


Рисунок 4.1 – Диаграмма затрат

4.9 Определение возможной (договорной) цены ПП

Величина возможной (договорной) цены ПП устанавливается на основе эффективности, качества и сроков её выполнения на уровне, отвечающем экономическим интересам заказчика (потребителя) и исполнителя.

Договорная цена $Ц_д$ для прикладных ПП рассчитывается по формуле (4.7):

$$Ц_Д = З_{НИР}\left(1+\frac{Р}{100}\right), \quad (4.7)$$

где $З_{НИР}$ – затраты на разработку ПП, тг;

Р – средний уровень рентабельности ПО, (%). Данный параметр принят равным 25%.

$Ц_Д$ = 1 275 088,01 * (1+25/100) = 1 275 088,01+ (1 275 088,01 * 0,25) = 1 593 860,01 тг.

Прибыль = 1 275 088,01 * 0,25 = 318 772,00

Далее определяется цена реализации с учетом налога на добавленную стоимость (НДС), ставка (НДС) устанавливается законодательно. Налоговым Кодексом РК. На 2019 год ставка НДС установлена в размере 12%.

Цена реализации с учетом НДС рассчитывается по формуле (4.8):

$$Ц_Р = Ц_Д + Ц_Д * НДС \quad (4.8)$$

$Ц_р$ = 1 593 860,01 + 1 593 860,01 * 0,12 = 1 746 870,57 тг.

В данном разделе были произведены расчеты затрат на приобретение необходимого оборудования, материальных ресурсов и программного обеспечения для разработки приложения журнала посещаемости, включая расчет затрат на оплату труда.

Смета затрат на программное обеспечение составила 1 275 088,01 тенге. Возможная договорная цена программного продукта, с учетом НДС составила 1 746 870,60 тенге. Рентабельность (прибыль) программного обеспечения составляет 318 772,00тенге.

5 Безопасность жизнедеятельности

5.1 Определение параметров рабочей среды

Организация рабочей среды для офиса программистов подразумевает наличие множественных факторов удобства для эффективной работы. Для того чтобы процесс работы сотрудника шла своим чередом необходимы профильные оборудования, мощные системные блоки, яркие большие мониторы, хорошее освещение, просторные рабочие пространства, надежное интернет соединение.

Офис компании расположен в двухэтажном жилом помещении, где одна комната из одной квартиры переделана в рабочий офис, с отдельной входной группой на улицу. Планировка помещения представлена на рисунке 5.1.

Рабочее помещение рассчитано для работы двух людей, в лице аналитика и меня в лице программиста разработчика.

Параметры офиса:

- длина L = 6 метров;
- ширина W = 5 метров;
- высота H = 2.75 метров.

В офисе имеется один оконный проем, с солнечной стороны данного построения, естественный свет попадает в комнату без проблем, а вентиляции в данной комнате не имеется, поэтому я буду вести расчет кондиционирования воздуха в данном помещении, для комфортабельности условии.

Используемая техника:

- два стационарных компьютера;
- принтер;
- роутер.

Таблица 5.1 – Параметры

Город		Алматы
Параметры (длина, ширина, высота)		6 м, 5 м, 2.75 м
Данные по бытовой технике	Количество, шт	4
	Мощность, $P_{об}$, кВт/ч	0,7
	КПД, η	0,6
Данные по освещению	Мощность. N ос.уст., Вт/м^2	60
	Вид ист. св.	Лампа накаливания
Число трудящихся		2
Оконные проемы	Количество	1
	Площадь, м^2	1,92
	Вид	Шторы, небольшое загрязнение

	Расположение	ЮЗ

Продолжение таблицы 5.2

Расчетное время, час		9-10
Температура в офисе, ^{0}C	Летнее время	26
	Зимнее время	18

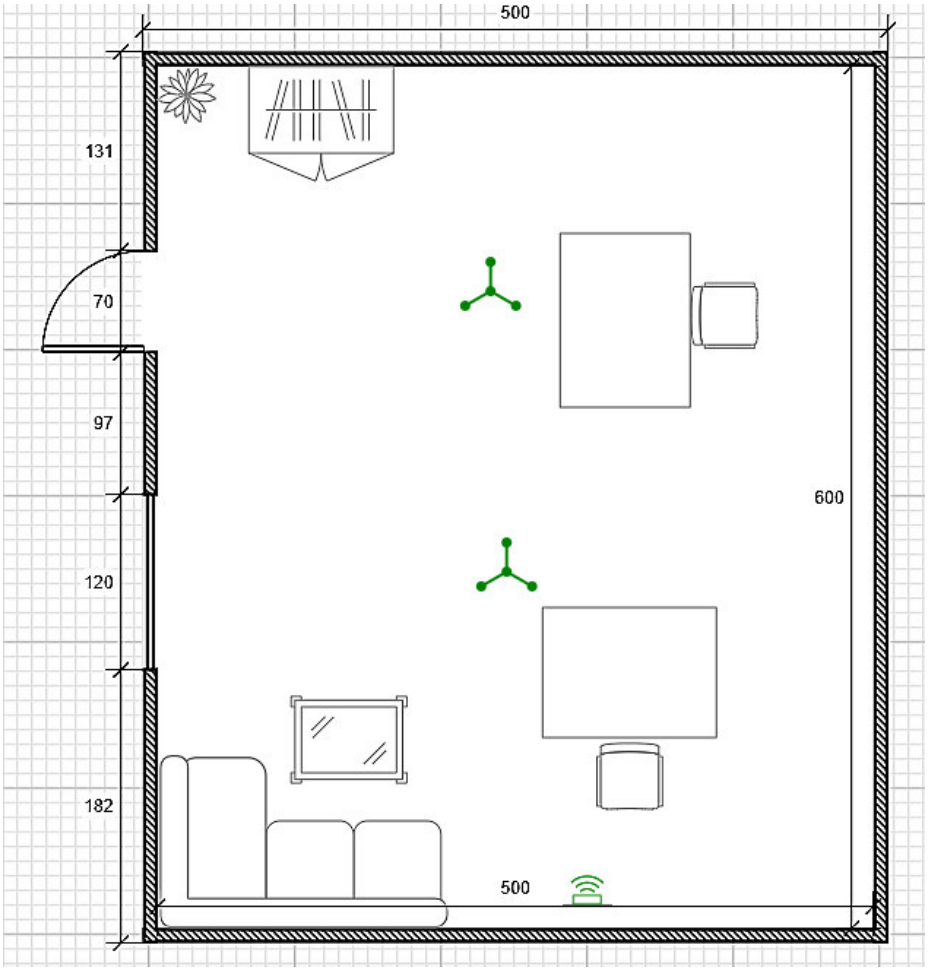


Рисунок 5.1 – Планировка помещения

5.2 Расчет тепловых нагрузок в офисе

В помещениях действуют тепловые нагрузки, возникающие вне помещения(наружные) возникающие внутри помещения(внутренние).

Наружные тепловые нагрузки имеют следующие параметры:

- тепловой баланс в учете разницы температур с помощью проемов помещения;
- разница температур в связи со временами года меняется, то есть летом разница является положительной, поэтому приток тепла происходит внутрь, в зимнее время приток тепла происходит в противоположном направлении;
- поступления тепла от застекленных проемов;
- поступление тепла от инфильтрации.

Тепловой баланс вычисляется по формуле 1.1:

$$Q_t = V_o * X_o * (t_i * t_o), \quad (1.1)$$

Где: V_o – объём помещения, м^3;

X_o – удельная тепловая единица, Вт/м^3*°С;

t_i – наружная температура;

t_o – внутренняя температура;

$$V_o = 6 * 5 * 2{,}75 = 82{,}5 \text{ м}^3,$$

$$X_o = 0{,}4 \text{ Вт/м}^3 * °C ,$$

Для теплого времени года:

- t_i = 30,2 °С;
- t_o = 26 °С;

$$Q_t = 82{,}5 * 0{,}4 * 4{,}2 = 138{,}6 \text{ Вт},$$

Для холодного времени года:

- t_i = - 15,0°С;
- t_o = 18 °С;

$$Q_t = 82{,}5 * 0{,}4 * |-33| = 1089 \text{Вт},$$

Переизбыток тепла от солнечного света в зависимости от типа стекла поглощается до 95% средой помещения, остальное отражается.

Поступление тепла от солнечного света через стеклянные поверхности вычисляется по формуле 1.2:

$$Q_s = (q^1 * F_0^1) + (q^2 * F_0^2) * \beta_{с.з} , \quad (1.2)$$

Где: q^1, q^2 – тепловые потоки от прямой и рассеянной солнечной спектрации, Вт/м^2;

F_0^1, F_0^2 – площади оконного проема, облучаемые солнечным светом;

$\beta_{с.з}$ – коэффициент пропускания тепла, в нашем примере 0,15;

При отсутствии козырьков, ребер, веранд, для длительности влияния солнечного света выбирается следующий параметр $F_0^1 = F_0^п = F_0 = 0$

Формула вычисления поступления тепла от солнечного света теперь будет вычисляться по формуле 1.3:

$$Q_s = (q^2 * F_0) * \beta_{с.з} = q_{вр} * K_1^t * K_2^t * \beta_{с.з} * n * S_0, \quad (1.3)$$

Где: $q_{вр}$ – тепловые потоки рассеянной радиации, Вт/м^2. Для широты в 44 °СШ утром в начале рабочего дня при юго-западном расположении она составляет 97 Вт/м^2;

F_0 – площадь оконного проема, она составляет 1,92 м^2;

K_1^t – коэффициент затемнения = 1,03;

K_2^t – коэффициент загрязнения = 0,90.

$$Q_s = 97 * 1,03 * 0,90 * 0,15 * 1,92 = 25,89 \text{ Вт},$$

Внутренние нагрузки теплового баланса складывается из тепла:

- выделяемый людьми;
- искусственным освещением;
- бытовой техникой.

Тепло выделяемое людьми зависит от интенсивности человека. Летом при 26 °С один человек выделяет в среднем 175 Вт. В зимнее время при 18 °С человек выделяет в среднем 100 Вт.

Поступление тепла от искусственного освещения, компьютерного оборудования вычисляется по формуле 1.4:

$$Q_{осв} = \eta * N_{осв} * F_{пол}, \quad (1.4)$$

Где: η = коэффициент перехода энергии в тепло(0,92 – 0,97);

$N_{осв}$ = установленная мощность лампы (60 Вт/м^2);

$F_{пол} = 6 * 5 = 30$ м^2;

$$Q_{осв} = 0,92 * 60 * 30 = 1656 \text{ Вт},$$

Тепло поступающее от техники вычисляется по формуле 1.5:

$$Q_т = N_{уст} * K, \quad (1.5)$$

$$Q_т = 0,7 * 4 * 0,6 * 10^3 = 1,68 \text{ кВт},$$

Теплопритоки от находящейся в офисе техники это 30% мощности оборудования:

$$Q_о = 1 * 0.3 * 0.3 * 10^3 = 0.09 \text{ кВт},$$

5.3 Расчет теплового баланса

По выше выполненным расчетам можно составить тепловой баланс данного помещения:

$$Q_{лето} = \sum 25{,}89+175+1656+ 1680+90+138{,}6 = 3765{,}49 \text{ Вт},$$

$$Q_{зима} = \sum 25{,}89+100+1656+ 1680+90+1089 = 4640{,}89 \text{ Вт},$$

Рассчитывать потребный воздухообмен будем по формуле 1.6:

$$L = \frac{Q*860}{C * 8 * \gamma}, \quad (1.6)$$

Где: C – 0,24 ккал/(кг * °C) – теплоемкость воздуха;
γ – 1206 кг/м^3 – удельная масса приточного воздуха.

$$L = \frac{3765{,}49*860}{0.24*8*1206} = 1398{,}52 \text{ м}^3/\text{час},$$

Определение кратности воздухообмена:

$$N = L/V_{офис} = 1398{,}52 / 82{,}5 = 16{,}95 \text{ час}^{-1}$$

5.4 Выбор кондиционера. Схема расположения

Подводя итоги расчета, для удаления лишнего тепла и эффективного кондиционирования воздуха нужно использовать систему вентиляции, которая сможет подать требуемую подачу воздуха, то есть 1398,52 м^3/час.

Расчетная мощность кондиционера должна быть в диапазоне от -5% до +15% расчетной мощности выбираемого кондиционера, на данный момент расчетная мощность составляет 3,7 кВт. То есть выбирать можно до 4,5 кВт, мой выбор остановился на кондиционере Almacom ACF-36HM, вентиляцию воздуха можно поднять до максимальных 1750 м^3/час.

Таблица 5.2 – Технические параметры

Площадь	90–100 м²
Потребляемая мощность (охлаждение)	3,45 кВт
Потребляемая мощность (нагрев)	3,58 кВт
Особенности	LED дисплей, Авторестарт, Вентиляция, Обогрев, Охлаждение

Продолжение таблицы 5.2

Цвет	Белый
Тип кондиционера	Напольно-подпотолочный
Расход воздуха в помещении	1750 $м^3$/час
Мощность обогрева	10,70 кВт
Мощность охлаждения	10,55 кВт
Вес комплекта в упаковке (кг)	121
Рекомендуемая площадь	30 - 50 $м^2$

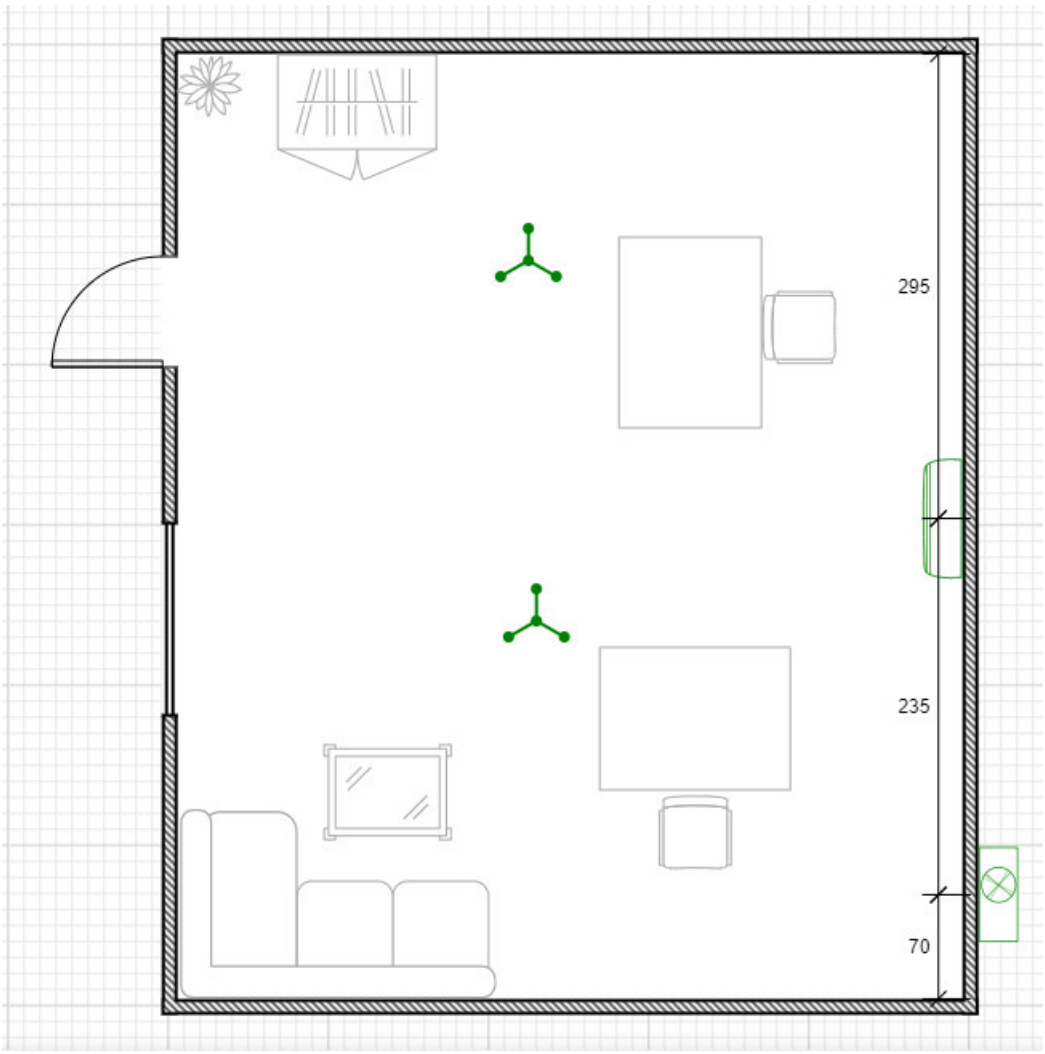


Рисунок 5.2 – Расположение системы кондиционирования

Заключение

В результате разработки сервиса для управления списком задач получилось удобное и простое в использовании мобильное приложение, и веб-приложение. Мобильное приложение поддерживает много версий платформы Android, при этом оно реализовано с применением принципов Material Design, характерных для современных версий платформы. Интерфейс веб-приложения аналогичен интерфейсу мобильного приложения, за исключением некоторых элементов. Сервер позволяет пользователю получать доступ и управлять своими задачами из одной и той же учетной записи, из мобильного устройства на платформе Android. Реализован основной функционал, которого достаточно для продуктивного управления списком студентов и их посещаемости. К основному функционалу относится возможность добавления, изменения, завершения и удаления студентов, фиксация их посещаемости, создание пометок и установка событий.

Для взаимодействия со студентами не нужно перемещаться по большому количеству меню. Навигация по спискам студентам в случае мобильного приложения вынесена в боковое меню, а в случае веб-приложения в верхнее меню страницы. Это позволяет максимально эффективно использовать место на экране и концентрировать внимание пользователя на списке задач. В случае дальнейшего развития сервиса для управления списком студентов возможно добавление нового функционала, без существенного изменения принципов взаимодействия мобильного приложения с сервером. При взаимодействии с сервером характерные для задачи данные не берутся в расчет, используются только нужные для синхронизации служебные поля. В случае расширения функционала сервиса в протоколе обмена информацией с сервером может быть сохранена обратная совместимость. Для пользователей сервиса есть возможность получить доступ в случае утраты пароля. Если пользователь сервиса все же затрудняется в выборе действия, хочет узнать какой функционал может предоставить сервис, на этот случай предусмотрен раздел справки. В разделе справки кратко описаны действия, которые необходимо выполнить для определенного результата, например, куда нужно нажать для того чтобы вернуть студента из списка удаленных. Если пользователь не хочет, чтобы мобильное приложение выполняло синхронизацию с сервером, то для этого есть возможность отключения синхронизации в настройках приложения.

Список использованной литературы

1 Remember The Milk [Электронный ресурс]. – Режим доступа: https://www.rememberthemilk.com/help – Загл. с экрана. – яз. англ.

2 AnyDo [Электронный ресурс]. – Режим доступа: https://www.any.do/anydo – Загл. с экрана. – яз. англ.

3 Evernote [Электронный ресурс]. – Режим доступа: https://evernote.com – Загл. с экрана. – яз. англ.

4 Material Design Guide Line [Электронный ресурс]. – Режим доступа: https://www.google.com/design/spec/material-design/introduction.html – Загл. с экрана. – яз. англ.

5 ECMAScript 5.1 specification [Электронный ресурс]. – Режим доступа: http://www.ecma-international.org/ecma-262/5.1 – Загл. с экрана. – яз. англ.

6 Android [Электронный ресурс]. – Режим доступа: https://en.wikipedia.org/wiki/Android_(operating_system) – Загл. с экрана. – яз. англ.

7 Dalvik [Электронный ресурс]. – Режим доступа: https://en.wikipedia.org/wiki/Dalvik_(software) – Загл. с экрана. – яз. англ.

8 Intellij IDEA [Электронный ресурс]. – Режим доступа: https://www.jetbrains.com/idea/features – Загл. с экрана. – яз. англ.

9 Google App Engine [Электронный ресурс]. – Режим доступа: https://cloud.google.com/appengine – Загл. с экрана. – яз. англ.

10 Node Package Manager [Электронный ресурс]. – Режим доступа: https://npmjs.com – Загл. с экрана. – яз. англ.

11 MEAN [Электронный ресурс]. – Режим доступа: https://en.wikipedia.org/wiki/MEAN_(software_bundle) – Загл. с экрана. – яз. англ.

12 Bitbucket [Электронный ресурс]. – Режим доступа: https://bitbucket.org/product/features – Загл. с экрана. – яз. англ.

13 REST [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/REST – Загл. с экрана. – яз. рус.

14 mail.yandex.ru [Электронный ресурс]. – Режим доступа: https://mail.yandex.ru – Загл. с экрана. – яз. рус.

15 Bootstrap [Электронный ресурс]. – Режим доступа: http://getbootstrap.com/getting-started – Загл. с экрана. – яз. англ.

16 Экспертное оценивание [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Экспертное_оценивание – Загл. с экрана. – яз. рус.

17 Material Design [Электронный ресурс] // Google. URL: https://material.google.com/ (дата обращения: 25.10.2016). 83

18 Don’t repeat yourself [Электронный ресурс] // Википедия. URL: http://ru.wikipedia.org/?oldid=75063200 (дата обращения: 29.10.2016).

19 Фабричный метод (шаблон проектирования) [Электронный ресурс] // Википедия. URL: http://ru.wikipedia.org/?oldid=82081042 (дата обращения: 05.11.2016).

20 Замыкание (программирование) [Электронный ресурс] // Википедия. URL: http://ru.wikipedia.org/?oldid=81810616 (дата обращения: 11.11.2016).

21 Пол Дейтел, Харви Дейтел, Александер Уолд. Android для разработчиков. 3-е издание. СПб.: Питер, 2016. 512 с.

22 Марио Цехнер. Программирование игр под Android. СПб.: Питер, 2013. 688 с.

23 Android Development [Электронный ресурс] // Википедия. URL: http://ru.wikipedia.org/?oldid=82147331 (дата обращения: 02.11.2016).

Приложение А
(Обязательное)

Техническое задание на разработку мобильного приложения «Журнал посещаемости»

Содержание:
- концепция и основная идея;
- задачи, решаемые при помощи приложения;
- этапы работ по созданию системы;
- порядок оформления и предъявления заказчику результатов работ по разработке мобильного приложения;
- публикация приложения;
- требования к проекту и программному обеспечению;
- общие требования к дизайну экранов пользователя;
- структура и описание административного интерфейса;
- главная страница;
- управление клиентами /пользователями;
- управление стандартными полями профиля пользователя;
- общая конфигурация приложения;
- группы пользователей;
- администратор;
- клиент;
- меню клиента;
- карта проекта / мобильного приложения - экраны пользователя;
- главный экран;
- экран авторизации;
- экран регистрации пользователя;
- экран посещении.

Концепция и основная идея.

Создание данного мобильного приложения необходимо как дополнение для заказа продуктов питания с доставкой с сайта http://limoncello.com.ua/ для владельцев мобильных на базе основных операционных систем IOS и Android.

Задачи, решаемые при помощи приложения:

Основная цель — упростить заказ продуктов питания для пользователей сайта, владеющих мобильными устройствами на базе операционной системы AndroidOS.

Этапы работ по созданию системы:

Работа по созданию данного приложения разделяется на следующие этапы:
- этап оценки стоимости и срока разработки;
- этап разработки мокапов экранов пользователя и администратора;
- этап разработки дизайна;
- этап разработки интерфейса администратора;

- этап разработки интерфейса пользователя;
- этап верстки интерфейсов пользователя и администратора;
- тестирование юзабилити и безопасности проекта;
- публикация проекта;

Порядок оформления и предъявления заказчику результатов работ по разработке мобильного приложения.

Исполнитель ведет разработку проекта на собственных серверах. После окончания разработки и завершения всех этапов тестирования проекта заказчик принимает решение о выборе хостинг пространства для размещения собств. просчитывает разработчик на этапе оценки стоимости и срока разработки проекта.

Общие требования к дизайну экранов пользователя:

Все макеты должны быть выполнены в одном из 2-х форматов — PSD (Photoshop Document) или TIFF (Tagged Image File Format). Промежуточные версии макетов допускается предоставлять в любом формате, доступном к просмотру в системе windows без установки дополнительного программного обеспечения. Каждый элемент дизайна должен быть представлен в отдельном слое. Предпочтительно при разработке получить шаблон «резиновый» — в шапке и подвале страницы не менее 20-50 пикселей с правой стороны должны быть от рисованы так, чтобы их без проблем можно было дублировать. Возможна разработка шаблона под определенные разрешения по согласованию с заказчиком. Фон экрана должен быть однородным и повторяющимся. В том случае, если требуется использовать текстурированные — текстура точно так же должна быть однородной и легко дублируемой. Мелкие элементы дизайна (иконки, стрелки, буллиты и т.д.) в случае их многократного повторения в макете (например, маркированный список, меню и прочее) должны быть представлены в отдельных файлах. В самом макете все повторения могут размещаться в одном слое. Внешний вид каждого из экранов пользователя должен быть разработан под оба положения экрана мобильных устройств (вертикальное и горизонтальное) Структура и описание административного интерфейса Главная страница админ интерфейса — содержит ссылки для перехода к другим разделам интерфейса администратора, блок последних зарегистрированных пользователей, блок последних заказов, блок последних добавленных товаров, блок последних сообщений. Общее количество пользователей в системе и другую полезную информацию, согласованную с заказчиком.

Управление каталогом управление категориями — добавление/изменение существующих категорий товаров, включающих следующие параметры: название категории, порядком вывода категорий каталога, изображение категории товара, родительская категория каталога (при необходимости). Меню клиента Меню пользователя должно вызываться с

помощью визуального активного элемента. Это может быть иконка меню либо направление, в котором надо потянуть экран для получения доступа к меню.

Меню пользователя должно содержать ссылки для перехода на следующие пункты: Главная Регистрация Вход (все пункты ниже доступны только для авторизованных пользователей).

Главный экран.

На данном экране пользователям будет предоставлен выбор основных возможностей:

- пройти авторизацию (ВОЙТИ)
- пройти регистрацию для более удобного заказа (РЕГИСТРАЦИЯ)

На данном экране, помимо перечисленных выше ссылок должен быть предусмотрен ротатор. Это могут быть страницы новости, страница акции, страница конкретного товара, страница категории либо просто статическая страница, созданная менеджером.

Экран авторизации.

Данный экран приложения содержит следующие элементы:

- поле ввода логина пользователя, указанного при регистрации;
- поле ввода пароля пользователя, указанного при регистрации;
- ссылка для восстановления пароля на email указанный при регистрации;
- ссылка на экран регистрации пользователя.

Экран регистрации пользователя.

Данный экран, представляет собой форму, по заполнении которой пользователь получит email подтверждение прохождения процедуры регистрации в приложении. После заполнения пользователем формы регистрации приложение должно переадресовывать пользователя на страницу авторизации. Экран регистрации должен содержать следующие поля для заполнения:

- ФИО пользователя — Фамилия, Имя и Отчество пользователя, региструющегося на сайте;
- логин пользователя — никнейм пользователя, под которым он сможет проходить процедуру авторизации в приложении. Не может начинаться с пробела;
- пароль пользователя — обязательно использование цифр и букв, минимальное количество символов — 6;
- email пользователя — поле, содержащее контактный адрес электронной почты пользователя для прохождения процедуры восстановления пароля, получения уведомлений от системы и администраторов;
- телефон пользователя — поле номера телефона пользователя, для получения sms уведомлений и сообщений от системы и администраторов.

Приложение Б
(обязательное)

Листинг Программного кода

```
class TeacherController extends Controller
{
    public function index()
    {
        $teachers = Teacher::query()->where('status', '=', true)-
>get();

        if ($teachers)
        {
            return response()-
>json(TeacherResource::collection($teachers), Response::HTTP_OK);
        }
        else
        {
            return response()->json(null, Response::HTTP_BAD_REQUEST);
        }
    }
}

class UserController extends Controller
{
    public function index()
    {
        $users = User::all();

        if ($users)
        {
            return response()->json(UserResource::collection($users),
Response::HTTP_OK);
        }
        else
        {
            return response()->json(null, Response::HTTP_BAD_REQUEST);
        }
    }
}

APP_NAME=Laravel
APP_ENV=local
APP_KEY=base64:smX1sfm9wxprefLF6qlnmtW73xPba6SW81TLLNwVZZY=
APP_DEBUG=true
APP_URL=http://localhost

LOG_CHANNEL=stack

DB_CONNECTION=pgsql
DB_HOST=127.0.0.1
DB_PORT=5432
DB_DATABASE=attendance_app
DB_USERNAME=maks
```

```
DB_PASSWORD=123
BROADCAST_DRIVER=log
CACHE_DRIVER=file
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
SESSION_LIFETIME=120

REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379

MAIL_DRIVER=smtp
MAIL_HOST=smtp.mailtrap.io
MAIL_PORT=2525
MAIL_USERNAME=null
MAIL_PASSWORD=null
MAIL_ENCRYPTION=null

AWS_ACCESS_KEY_ID=
AWS_SECRET_ACCESS_KEY=
AWS_DEFAULT_REGION=us-east-1
AWS_BUCKET=

PUSHER_APP_ID=
PUSHER_APP_KEY=
PUSHER_APP_SECRET=
PUSHER_APP_CLUSTER=mt1

MIX_PUSHER_APP_KEY="${PUSHER_APP_KEY}"
MIX_PUSHER_APP_CLUSTER="${PUSHER_APP_CLUSTER}"

<?php

/**
 * Laravel - A PHP Framework For Web Artisans
 *
 * @package  Laravel
 * @author   Taylor Otwell <taylor@laravel.com>
 */

define('LARAVEL_START', microtime(true));

/*
|--------------------------------------------------------------------------
----
| Register The Auto Loader
|--------------------------------------------------------------------------
----
|
| Composer provides a convenient, automatically generated class loader
for
| our application. We just need to utilize it! We'll simply require it
| into the script here so that we don't have to worry about manual
| loading any of our classes later on. It feels great to relax.
|
```

```
*/
require __DIR__.'/../vendor/autoload.php';

/*
|--------------------------------------------------------------------------
| Turn On The Lights
|--------------------------------------------------------------------------
|
| We need to illuminate PHP development, so let us turn on the lights.
| This bootstraps the framework and gets it ready for use, then it
| will load up this application so that we can run it and send
| the responses back to the browser and delight our users.
|
*/

$app = require_once __DIR__.'/../bootstrap/app.php';

/*
|--------------------------------------------------------------------------
| Run The Application
|--------------------------------------------------------------------------
|
| Once we have the application, we can handle the incoming request
| through the kernel, and send the associated response back to
| the client's browser allowing them to enjoy the creative
| and wonderful application we have prepared for them.
|
*/

$kernel = $app->make(Illuminate\Contracts\Http\Kernel::class);

$response = $kernel->handle(
    $request = Illuminate\Http\Request::capture()
);

$response->send();

$kernel->terminate($request, $response);

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="assistant.genuinecoder.s_assistant">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".main.AppBase">
```

```
            <intent-filter>

<action android:name="android.intent.action.MAIN" />

                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".main.profile.ProfileActivity"
            android:parentActivityName=".main.AppBase" />
        <activity
            android:name=".main.profile.StudentRegistration"
            android:parentActivityName=".main.profile.ProfileActivity"
/>
        <activity
            android:name=".main.notes.NoteCreate"
            android:parentActivityName=".main.notes.NoteActivity" />
        <activity
            android:name=".main.attendance.AttendanceActivity"
            android:parentActivityName=".main.AppBase" />
        <activity
            android:name=".main.components.ResultActivity"
            android:parentActivityName=".main.components.CgpaActivity"
/>
        <activity
            android:name=".main.components.CgpaActivity"
            android:parentActivityName=".main.AppBase" />
        <activity
            android:name=".main.notes.NoteActivity"
            android:parentActivityName=".main.AppBase" />
        <activity
            android:name=".main.schedule.Scheduler"
            android:parentActivityName=".main.AppBase" />
        <activity
            android:name=".main.schedule.CreateScheduleActivity"
            android:parentActivityName=".main.schedule.Scheduler" />
        <activity
            android:name=".main.components.SettingsActivity"
            android:label="@string/title_activity_settings"
            android:parentActivityName=".main.AppBase" />
        <activity
            android:name=".main.components.About"
            android:parentActivityName=".main.AppBase" />
        <activity android:name=".main.profile.EditStudentActivity"
            android:parentActivityName=".main.AppBase"/>
    </application>

</manifest>


apply plugin: 'com.android.application'

android {
    compileSdkVersion 28
```

```
    defaultConfig {
        applicationId "assistant.genuinecoder.s_assistant"
minSdkVersion 19
        targetSdkVersion 28
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner
"android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-
android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:28.0.0'
    implementation 'com.android.support.constraint:constraint-
layout:1.1.3'
    implementation 'com.android.support:design:28.0.0'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.2'
    androidTestImplementation
'com.android.support.test.espresso:espresso-core:3.0.2'


}

package assistant.genuinecoder.s_assistant.main.attendance;

import android.app.Activity;
import android.database.Cursor;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
import android.widget.Spinner;
import android.widget.Toast;

import java.util.ArrayList;

import assistant.genuinecoder.s_assistant.R;
import assistant.genuinecoder.s_assistant.main.AppBase;
import assistant.genuinecoder.s_assistant.main.components.ListAdapter;

public class AttendanceActivity extends AppCompatActivity {

    public static String time, period;
```

```
    ListView listView;
    ListAdapter adapter;
ArrayAdapter<String> adapterSpinner;
    ArrayList<String> names;
    ArrayList<String> registers;
    Activity thisActivity = this;
    Spinner spinner;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_attendance);

        time = getIntent().getStringExtra("DATE");
        period = getIntent().getStringExtra("PERIOD");

        Log.d("Attendance", time + " -- " + period);
        listView = (ListView) findViewById(R.id.attendanceListViwe);
        names = new ArrayList<>();
        registers = new ArrayList<>();

        Button btn = (Button) findViewById(R.id.loadButton);
        assert btn != null;
        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                loadListView(v);
            }
        });

        Button btnx = (Button) findViewById(R.id.buttonSaveAttendance);
        assert btnx != null;
        btnx.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Toast.makeText(getBaseContext(), "Saving",
Toast.LENGTH_LONG).show();
                adapter.saveAll();
            }
        });

        spinner = (Spinner) findViewById(R.id.attendanceSpinner);
        adapterSpinner = new ArrayAdapter<String>(this,
android.R.layout.simple_spinner_dropdown_item, AppBase.divisions);
        assert spinner != null;
        spinner.setAdapter(adapterSpinner);

    }

    public void loadListView(View view) {
        names.clear();
        registers.clear();
        String qu = "SELECT * FROM STUDENT WHERE cl = '" +
spinner.getSelectedItem().toString() + "' " +
                "ORDER BY ROLL";
```

```
        Cursor cursor = AppBase.handler.execQuery(qu);
        if (cursor == null || cursor.getCount() == 0) {

Log.e("Attendance", "Null cursor");
        } else {
            int ctr = 0;
            cursor.moveToFirst();
            while (!cursor.isAfterLast()) {
                names.add(cursor.getString(0) + " (" + cursor.getInt(4)
+ ')');
                registers.add(cursor.getString(2));
                cursor.moveToNext();
                ctr++;
            }
            if (ctr == 0) {
                Toast.makeText(getBaseContext(), "No Students Found",
Toast.LENGTH_LONG).show();
            }
            Log.d("Attendance", "Got " + ctr + " students");
        }
        adapter = new ListAdapter(thisActivity, names, registers);
        listView.setAdapter(adapter);
    }
}


package assistant.genuinecoder.s_assistant.main.database;

import android.app.Activity;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.util.Log;
import android.widget.Toast;

public class DatabaseHandler {
    Activity activity;
    private SQLiteDatabase database;

    public DatabaseHandler(Activity activity) {
        this.activity = activity;
        database = activity.openOrCreateDatabase("ASSIST",
activity.MODE_PRIVATE, null);
        createTable();
    }

    public void createTable() {
        try {
            String qu = "CREATE TABLE IF NOT EXISTS STUDENT(name
varchar(1000)," +
                    "cl varchar(100), " +
                    "regno varchar(100) primary key, contact
varchar(100),roll integer);";
            database.execSQL(qu);
        } catch (Exception e) {
```

```
            Toast.makeText(activity, "Error Occured for create table",
Toast.LENGTH_LONG).show();
        }
        try {
String qu = "CREATE TABLE IF NOT EXISTS ATTENDANCE(datex date," +
                    "hour int, " +
                    "register varchar(100) ,isPresent boolean);";
            database.execSQL(qu);
        } catch (Exception e) {
            Toast.makeText(activity, "Error Occured for create table",
Toast.LENGTH_LONG).show();
        }

        try {
            String qu = "CREATE TABLE IF NOT EXISTS NOTES(title
varchar(100) not null," +
                    "body varchar(10000), cls varchar(1000), sub
varchar(1000) ,datex TIMESTAMP default CURRENT_TIMESTAMP);";
            database.execSQL(qu);
        } catch (Exception e) {
            Toast.makeText(activity, "Error Occured for create table",
Toast.LENGTH_LONG).show();
        }

        try {
            String qu = "CREATE TABLE IF NOT EXISTS SCHEDULE(cl
varchar(100),subject varchar(1000)," +
                    "timex time, day_week varchar(100));";
            database.execSQL(qu);
        } catch (Exception e) {
            Toast.makeText(activity, "Error Occured for create table",
Toast.LENGTH_LONG).show();
        }
    }

    public boolean execAction(String qu) {
        Log.i("DatabaseHandler", qu);
        try {
            database.execSQL(qu);
        } catch (Exception e) {
            Log.e("DatabaseHandler", qu);
            Toast.makeText(activity, "Error Occured for execAction",
Toast.LENGTH_LONG).show();
            return false;
        }
        return true;
    }

    public Cursor execQuery(String qu) {
        try {
            return database.rawQuery(qu, null);
        } catch (Exception e) {
            Log.e("DatabaseHandler", qu);
            Toast.makeText(activity,"Error Occurred for
execAction",Toast.LENGTH_LONG).show();
```

```
        }
        return null;
    }
}
package assistant.genuinecoder.s_assistant.main.notes;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.database.Cursor;
import android.os.Bundle;
import android.support.design.widget.FloatingActionButton;
import android.support.v7.app.AppCompatActivity;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;

import java.util.ArrayList;

import assistant.genuinecoder.s_assistant.R;
import assistant.genuinecoder.s_assistant.main.AppBase;

public class NoteActivity extends AppCompatActivity implements
ListView.OnItemClickListener, ListView.OnItemLongClickListener {

    ListView listView;
    ArrayAdapter adapter;
    ArrayList titles;
    ArrayList contents;
    Activity activity = this;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_note);

        titles = new ArrayList();
        contents = new ArrayList();

        FloatingActionButton fab = (FloatingActionButton)
findViewById(R.id.fab_Note);
        assert fab != null;
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent launchIntent = new Intent(activity,
NoteCreate.class);
                startActivity(launchIntent);
            }
```

```
        });

        listView = (ListView) findViewById(R.id.noteList);
        loadNotes();
listView.setOnItemClickListener(this);
        listView.setOnItemLongClickListener(this);
    }

    private void loadNotes() {
        titles.clear();
        contents.clear();
        String qu = "SELECT * FROM NOTES";
        Cursor cursor = AppBase.handler.execQuery(qu);
        if (cursor == null || cursor.getCount() == 0) {
            Toast.makeText(getBaseContext(), "No Notes Found",
Toast.LENGTH_LONG).show();
        } else {
            cursor.moveToFirst();
            while (!cursor.isAfterLast()) {
                titles.add(cursor.getString(0));
                contents.add(cursor.getString(1));
                cursor.moveToNext();
            }
            adapter = new ArrayAdapter(this,
android.R.layout.simple_list_item_1, titles);
            listView.setAdapter(adapter);
        }
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        super.onCreateOptionsMenu(menu);
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.note_menu, menu);
        return true;
    }

    @Override
    public void onItemClick(AdapterView<?> parent, View view, int
position, long id) {
        AlertDialog.Builder alert = new AlertDialog.Builder(activity);
        alert.setTitle(titles.get(position).toString());
        alert.setMessage(contents.get(position).toString());
        alert.setPositiveButton("OK", null);
        alert.show();
    }

    @Override
    public boolean onItemLongClick(AdapterView<?> parent, View view,
int position, long id) {
        AlertDialog.Builder alert = new AlertDialog.Builder(activity);
        final String title = titles.get(position).toString();
        final String body = contents.get(position).toString();
        alert.setTitle("Delete ?");
        alert.setMessage("Do you want to delete this note ?");
```