МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационные системы и технологии

Специальность 1-98 01 03 “Программное обеспечение информационной

безопасности мобильных систем”

Специализация Инженер-программист

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

«Мобильное приложение для учёта расходов “WalletLite”»

Выполнил студент Скорина Михаил Александрович

(Ф.И.О.)

Руководитель проекта ст. преподаватель Северинчик Н.А. (учен. степень, звание, должность, Ф.И.О., подпись)

Заведующий кафедрой к.т.н., доц. Смелов В.В.

(учен. степень, звание, должность, Ф.И.О., подпись)

Консультант: ст. преподаватель Северинчик Н.А. (учен. степень, звание, должность, Ф.И.О., подпись)

Нормоконтролер: ст. преподаватель Северинчик Н.А. (учен. степень, звание, должность, Ф.И.О., подпись)

Курсовой проект защищен с оценкой

Минск 2020

**СОДЕРЖАНИЕ**

[Введение 3](#__RefHeading___Toc5815_1724808274)

[1 Обзор аналогов приложений «WalletLite» 4](#__RefHeading___Toc7824_1511321520)

[1.1 Приложение «Wallet – Money,Budget, Finance & Expense Tracker» 4](#__RefHeading___Toc7826_1511321520)

[1.2 Приложение «Дзен-мани: учет расходов и доходов» 5](#__RefHeading___Toc7828_1511321520)

[1.3 Выводы по главе «Обзор аналогов приложений ”WalletLite”» 6](#__RefHeading___Toc7830_1511321520)

[2 Проектирование программного продукта 7](#__RefHeading___Toc7832_1511321520)

[2.1 Проектирование мобильного приложения 7](#__RefHeading___Toc7834_1511321520)

[2.2 Проектирование базы данных 9](#__RefHeading___Toc7836_1511321520)

[2.3 Проектирование сервера 11](#__RefHeading___Toc7838_1511321520)

[2.4 Проектирование веб-приложения 11](#__RefHeading___Toc5999_1724808274)

[2.5 Выводы по главе «Проектирование программного продукта» 11](#__RefHeading___Toc7840_1511321520)

[3 Программная реализация приложений 13](#__RefHeading___Toc7842_1511321520)

[3.1 Технические средства для разработки 13](#__RefHeading___Toc7844_1511321520)

[3.2 Разработка баз данных 13](#__RefHeading___Toc7846_1511321520)

[3.3 Разработка мобильного приложения 14](#__RefHeading___Toc7848_1511321520)

[3.4 Разработка сервера 16](#__RefHeading___Toc6001_1724808274)

[3.5 Разработка веб-приложения 17](#__RefHeading___Toc6003_1724808274)

[3.6 Вывод по главе «Программная реализация приложений» 18](#__RefHeading___Toc7852_1511321520)

[4. Руководство пользователя 19](#__RefHeading___Toc7854_1511321520)

[4.1 Выводы по главе «Руководство пользователя» 29](#__RefHeading___Toc7856_1511321520)

[ЗАКЛЮЧЕНИЕ 30](#__RefHeading___Toc7858_1511321520)

[Список использованных источников 31](#__RefHeading___Toc7860_1511321520)

[ПРИЛОЖЕНИЕ А 31](#__RefHeading___Toc7862_1511321520)

[ПРИЛОЖЕНИЕ Б 32](#__RefHeading___Toc7864_1511321520)

[ПРИЛОЖЕНИЕ В 34](#__RefHeading___Toc7866_1511321520)

# **Введение**

Каждый человек для осуществления своих потребностей и желаний нуждается в финансовых средствах, но многие люди не контролируют куда и сколько денег ушло за определенный период, и даже если люди ведут учёт расходов, то это требуется постоянно сохранять чеки и после добавлять уже дома в какую-то программу расходов или сохранять на бумажном носителе.

Так как в современном мире почти у каждого человека есть смартфон или планшет, который он носит постоянно чтобы оставаться на связи в социальных сетях, развлечений и поиска информации в любой момент, то появились приложения удовлетворяющие и такие потребности как ведение расходов.

Целью курсовой работы будет разработка такого приложения, которое позволит осуществить хранение информации о платежах совершаемых пользователем. Так же пользователь сможет просматривать статистику по совершаемым платежам с разделением по категориям трат. Для удобства пользователя он может синхронизировать свои данные с сервером и после используя другое устройство получить их, или при потере или порче устройства не беспокоиться о том что он может потерять информацию о своих платежах. Для обеспечения безопасности приложение шифрует данные пользовательским паролем и при открытии приложения запрашивает его. В случае необходимости пользователь может удалить все свои платежи в несколько нажатий. Так же пользователь может экспортировать данные для их последующей обработки или импорта в другую копию приложения.Для этого требуется решить следующие задачи:

* Изучить аналогичные приложение и литературные источники о реализации подобных решений
* Спроектировать приложение для учета расходов
* Реализовать приложение для учёта расходов
* Написать руководство пользователя

Данное приложение должно помочь пользователям сохранять информацию от своих тратах, получать статистику, а главное обезопасить информацию от чужих глаз и преступников которые могут заполучить устройство пользователя. Безопасность данных в современном мире является очень серьёзной проблемой, так как многие программные средства которые мы используем постоянно не обеспечивают достаточной защиты.

# **1 Обзор аналогов приложений «WalletLite»**

# **1.1 Приложение «Wallet – Money,Budget, Finance & Expense Tracker»**

Данное приложение на сегодняшний день является популярным программным средством ведения учёта расходов. Доступ к функционалу осуществляется только после регистрации. Основная функция－ Добавление платежей пользователя и демонстрация статистики.

Приложение предоставлено на скриншоте ниже (Рисунок 1.1.1-Приложение «Wallet – Money,Budget, Finance & Expense Tracker»).

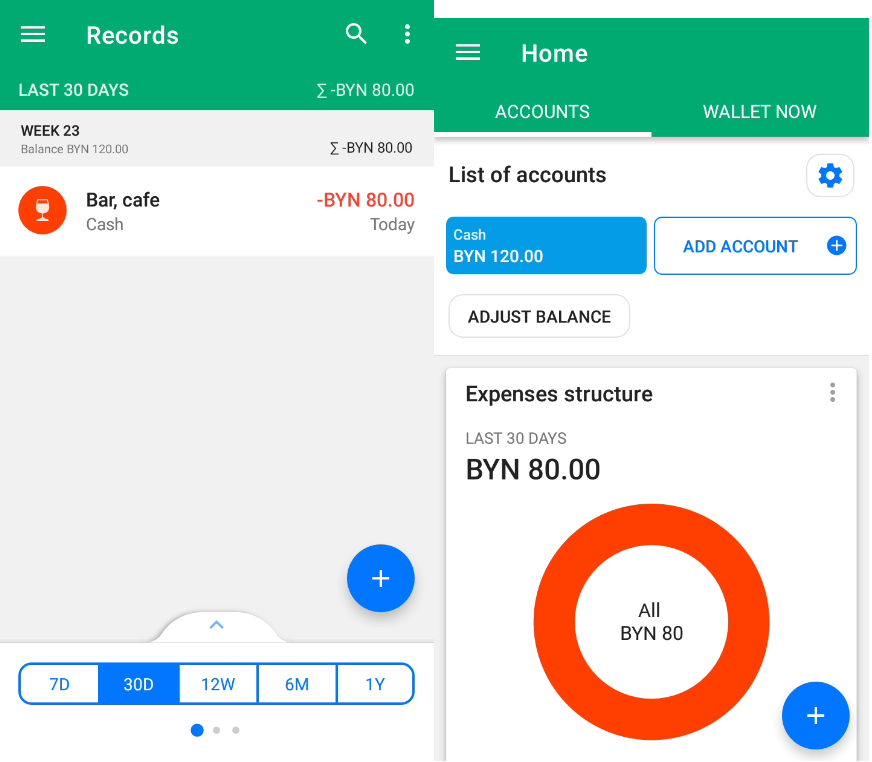


Рисунок 1.1.1 –Приложение «Wallet – Money,Budget, Finance & Expense Tracker»

Также есть возможность синхронизации и получения данных на другому устройстве и установки жизни кошелька.

# **1.2 Приложение «Дзен-мани: учет расходов и доходов»**

В данном приложении в дополнение к функция предыдущего можно ещё добавлять платежи которые пришли в смс, и выполнять синхронизацию с банкингом.

Интерфейс приложения представлен на скриншоте ниже (Рисунок 1.2.1-Приложение «Дзен-мани: учет расходов и доходов»).

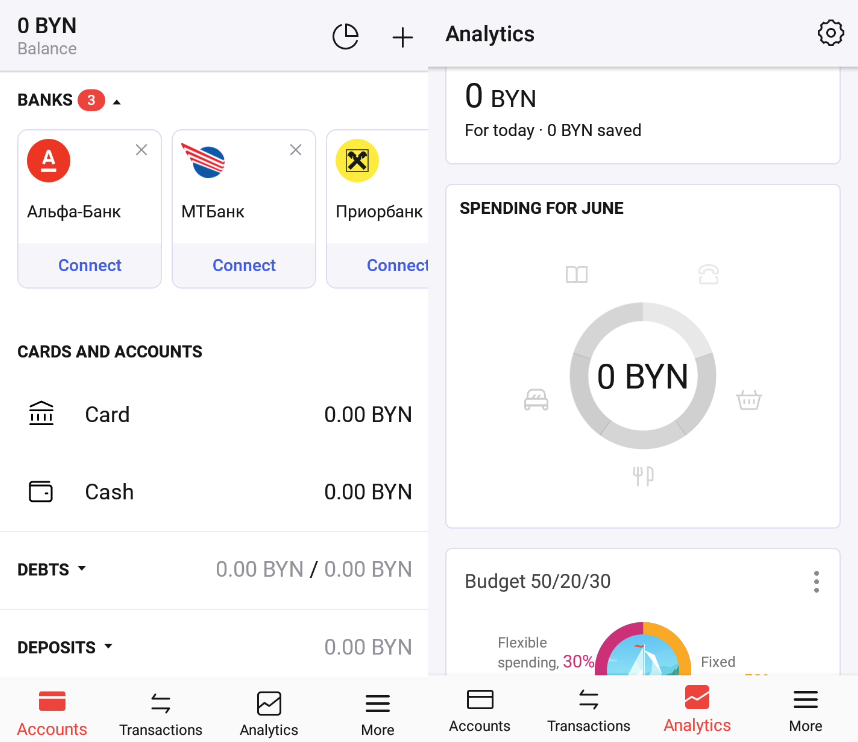


Рисунок 1.2.1 – Приложение «Дзен-мани: учет расходов и доходов»

Приложение имеет богатый функционал, в том числе вход в приложение по пин-коду или отпечатку пальца, можно добавлять различные типы карт а так же синхронизировать данные с банков, но база данных не шифруется, и нет возможности установить пароль вместо отпечатка пальца и пинкода.

# **1.3 Выводы по главе «Обзор аналогов приложений ”WalletLite”»**

В данной главе были показаны два наиболее популярных приложения, которые являются аналогами приложения «WalletLite»: «Wallet – Money,Budget, Finance & Expense Tracker» и «Дзен-мани: учет расходов и доходов». В результате подытожим сравнения.

Мое приложение является прототипом “Wallet – Money,Budget, Finance & Expense Tracker”. У него достаточно удобный и интуитивно понятный дизайн, богатый функционал. В отличие от этого приложения, в моем программном средстве есть шифрование базы данных и функция экстренного сброса данных и экпорт бд и json и нет обязательной регистрации.

К плюсам приложения “Wallet – Money,Budget, Finance & Expense Tracker” относится режим тёмного дизайна, и богатое число типов счетов.

К плюсам приложения “Дзен-мани: учет расходов и доходов” относится наличие возможности установить пин-код на приложение и синхронизация с банкингом.Так же у приложения крайне богатый функционал, есть возможность добавление регулярных платежей, которые автоматически будут добавлены в список платежей при наступлении времени.

К минусам относится отсутствие шифрования базы данных и обязательная регистрация, так же нет экспорта в нужный формат, перенос данных организован только при помощи синхронизации на втором устройстве.

# **2 Проектирование программного продукта**

# **2.1 Проектирование мобильного приложения**

В данной курсовой работе необходимо было разработать приложение, позволяющее пользователям осуществлять добавление и хранение информации о счетах и платежах, а так же возможность экспорта этих данных.

Если пользователь первый раз запускает приложение на устройстве, то открывается экран с полем для ввода пароля которым будет шифроваться база данных. У пользователя есть возможность после экспортировать эту базу данных или json файл.

После ввода пароля пользователь попадает на главную активность приложения, где может уже добавить платёж, но при попытке добавить платёж при отсутствии счетов он получит сообщение что надо с начало добавить счёт

После пользователь открывает боковое меню и выбирает пункт с названием «счет», после там нажимает кнопку в виде круга и попадает на новую активность для добавления новой записи счета, после заполнения полей и нажатия кнопки добавить пользователь переходит назад на вкладку с платежами и добавляет платёж.

После пользователь может найти данные статистики во вкладке «Статистика», там он может выбрать период и валюту платежей, после выведется информация о платежах.

Пользователь может перейти на вкладку «Экпорт\Импорт» и получить копию базы данных, экспорт json или импортировать базу данных.

Затем пользователь может перейти на вкладку «Синхронизация», ввести логин и пароль и нажать «Вход» или «Регистрация».

После входа можно выполнить проверку соединения с сервером нажав на кнопку «Проверка соединения с сервером», для синхронизации нажать «Синхронизировать».

Для экстренного сброса приложения можно воспользоваться кнопкой экстренного сброса приложения которая находится в вверху с правка в меню с троеточием, после будут удаленны данные пользователя как на клиенте так и на сервере.

Так же пользователю доступен просмотр всех платежей и счетов в веб интерфейсе. Для этого достаточно перейти на определенную страницу в браузере и получить форму с логин и паролем и кнопкой «получить данные»

После ввода данных и нажатия кнопки пользователь увидит ниже данные из своей учетной записи.

Главная особенность спокойной работы в автономной режиме и перенос данных с использованием других носителей.

Для импорта данных надо получить файл который вы экспортировали ранее вы должны указать путь к вашему файлу с базой данных.

Можно ознакомиться с UML-диаграммой, а именно диаграммой вариантов использования. Она отображает функциональность программного продукта с точки зрения получения значимого результата для пользователя. (Рисунок 2.1.)

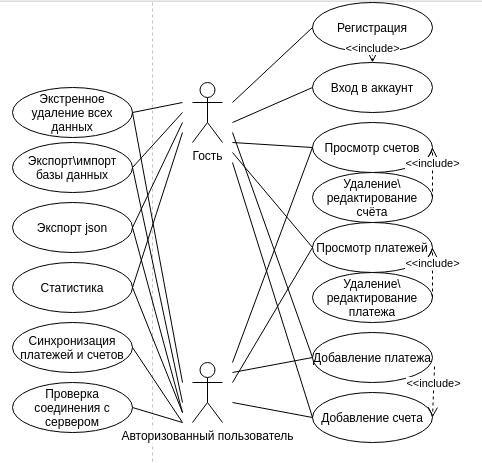


Рисунок 2.1.1 – Диаграмма вариантов использования

Таким образом приложение должно выполнять функции:

* Регистрация пользователя;
* Ввод данных пользователя;
* Авторизация пользователя;
* Проверка подключения к серверу
* Синхронизация данных пользователя;
* Вход в приложение по паролю;
* Добавление, редактирование и удаление счетов;
* Добавление, редактирование и удаление платежей;
* Экстренный сброс данных;
* Просмотр статистики по платежам;
* Экпорт данных в json;
* Экпорт базы данных;
* Импорт базы данных.

# **2.2 Проектирование базы данных**

В базе данных данного приложения созданы для использования 3 таблицы. База генерируется API приложением, в использованием sqlAchemy. Ниже представлена характеристика таблиц.

В таблице «users» содержится информация для авторизации пользователей. Столбцы данной таблицы:

* id – уникальный идентификатор пользователя, целочисленный тип данных, является первичным ключом;
* email – строковый тип данных,максимальной длины, уникальное значение;
* passhash – строковый тип данных, максимальной длины, содержит хеш пароля(Приложение В);
* blocked – логический тип данных, указывает на состояние блокировки учетной записи пользователя;

В таблице «wallets» содержится информация о счетах пользователей. Столбцы данной таблицы:

* id – уникальный идентификатор записи,целочисленный тип данных, первичный ключ;
* userId – вторичный ключ, ссылается на первичный ключ таблицы users, служит для связи этих таблиц;
* walletid – целочисленный тип данных, указывает на локальный идентификатор счета пользователя;
* name – содержит описание(номер) счета, строковый тип данных, максимальной длины;
* typemoney – содержит тип хранимой валюты счета пользователя, строковый тип данных;
* money – содержит количество хранимых средств на счете, строковый тип данных, максимальной длины;
* modifieddate – содержит дату последнего изменения данных счета пользователя, целочисленный тип данных;
* Followers – содержит количество подписчиков пользователя, целочисленный тип данных, неуникальное значение, не допускается null;
* Follows – содержит количество подписок пользователя, целочисленный тип данных, неуникальное значение, не допускается null;

В таблице «payments» содержатся платежи пользователей. Столбцы данной таблицы:

* id – уникальный идентификатор записи,целочисленный тип данных, первичный ключ;
* userId – вторичный ключ, ссылается на первичный ключ таблицы users, служит для связи этих таблиц;
* walletid – целочисленный тип данных, указывает на локальный идентификатор счета пользователя;
* paymentid – целочисленный тип данных, указывает на локальный идентификатор платежа пользователя;
* type – строковый тип данных, указывает какой тип платежа;
* summary – информация о платеже, строковый тип данных;
* money – содержит количество потраченных средств, строковый тип данных, максимальной длины;
* date – содержит дату добавления , строковый тип данных, максимальной длины;
* modifieddate – содержит дату последнего изменения данных платежа пользователя, целочисленный тип данных;

Схема базы данных представлена на Рисунок 2.2.1.

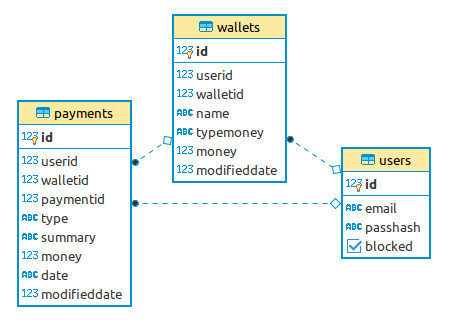


Рисунок 2.2.1 – Схема базы данных

Таким образом, таблица users содержит ключевое значение Id, которое используется в остальных таблицах для доступа к данным пользователя. Остальные таблицы хранят платежи и счета пользователя.

# **2.3 Проектирование сервера**

Для синхронизации двух и более клиентов необходимо серверное приложение. Серверная часть приложения должна предоставлять WEB API для доступа к каждой сущности в базе данных. Передаваемые или получаемые данные от сервера должны передаваться в формате JSON.

Серверное приложение должно быть построено на основе архитектурного стиля REST и будет использоваться для взаимодействия компонентов распределённого приложения в сети.

Сервер содержит в себе реализованные сервисы для взаимодействия с каждой сущностью базы данных.

При обращении к серверу по заданному маршруту и определенному HTTP(HTTPS) методу сервер будет определять с какой сущностью базы данных необходимо взаимодействовать и какие действия над данными необходимо будет произвести.

Используя HTTP метод POST сервер будет получать данные от пользователя в формате json, обрабатывать и передавать ответ так же в формате json, это основной метод который обрабатывает сервер.

Сервер будет получать данные для аутентификации, авторизации, синхронизации изменений пользовательских данных с сервером.

# **2.4 Проектирование веб-приложения**

В случае не возможности воспользоваться мобильным устройством у пользователя должна быть способ получить данные с сервера. Для этого должно быть разработано веб-приложение которое позволяет через браузер выполнить переход на определенную страницу заполнить данные для аутентификации и авторизации и получить информацию о счетах и платежах. Для этого требуется создать страницу которая содержит форму для ввода логина и пароля пользователя, а так же таблицы в которые должны приходить данные в случае успешной авторизации пользователя, для этого можно на страницы использовать ajax запросы к api.

Передача данных должна происходить по протоколу HTTPS, использоваться метод POST и формат данных должен соответствовать требованиям сервера, тоесть сообщения к серверу должны быть в формате json.

# **2.5 Выводы по главе «Проектирование программного продукта»**

В данной главе было рассмотрено проектирование каждого отдельного компонента приложения.

По части мобильного приложения был определен основной функционал приложения, также были разработаны диаграмма вариантов использования приложения и описаны варианты использования программного средства.

Была также спроектирована база данных, где были определены необходимые таблицы, охарактеризованы поля каждой из них, определены типы данных и ограничения целостности. Также были установлены связи между таблицами, заключающиеся в создании первичных и внешних ключей.

При проектировании серверного приложения был определен архитектурный стиль взаимодействия между клиентом и сервером, формат передачи данных, обозначили HTTP методы которые будут использованы в работе с серверной частью.

При проектировании веб приложения был определен функционал, а именно определена возможность передачи данных на сервер для аутентификации, получение токена и получения данных о платежах и счетах пользователя по токену с использованием запросов на сервер по протоколу HTTP с методом POST и передача информации в формате json.

# **3 Программная реализация приложений**

# **3.1 Технические средства для разработки**

Для разработки приложений курсового проекта были использованы следующие технические средства:

SQLite – компактная встраиваемая реляционная база данных.

PgAdmin 4 – менеджер управления реляционными БД Postgres.

Postresql Server – СУБД Postgres.

Uvicorn – ASGI сервер.

FastAPI-фреймворк разработки API

ASGI- Интерфейс асинхронного серверного шлюза

SQL – декларативный язык программирования, применяемый для создания, модификации, и управления данными в РБД.

Android Studio – интегрированная среда разработки для работы с платформой Android.

Visual Studio Code – кроссплатформенная среда разработки программного обеспечения.

Java – строго типизированный объектно-ориентированный язык программирования.

Python — высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода.

Postman – набор инструментов тестирования API.

Vue.js — Прогрессивный JavaScript-фреймворк

JavaScript — прототипно-ориентированный сценарный язык программирования. Является реализацией языка ECMAScript (стандарт ECMA-262). JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений.

Vue.js — Прогрессивный JavaScript-фреймворк

DBeaver — Бесплатный универсальный инструмент для баз данных

# **3.2 Разработка баз данных**

Для хранения данных на телефоне будет использоваться встраиваемая СУБД SQLite. Для сервера будет использована СУБД Postgres.

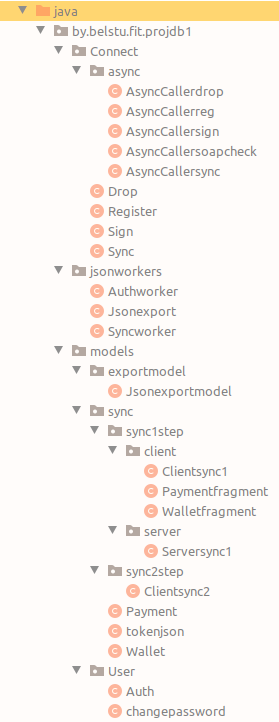
Код создания структуры мобильной базы данных представлен в Приложении Б, серверной в Приложении В.

Мобильная база данных содержит в себе информацию об платежах и счетах пользователя и используется при синхронизации пользователя с серверной СУБД. База данных шифруется с помощью Sqlite Chipler.

# **3.3 Разработка мобильного приложения**

Мобильное приложение было реализовано в среде разработки – Android Studio 3.6.3. Для написания приложения был использован язык программирования Java.

Графическая структура проекта представлена на Рисунок 3.3.1.



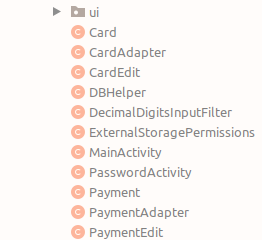


Рисунок 3.3.1 – Графическая структура проекта

Пакет Connect хранит в себе классы, которые отвечают за синхронизацию с сервером (см. Таблица 3.3.1).

**Таблица 3.3.1 – Содержание пакета Connect**

|  |  |  |
| --- | --- | --- |
| Класс Register – Запрос на регистрацию на сервере | | |
| Connect() | Метод подключения к серверу и передачу json серверу и получения json с сервера | |
| Info() | Проверка ответа сервера и получения содержимого ответа | |
| Класс Drop – Удаление данных с сервера | | |
| Connect() | Метод подключения к серверу и передачу json серверу и получения json с сервера | |
| Info() | Проверка ответа сервера и получения содержимого ответа | |
| Класс Sign – получение токена от сервера | | |
| Connect() | | Метод подключения к серверу и передачу json серверу и получения json с сервера |
| Info() | | Проверка ответа сервера и получения содержимого ответа |
| Класс Sync – синхронизация с сервером | | |
| Connect() | | Метод подключения к серверу и передачу json серверу и получения json с сервера |

Ознакомиться с диаграммой классов разработанного приложения можно в приложении А.

# **3.4 Разработка сервера**

Сервер было реализовано в среде разработки – Visual Studio Code. Для написания приложения был использован FastAPI фреймвор на языке Python.

Графическая структура проекта представлена на Рисунок 3.4.1.

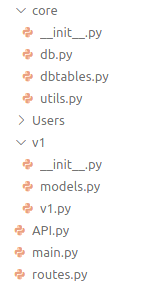


Рисунок 3.4.1 – Графическая структура проекта

Папка core хранит в себе классы конфигурации бд,таблиц и функции используемые в API при обработки запросов(см. Таблица 3.4.1).

**Таблица 3.4.1 – Содержание папки Core**

|  |  |
| --- | --- |
| Класс db – содержит код доступа к серверу | |
| SQLALCHEMY\_DATABASE\_URI | Содержит строку подключения |
| engine | Создание объекта SqlAchemy в который передается URI |
| SessionLocal | Создание сессии |
| Класс dbtables – модели таблиц базы данных | |
| users | Модель таблицы с пользователями |
| wallets | Модель таблицы со счетами |
| payments | Модель таблицы с платежами |
| Класс utils – функции работы с бд и обработки данных | |
| get\_db() | Получения объекта бд |
| gettoken() | Генерация токена |
| checktoken() | Проверка токена |
| sync1() | Первый этап синхронизации |
| sync2() | Второй этап синхронизации |
| syncdrop() | Функция удаления всех данных пользователя |
| webdata() | Возрат по токену платежей и счетов |
| XmlResponse() | Возращает xml |
| XmlBody() | Обработка xml |

Были расмотренны классы которые выполняют основые функции API приложения.

# **3.5 Разработка веб-приложения**

Web-приложение было реализовано в среде разработки – Visual Studio Code. Для написания приложения был использован vue.js фреймворк на языке JavaScript.

Графическая структура проекта представлена на Рисунок 3.5.2.

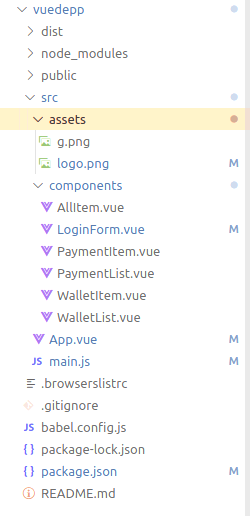


Рисунок 3.5.2 – Графическая структура проекта

Папка components хранит в себе классы компонентов страницы веб приложения, представляют собой фрагменты страницы(см. Таблица 3.5.2).

**Таблица 3.5.2 – Содержание папки components**

|  |  |
| --- | --- |
| Класс AllItem – содержит элементы для генерации таблиц счета и платежей | |
| WalletList | Является элементом генерирующим таблицу счетов |
| PaymentList | Является элементом генерирующим таблицу платежей |
| Класс LoginForm – Содержит форму аутентификации | |
| OnSubmit() | Метод передачи данных с формы в родительский класс |
| Класс WalletList – содержит код генерации таблицы и элементы | |
| WalletItem | Элемент представляющий один платёж |

Были расмотренны классы которые выполняют основые функции API приложения.

# **3.6 Вывод по главе «Программная реализация приложений»**

В данной главе было рассмотрена разработка каждого отдельного компонента приложения.

На первоначальном этапе были рассмотрены технологии и языки, которые применены для разработки курсового проекта. Были использованы языки Python,Java,JavaScript и технологии vue.js,node.js,Fast API,uvicorn.

Вторым этапом была разработка структуры базы данных на стороне сервера и клиента. Используются СУБД Postgres на стороне сервера и мобильная СУБД Sqlite со стороны клиента.

Третьим этапом стала разработка мобильного приложения, где была представлена графическая структура приложения, рассмотрен класс и методы, в частности. Мобильное приложение написано на языке Java и поддерживает шифрование базы данных а так же возможность экспорта данных как на сервер так и файл.

Четвертый этап – это разработка серверного приложения, где была представлена графическая структура приложения, рассмотрен класс и методы сервера. Данный сервер разработан на FastAPI фреймворке работающем на uvicorn сервере который написан на языке Python.

Пятый этап – это разработка веб-приложения, где была представлена графическая структура приложения, рассмотрены методы приложения. Данное веб-приложение разработано с использование vue.js, и работает на Node.js сервере Serve.

# **4. Руководство пользователя**

Для того, чтобы установить приложение на телефон, Вам необходимо иметь версию Android 5.1 и выше(версию API 19 и более). Версию Вашего телефона можно узнать в настройках.

Если Ваша версия удовлетворяет требованиям, то можно устанавливать приложение на телефон.

После установки приложения в списке приложений появится иконка(использовалась стандартная иконка приложения) и подпись “WalletLite” (рис.4.1)

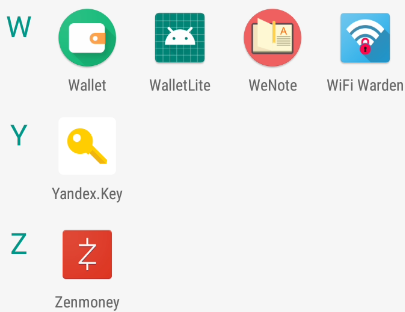


Рисунок 4.1 – Иконка приложения

Изначально открывается экран, на котором находится форма ввода пароля(рис.4.2).

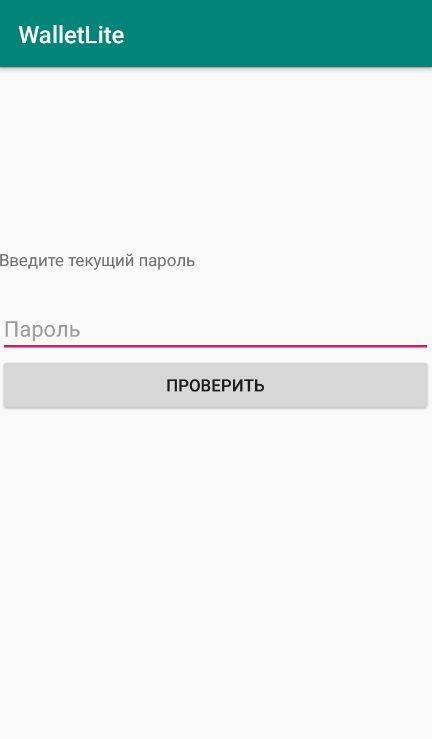


Рисунок 4.2 – Форма ввода пароля

Перед первым использованием приложения Вам необходимо ввести пароль приложения который будет использоваться всегда. Для этого нужно нажать на поле ввода, ввести пароль и нажать кнопку ниже, вы будете перенаправлены на главную активность.

При повторном запуске приложения вам потребуется вводить данный пароль, и нажимать кнопку ниже.

Если вы свернёте приложение или заблокируете устройство вам потребуется повторно вводить пароль на этой активности для перехода на главную активность приложения.

Если вам потребуется в последующем экспортировать базу данных, то при импорте пароль должен совпадать, либо вам придётся переоткрыть приложение и вводить новый пароль.

Не стоит забывать что база данных шифруется данным паролем и в случае если вы забудете данные от приложения вам может помочь только ранее синхронизированные данные на сервере.

После перехода на новую активность вам доступно представление добавления платежей, но предварительно вам надо создать первый счёт иначе программа вам не разрешит создавать платёж(рис4.3)



Рисунок 4.3 – Главный экран

С главного экрана вам доступна функция экстренного сброса всех данных, для этого вам потребуется нажать на троеточие сверху и выбрать «Экстренный сброс»

Для перехода к другим экранам приложения вам нужно вызвать боковое меню, для этого проведите от левого края экрана в право и вы получить боковое меню(рис.4.4):

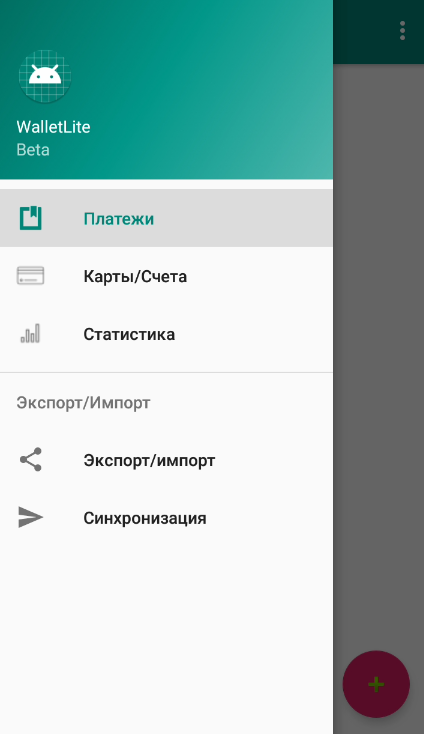


Рисунок 4.4 – Боковое меню приложения

Боковое меню даст вам возможность получить доступ ко всему оставшемуся функционалу приложения. Такому как «Карты\Счета», «Статистика», «Экспорт\импорт» и «Синхронизация».

Далее будет рассмотрено что ещё доступно из этого приложения с использованием этого бокового меню.

Для доступа к счетам нам надо с помощью бокового меню перейти к «Карты/Счета», здесь мы можем найти снизу кнопку в виде кружка красного розового цвета с знаком плюса в середине, который мы можем использовать для добавления нового счета, который в свою очередь будет использоваться для платежей(рис.4.5):



Рисунок 4.5 – Представление счетов

Далее мы можем нажать на эту круглую кнопку для добавления данных и рассмотреть окно добавления элемента(рис.4.6):

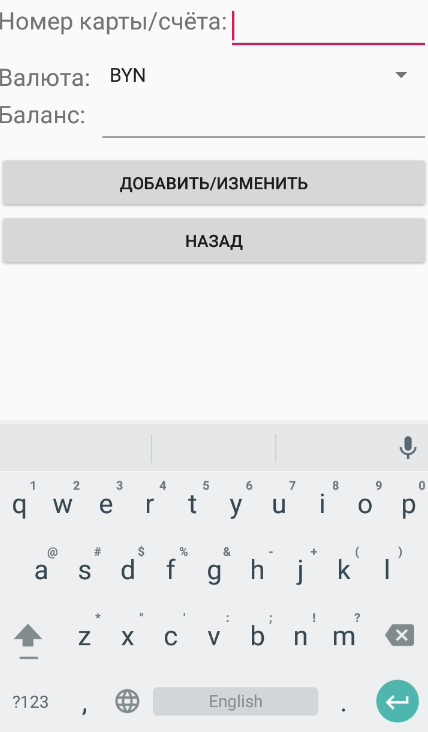


Рисунок 4.6 – Добавление счёта

Здесь мы можем найти несколько полей. Первое поле содержит номер счёта ,второе валюту, а третье баланс который есть на счету. После нажмите «добавить\изменить» чтобы новой элемент добавился.

Добавленный элемент можно отредактировать или удалить, для этого достаточно зажать на появившейся записи пальцем и дождаться появления контекстного меню, после этого вы можете выбрать удаление и изменение записи(рис.4.7):

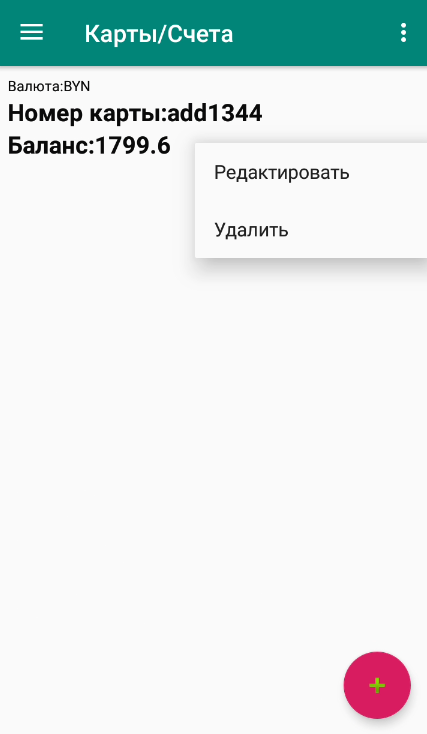


Рисунок 4.7 – контекстное меню при нажатии на элемент

Если вы выберите «Редактировать» то получить окно редактирования рассмотренное чуть ранее. В этом окне будут заполнены поля теми данными которые были в элементе.

Соответственно рассмотрим и окно редактирования платежей(рис.4.8):

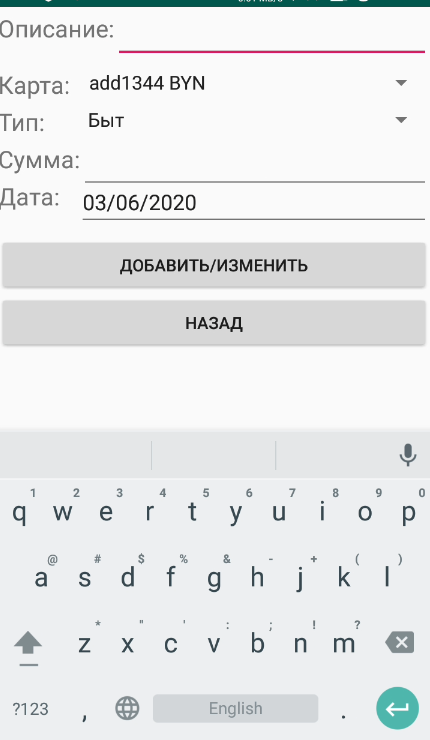


Рисунок 4.8 – Редактирование платежа

Здесь мы видим поле «Описание», которое предназначено для описания платежа, например для какие нужд он был совершен. Далее поле «Карта» оно содержит список из добавленных ранее карт. Поле «тип» содержит тип нашей траты, например быт или еда, следующее «Сумма» это то сколько мы потратили денег с данного счёта на наши покупки, и наконец-то «Дата» это день совершения платежа, используется при расчёте статистики.

Теперь рассмотрим саму статистику, для доступа к ней нам надо в боковом меню выбрать соответствующий пункт(рис.4.9):

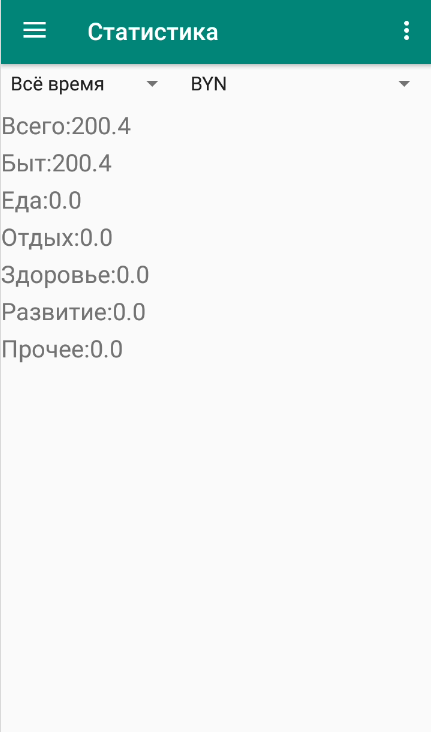


Рисунок 4.9 – Статистика по платежам

Мы можем используя выпадающий список выбрать период за который хотим получить статистику по затратам. Так же мы можем выбрать валюту которую мы использовали при платеже и расчет будет произведен конкретно по счетам использующих эту валюту.

При удалении платежей статистика этих платежах не ведётся и расчёт ведется только по существующим платежам.

Теперь рассмотрим вкладку «Экспорт\Импорт»(рис.4.10):

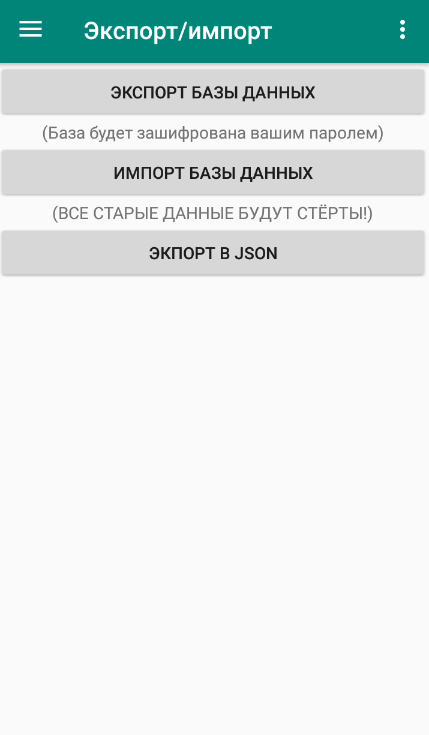


Рисунок 4.10 – Экспорт\импорт

При нажатии на кнопку «Экспорт базы данных» приложение сохранит копию нашей базы в данных на карту в памяти в папку Walletlite где будет лежать файл с название Walletlite.db

При нажатии на кнопку «Импорт базы данных» откроется окно где нужно выбрать расположение файла Walletlite.db и после база данных будет заменена на выбранную.

И последняя кнопка позволяет нам экспортировать данные нашей базы данных в json формат. Файл будет лежать в папке Walletlite на карте памяти.

И теперь рассмотрим последнее, синхронизация осуществляется на вкладке «Синхронизация»(рис.4.11)

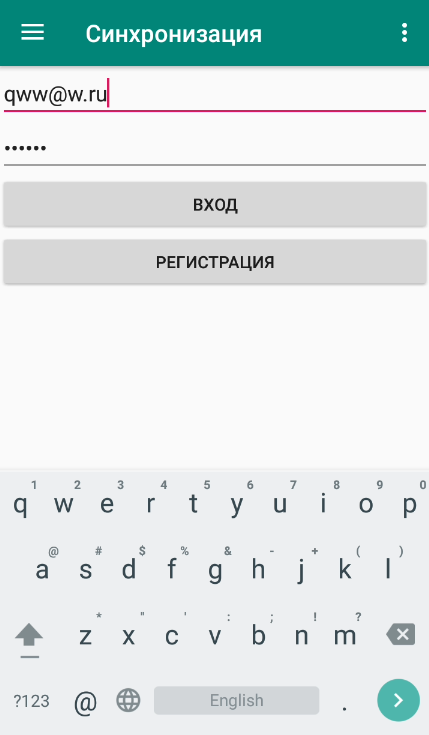


Рисунок 4.11 – Регистрация и вход

В этом окне мы вводим данные пользователя, а именно эмайл в качестве логина и пароль , после нажимай кнопку «Регистрация», после получения ответа нажимаем «Вход».

Теперь после входа мы можем получить доступ к серверу для синхронизации наших данных(рис.4.12). Для просмотра публикации нужно нажать на фото.

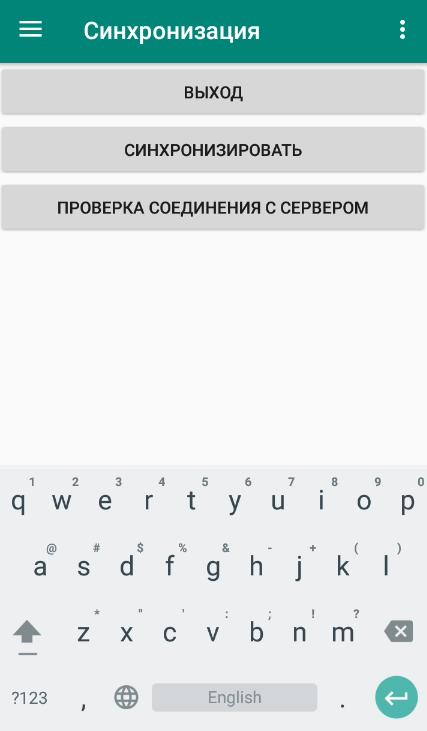


Рисунок 4.12 – Синхронизация

Теперь мы можем выполнить одно и трех действий. Выйти из аккаунта что попасть на форму входа нажав кнопку «Выход», проверить работу сервера нажав «Проверка соединения с сервером» и последнее это собственно произвести синхронизацию данных с сервером нажав «Синхронизировать».

# **4.1 Выводы по главе «Руководство пользователя»**

В данной главе было описано руководство пользователя.

Данное руководство позволяет пользователю узнать требования для установки приложения на телефон, а также как использовать все функции приложения. Были предоставлены скриншоты самого приложения для большей наглядности использования.

# **ЗАКЛЮЧЕНИЕ**

Целью моего курсового проекта было разработать приложение для хранения платежей пользователей, при этом максимально защитить данные от воровства при физическом доступе к устройству.

Были решены следующие поставленные задачи:

* Добавление и изменение номеров счетов, платежей.
* Экспорт в json файл.
* Доступ в приложения по паролю.
* Быстрое удаление всей платежей и счетов.
* Экспорт и импорт базы данных (с заменой существующих записей).
* Шифрование базы данных с помощью пароля.
* Статистика расходов по категориям исходя из выбранной валюты.
* Синхронизация данных с сервером (платежи и счета);

Не были решены следующие задачи:

* Импорт из json файла.
* Добавление и удаление категорий платежей.

Причиной не реализованного добавление и удаление категорий платежей является не удобность конечно использования статистики и сложность выбора из большого количества позиций. Импорт json-а не реализован т.к возможна возникновения ситуации что импорт чужого файла будет содержать записи с одинаковый идентификатором элементов платежей или счетов, что вызовет перезапись текущий записей.

В результате было рассмотрено 2 приложения и были выявлены их положительные и отрицательные стороны.

Был спроектировано приложение которое включает в себе компоненты СУБД Postgres, фреймоворк FastAPI,сервер uvicorn,СУБД SQLite, мобильное приложение на Android, vue.js приложение, Node.js сервер Serve.

Данная задача была реализована с помощью следующих технических средств:среда разработки Android Studio и Visual Studio Code, языков программирования Java и Python, javascript.

Были реализованы базы данных на стороне сервера и клиента.

При решении данной задачи была достигнута поставленная цель, был создана база данных для пользователей в которой храниться информация о самих пользователях,их платежах и счетах, разработан API который обрабатывает запросы клиента и позволяет регистрировать пользователя и синхронизировать его данные с базой данных, было так же разработано Android-приложение с интерфейсом который позволяет использовать как локальную базу данных шифруемую паролем так синхронизировать данные с сервером, так же разработано веб-приложение для общение с сервером.

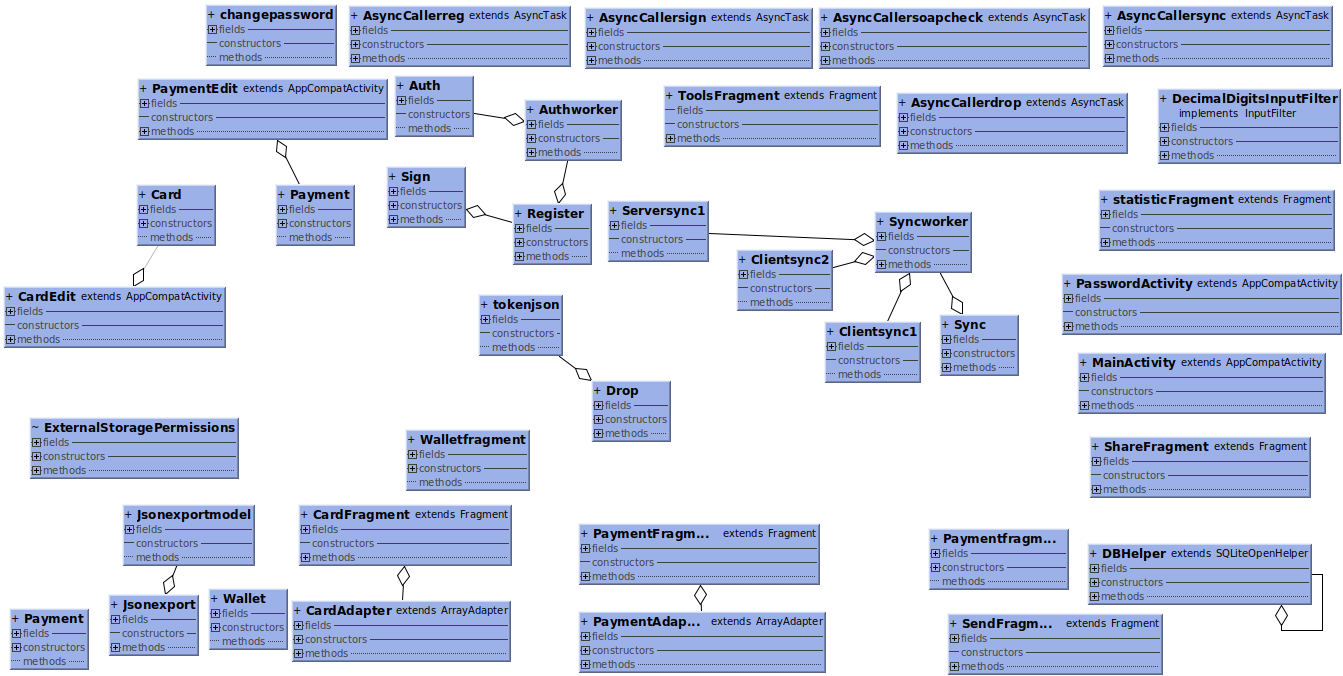
Разработано руководство, позволяющее пользователю узнать требования для установки приложения на телефон, а также как использовать все функции мобильного приложения.

# **Список использованных источников**

1. FastAPI framework, high performance, easy to learn, fast to code, ready for production. SQL (Relational) Databases [Электронный ресурс]. / fastapi.tiangolo.com – Режим доступа: [https://fastapi.tiangolo.com/tutorial/sql-databases/](https://fastapi.tiangolo.com/tutorial/body/) Дата доступа: 12.05.2020

1. FastAPI framework, high performance, easy to learn, fast to code, ready for production. Reguest body [Электронный ресурс]. / fastapi.tiangolo.com – Режим доступа: <https://fastapi.tiangolo.com/tutorial/body/> Дата доступа: 13.05.2020
2. FastAPI framework, high performance, easy to learn, fast to code, ready for production. Deployment [Электронный ресурс]. / fastapi.tiangolo.com – Режим доступа: [https://fastapi.tiangolo.com/deployment/](https://fastapi.tiangolo.com/tutorial/body/) Дата доступа: 13.05.2020
3. Solutions on Stackoverflow. How to Parse a JSON Object In Android [Электронный ресурс] – Режим доступа: <https://stackoverflow.com/>[questions/5566669/how-to-parse-a-json-object-in-android](https://stackoverflow.com/questions/5566669/how-to-parse-a-json-object-in-android) Дата доступа: 15.05.2020 .
4. «Сайт Александра Климова» по Android. Фрагменты [Электронный ресурс] – Режим доступа: <http://developer.alexanderklimov.ru/android/theory/fragments.php> Дата доступа: 18.05.2020
5. Youtube. Учим vue-cli 3 [Электронный ресурс] – Режим доступа: <https://www.youtube.com/watch?v=4O1twLpg-7A> Дата доступа: 20.05.2020
6. Youtube. FastAPI + Pydantic [Электронный ресурс] – Режим доступа: <https://www.youtube.com/watch?v=HdcvjEccO-4> Дата доступа: 25.05.2020

**ПРИЛОЖЕНИЕ А**Диаграмма классов Android-клиент



**ПРИЛОЖЕНИЕ Б**

Код структуры БД сервера создаемое API

import sqlalchemy as db

from sqlalchemy.ext.declarative import declarative\_base

from sqlalchemy import Boolean, Column, Integer, String, Numeric, create\_engine,ForeignKey, Date, or\_, and\_,func

Base = declarative\_base()

#таблица с пользователями

class Users(Base):

\_\_tablename\_\_ = "users"

id = Column(Integer, primary\_key=True, index=True)

email = Column(String, unique=True, index=True)

passhash = Column(String, nullable=False)

blocked=Column(Boolean, default="False")

#таблица с пользователями администраторами

class Admins(Base):

\_\_tablename\_\_ = "admins"

id = Column(Integer, primary\_key=True, index=True)

name = Column(String, nullable=False)

login = Column(String, nullable=False)

passhash = Column(String, nullable=False)

level = Column(Integer, nullable=False)

#таблица с конфигурацией

class SystemConfig(Base):

\_\_tablename\_\_ = "systemconfig"

id = Column(Integer, primary\_key=True, index=True)

enabled = Column(Boolean, default="True")

newregistration = Column(Boolean, default="True")

weblogin = Column(Boolean,default="True")

mobilesync = Column(Boolean,default="True")

#таблица с кошельками

class Wallets(Base):

\_\_tablename\_\_ = "wallets"

id = Column(Integer, primary\_key=True, index=True)

userid = Column(Integer,ForeignKey('users.id'),index=True)

walletid = Column(Integer,nullable=False,unique=True,index=True)

name = Column(String,nullable=False)

typemoney=Column(String, nullable=False)

money=Column(Numeric, nullable=False)

modifieddate=Column(Integer,nullable=False)

#таблица с кошельками

class Payments(Base):

\_\_tablename\_\_ = "payments"

id = Column(Integer, primary\_key=True, index=True)

userid = Column(Integer,ForeignKey('users.id'),index=True)

walletid = Column(Integer,ForeignKey('wallets.walletid'),index=True)

paymentid = Column(Integer,nullable=False,index=True)

type = Column(String,nullable=False)

summary=Column(String, default=" ")

money=Column(Numeric, nullable=False)

date=Column(String, nullable=False)

modifieddate=Column(Integer,nullable=False)

**ПРИЛОЖЕНИЕ В**

Код хеширования паролей в БД на сервере

#метод регистрации

@router.post("/register")

async def singup(item: Register,db: Session=Depends(get\_db)):

parts = item.email.split('@')

# Проверка валидности почты

if ((len(parts) != 2) or (len(parts[0]) == 0) or (len(parts[0]) > 64) or (len(parts[1]) == 0) or (len(parts[1]) > 255) or not ('.' in parts[1])):

return {'account':"email error"}

#проверка пароля

if (len(item.password)<6):

return {"account":"password error"}

result = db.query(Users).filter(Users.email == item.email).first()

if result:

return {'account':'email\_busy'}

#добавление в базу данных с генерацией хеша пароля

db.execute("INSERT INTO users(email,passhash) VALUES('"+item.email+"',crypt('"+item.password+"', gen\_salt('bf')))")

db.commit()

return {'account':'OK'}