**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

Индивидуальное домашнее задание

**по дисциплине «Нереляционные базы данных»**

Тема: **«Web-приложение электронный кошелек»**

|  |  |  |
| --- | --- | --- |
| Студент гр. 6382 |  | Костин С.В. |
| Студентка гр. 6382 |  | Черкасова Е.И. |
| Студент гр. 6382 |  | Середкин Д.Ю. |
| Преподаватель |  | Заславский М.М. |

Санкт-Петербург

2019

**ЗАДАНИЕ**

**на Индивидуальное домашнее задание**

|  |  |  |
| --- | --- | --- |
| Студенты Костин С.В., Черкасова Е.И., Середкин Д.Ю. | | |
| Группа 6382 | | |
| Тема работы: Web-приложение электронный кошелек | | |
| Исходные данные: необходимо реализовать многопользовательское web-приложение электронный кошелек, с помощью которого пользователи могу хранить, передавать денежные ресурсы в электронном виде и в разных валютах, криптовалюты включительно. | | |
| Содержание пояснительной записки:  «Содержание»  «Введение»  «Качественные требования к решению»  «Сценарий использования»  «Модель данных»  «Разработка приложения»  «Заключение»  «Список использованных источников» | | |
| Предполагаемый объем пояснительной записки:  Не менее 30 страниц. | | |
| Дата выдачи задания: 11.09.2019 | | |
| Дата сдачи реферата: 25.12.2019 | | |
| Дата защиты реферата: 25.12.2019 | | |
| Студент гр. 6382 |  | Костин С.В. |
| Студентка гр. 6382 |  | Черкасова Е.И. |
| Студент гр. 6382 |  | Середкин Д.Ю. |
| Преподаватель |  | Заславский М.М. |

**Аннотация**

В данном проекте было разработано многопользовательское web-приложение – электронный кошелек. С помощью данного приложения пользователь может хранить, передавать другим пользователям свои средства. Так же можно отслеживать курсы валют, чтобы выгодно продавать и покупать. Исходный код приложения можно найти на странице [GitHub](https://github.com/moevm/nosql2h19-blockchain). В качестве стека основных технологий разработки были MongoDB, Python3, React, Django.

**Summary**

In this project was designed multiuser web-application – web-wallet. With this app, users can store your money in any equivalent and send it to different users of our app. You can checkup course some currency for a good deal, buy or sell. You can find opensource code on [GitHub](https://github.com/moevm/nosql2h19-blockchain) page.

**содержание**

[введение 6](#_Toc27267342)

[1. Качественные требования к решению 7](#_Toc27267343)

[2. Сценарии использования 8](#_Toc27267344)

[3. Модель данных 12](#_Toc27267345)

[4. Разработанное приложение 17](#_Toc27267346)

[5. Выводы 18](#_Toc27267347)

[6. Приложения 18](#_Toc27267348)

[заключение 19](#_Toc27267349)

[список использованных источников 20](#_Toc27267350)

# **введение**

Цель работы – создать многопользовательское приложение – электронный кошелек.

Было решено создать web-приложение, где пользователь может хранить, передавать другим пользователям свои средства, отслеживать курсы валют. В качестве стека основных технологий разработки были MongoDB, Python3, React, Django.

# **Качественные требования к решению**

* 1. Текущие

Необходимо реализовать многопользовательское web-приложение электронный кошелек, с помощью которого пользователи могу хранить, передавать денежные ресурсы.

* 1. Перспективные

Возможно создание собственной платформы для торговли валютой, поддержка всех типов валют и их конвертация.

# **Сценарии использования**

* 1. Макеты UI
     1. Страница с информацией о статистики стоимости валюты рис. 1.

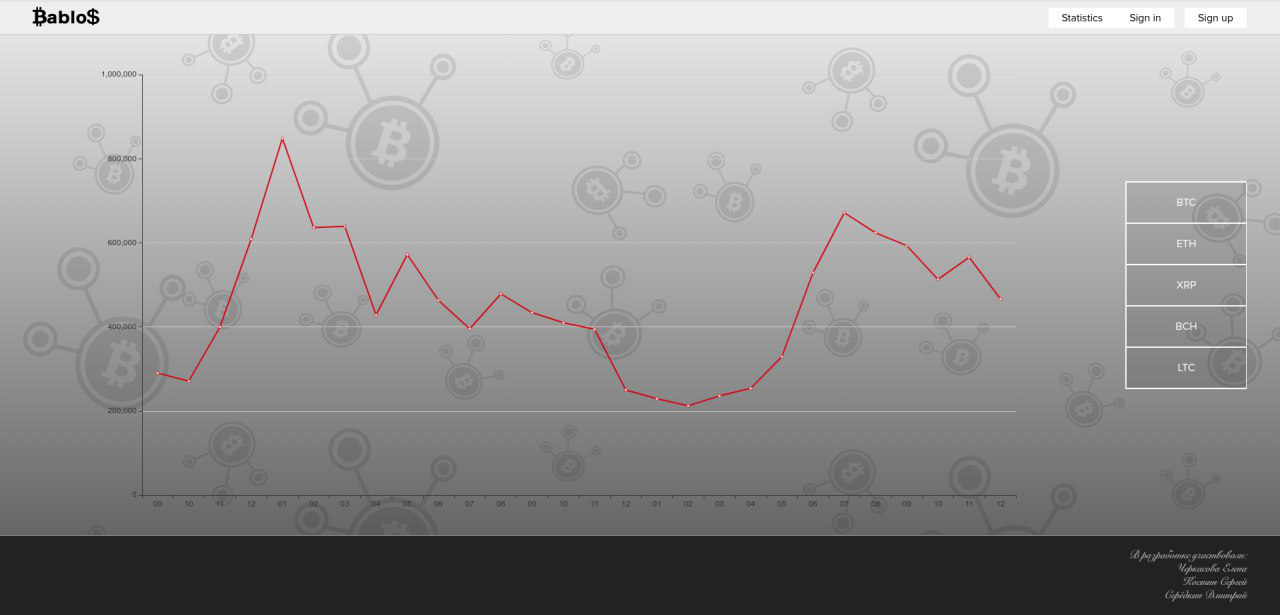


Рисунок 1 – Страница с информацией о статистики стоимости валюты

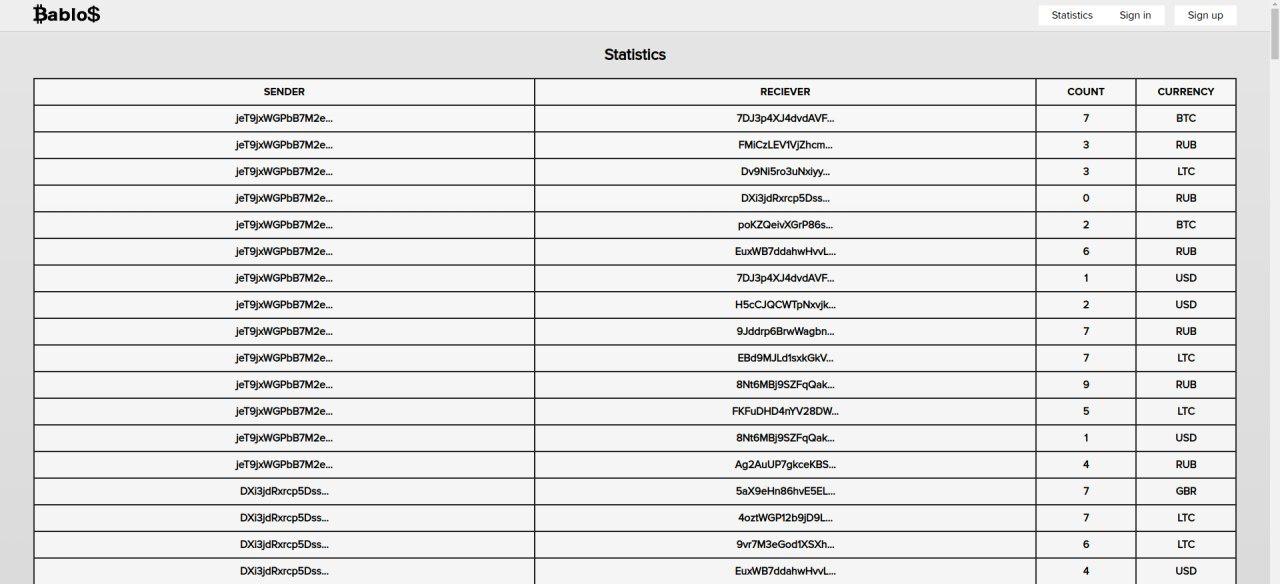
* + 1. Страница с информацией о транзакциях пользователей рис 2, 3.

Рисунок 2 – Страница с информацией о транзакциях пользователей

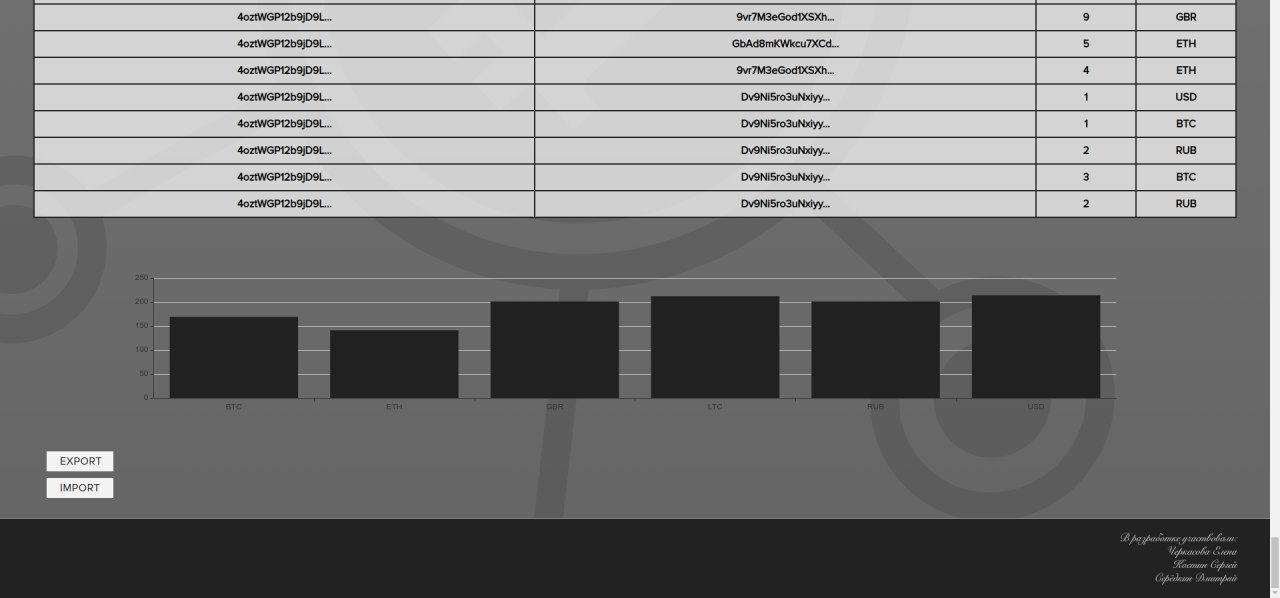


Рисунок 3 – Страница с информацией о транзакциях пользователей

* + 1. Страница входа в личный кабинет рис. 4**.**

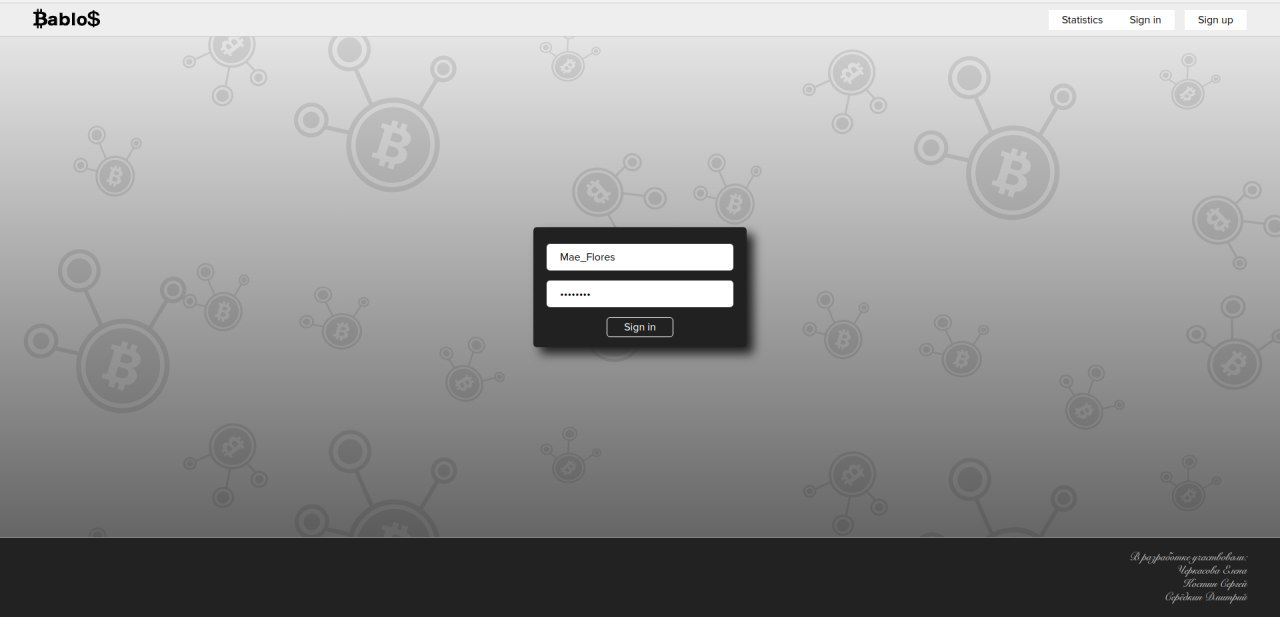


Рисунок 4 – Страница входа

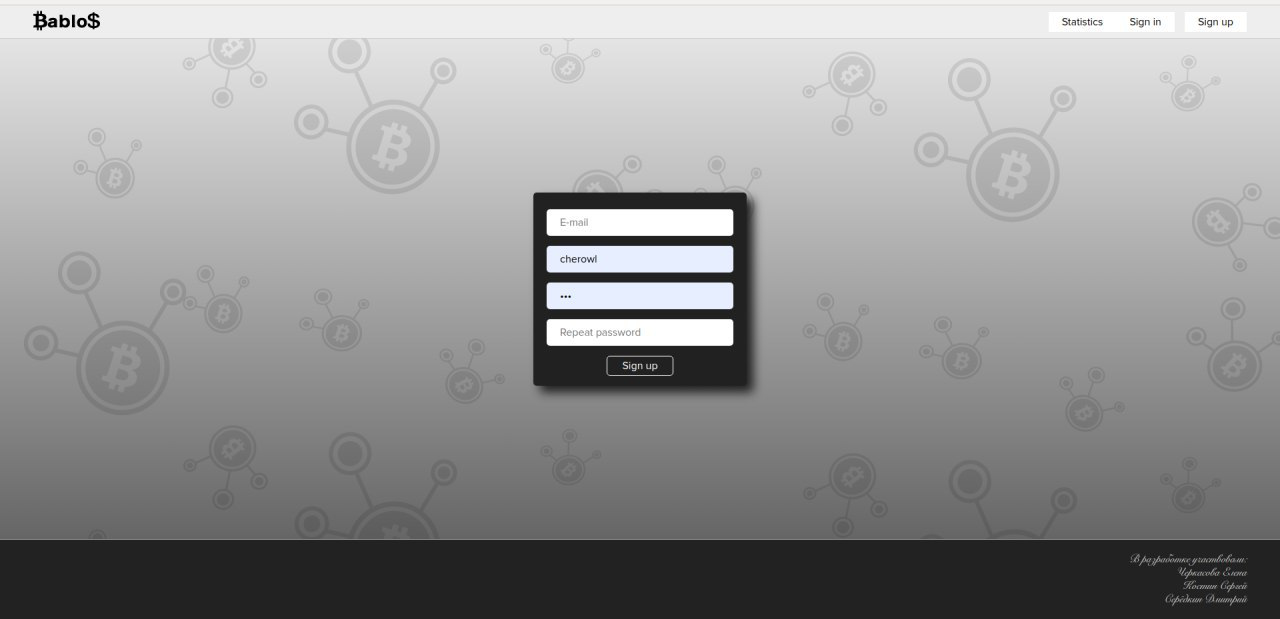
* + 1. Страница регистрации ново пользователя рис. 5**.** 

Рисунок 5 – Страница для создания аккаунта

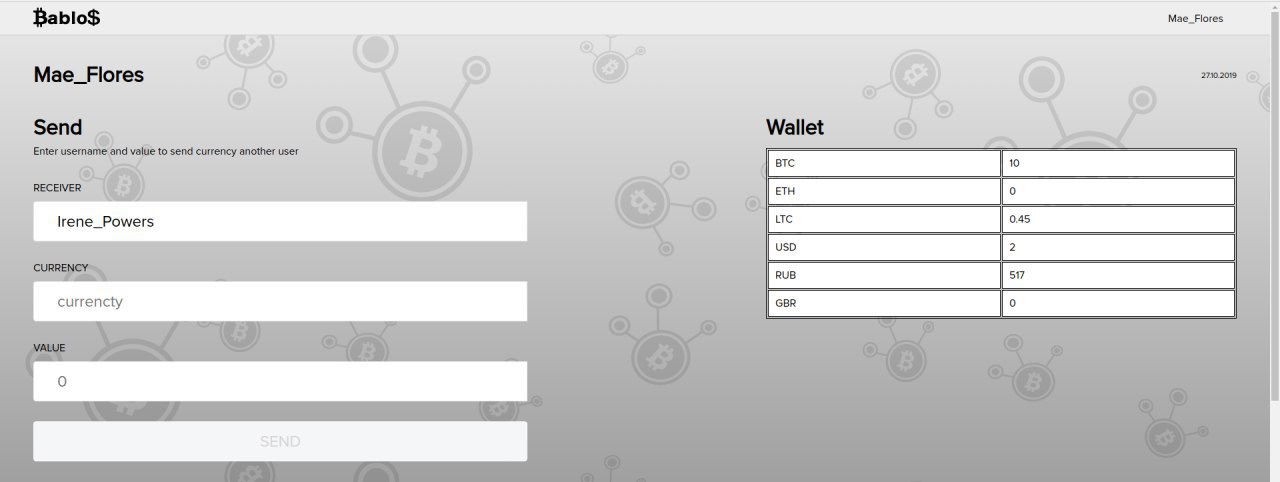
* + 1. Страница личного кабинета рис. 6. 

Рисунок 6 – Личный кабинет

* + 1. Страница пополнения кошелька рис 8.



Рисунок 8 – Пополнение счета

* 1. Сценарии использования
     1. Импорт данных и экспорт данных
     2. Представление данных

**Регистрация пользователя**: для регистрации пользователю необходимо нажать кнопку Sign up в header области сайта. Вы попадете на страницу регистрации. Вам нужно будет заполнить поля и нажать кнопку Sign up.

**Логирование пользователя**: для входа в свою учетную запись пользователю необходимо нажать кнопку Sign in в header области сайта. Вы попадете на страницу входа. Вам нужно будет логин и пароль и нажать кнопку Sign in.

**Выполнение транзакции**: для выполнения транзакции перейдите на страницу с учетной записью. Затем нажмите кнопку Remittance. В выпадающем окне введите логин пользователя, которому хотите отправить средства, выберите тип средств и количество. Нажмите кнопку Apply.

**Пополнение кошелька**: для пополнения кошелька нажмите кнопку Add purchase. В появившемся окне введите сумму, выберете метод платежа и нажмите кнопку Add.

* + 1. Анализ данных

На рис. 9 представлена диаграмма распределения количества транзакций по валютам.



Рисунок 9 – Диаграмма распределеня по валютам.

# **Модель данных**

* 1. Нереляционная модель данных
     1. Графическое представление

На рис. 3 вы можете видеть графическое представление схемы в MongoDB.

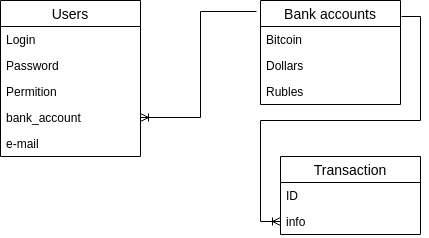


Рисунок 3 – Схема БД в MongoDB.

* + 1. Описание назначений коллекций, типов данных и сущностей

Users – коллекция пользователи. Предназначена для хранения данных о пользователей приложения, состоит из 4-х полей:

username – text

password – text

permission – text

e-mail – text

Bank accounts – коллекция счет. Предназначена для хранения данных о средствах пользователя:

username – ObjectId

bitcoin – double

dollars – double

rubles – double

Transactions – коллекция для хранения информации о всех транзакциях пользователей:

user\_id – ObjectId

key\_pair – string

metadata – string

* + 1. Оценка удельного объема информации, хранимой в модели

ObjectId = 12 bytes

String - 1 symb = 4 bytes \* 10 symbols = 40 b

Double = 8 bytes

users = {"login": string, password: string, permission: string, e-mail: string} = 40 b

bank\_accounts = {"user\_id": string, bitckoin: double, dollars: double, rubles: double } = 64 b

transaction = {user\_id: string, key\_pair: string, meta\_data: string} = 120 b

**"Clean" data volume** **V = S\*40 + S\*64 + S\*N\*120 = S\*(104 + N\*120)**

**Actual volume** **V = S\*173 + S\*44 + S\*N\*200 = S\*(213+ N\*200)**

**Избыточность модели**

Где N – Количество транзакций, S – Количество пользователей

Таким образом, на каждую запись в среднем приходится 109+80N потенциально дублирующийся байт данных, и реальный объем БД потенциально больше идеального размера в 1.6 раз больше.

Рисунок 10 – Сравнение роста ращмера данных

* + 1. Запросы к модели, с помощью которых реализуются сценарии использования

db.users.find\_one({"login": login})

db.users.update\_one(

{"\_id": user\_id},

{"$set": {"password": password, "e-mail" : e\_mail}}

)

db.bank\_accounts.delete\_one({"\_id": user["bank\_account"]})

db.bank\_accounts.insert\_one(init\_bank\_account).inserted\_id

db.users.create\_index([("login", pymongo.ASCENDING)], unique=True)

db.bank\_accounts.update\_one({"\_id": instance\_user\_from["bank\_account"]}, { "$inc": {resources: -value\_remitt}})

* 1. Аналог модели данных для SQL СУБД

На рис. 11 вы можете видеть графическое представление схемы в MySQL.

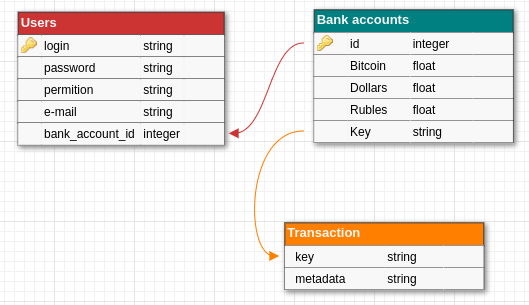


Рисунок 11 – Схема БД в MongoDB.

* 1. Сравнение моделей
     1. Удельный объем информации

String - 1 symb = 4 bytes \* 10 symbols = 40 b

Float = 4 bytes

int = 4 b

foreign key users = 2\*int

foreign key transactions = string = 40 b

bank\_accounts = {bitckoin: float, dollars: float, rubles: float } = 12 b

users = {"login": string, password: string, permission: string, bank\_account: int, e-mail: string} = 164 b

transaction = {user\_id: int, key\_pair: string, meta\_data: string} = 24 b

**"Clean" data volume** **V = S\*12 + S\*164 + S\*N\*24 = S\*(176 + N\*24)**

**Actual volume = S\*12 + S\*8 + S\*164 + S\*N\*(24 + 40) = S\*(184 + N\*64)**

**Избыточность модели**

Где N – Количество транзакций, S – Количество пользователей

Таким образом, на каждую запись в среднем приходится 12+40N потенциально дублирующийся байт данных, и реальный объем БД потенциально больше идеального размера в 2.6 раз больше.

Рисунок 12 – Сравнение роста размера данных

* + 1. Запросы по отдельным юзкейсам

insert into Users (login, password, permition, `e-mail`, bank\_account\_id) values ("akuma", "123ads", "admin", "akuma@mail.com", 1);

insert into `Bank accounts` (Bitcoin, Dollars, Rubles, `Key`) values (12.1, 0.0, 1.0, "sdsdfsfSDsfsfqe");

insert into Transaction (Bitcoin, Dollars, Rubles, `Key`) values ("sdsdfsfSDFsf3esf3", "27.11.19-23:19");

update `Bank accounts` set Bitcoin = 9, Dollars = 120 where id = 8;

select login, bank\_account\_id, Bitcoin from `Bank accounts` bk join Users u on bk.id = u.bank\_account\_id;

select count(`permition`) as CountUsers from Users where `permition`="user";

select login, `Key`, metadata from `Bank accounts` bk join Users u on bk.id = u.bank\_account\_id join Transaction t on t.`key` = bk.`Key`;

* + 1. Вывод - что лучше, SQL или NoSQL модель

Обе модели достаточно хороши и имеют свое преимущество. Но более гибкой в плане изменения структуры базы данных и добавления какого-либо функционала является NoSQL модель, что, конечно, может нести за собой увеличение необходимого дискового пространства.

1. **Разработанное приложение**
   1. Краткое описание

Многопользовательское web-приложение – электронный кошелек. С помощью данного приложения пользователь может хранить, передавать другим пользователям свои средства. Так же может отслеживать курсы валют, чтобы выгодно продавать и покупать.

* 1. Схема экранов приложения

На рис. 13 представлен граф макета пользовательского интерфейса.

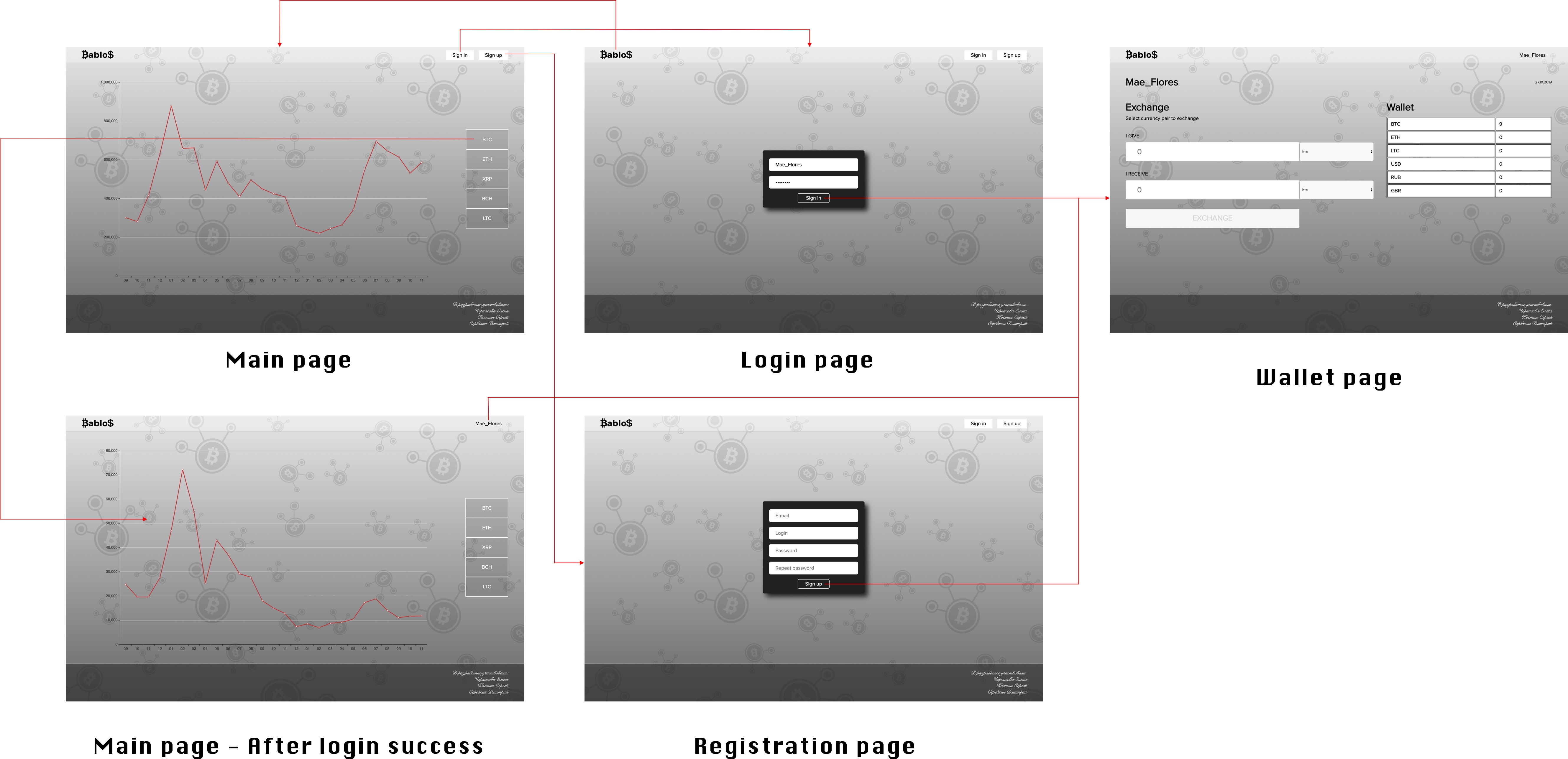


Рисунок 13 - Граф интерфейса

* 1. Использованные технологии

В качестве back-end составляющей использовался Django фрэймворк и python 3.6.

В качестве front-end составляющей использовался React.

Все данные хранятся в MongoDB.

* 1. Ссылка на приложение

GitHub: [<https://github.com/moevm/nosql2h19-blockchain>]

# **Выводы**

* 1. Достигнутые результаты

В ходе работы было разработано многопользовательское web-приложение электронный кошелек, с помощью которого пользователи могу хранить, передавать денежные ресурсы.

# **Приложения**

* 1. Документация по сборке и развертыванию приложения

Документация по сборе и доп. информация представлена в файле README.md на GitHub странице нашего приложения.

# **заключение**

В результате работы было создано многопользовательское web-приложение – электронный кошелек. Была разработана схема базы данных для хранения данных о пользователях приложения. Произведено сравнение ее с другой возможно используемой БД для определения эффективности выбранного решения. Основой приложения является клиентская часть на React и серверная часть на Django.

# **список использованных источников**

1. React Documentation, Электронный ресурс URL : <https://reactjs.org/docs/getting-started.html> .
2. Django Documentation, Электронный ресурс URL : <https://docs.djangoproject.com/en/3.0/>.
3. PyMongo Documentation, Электронный ресурс URL : <https://docs.mongodb.com/manual/>.
4. MongoDB Documentation, Электронный ресурс URL : <https://api.mongodb.com/python/current/api/index.html>.