МИНОБРНАУКИ РОССИИ САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ «ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА) Кафедра МО ЭВМ

ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ

по дисциплине «Введение в нереляционные базы данных»

Тема: Криптовалюты Mongo.

Студенты гр. 8303 _____ Стукалев А.И.

_____ Деркач Н.В.

_____ Логинов Е.А.

Преподаватель _____ Заславский М.М.

Санкт-Петербург

2021

ЗАДАНИЕ

Студенты

Стукалев А.И.

Деркач Н.В.

Логинов Е.А.

Группа 8303

Тема проекта: Криптовалюты Mongo.

Исходные данные:

Необходимо реализовать приложение для добавления различных криптовалют в систему, мониторинга их курса, конвертации с использованием СУБД MongoDB и веб-технологии Spring Boot.

Содержание пояснительной записки:

«Содержание»

«Введение»

«Качественные требования к решению»

«Сценарий использования»

«Модель данных»

«Разработка приложения»

«Вывод»

«Приложение»

Предполагаемый объем пояснительной записки:

Не менее 10 страниц.

Дата выдачи задания:

Дата сдачи реферата:

Дата защиты реферата:

Студенты гр. 8303		Стукалев А.И.
		Деркач Н.В.
		Логинов Е.А.
Преподаватель		Заславский М.М.

АННОТАЦИЯ

В рамках данного курса было предложена разработать веб-приложение для мониторинга курсов различных криптовалют за выбранный период. Основная цель данного проекта – получение практических навыков работы с нереляционными СУБД на примере MongoDB. Во внимание будут приниматься такие аспекты как производительность и удобство разработки. Найти исходный код и документацию можно по ссылке: https://github.com/moevm/nosql2h21-crypto-mongodb

ANNOTATION

In this course, a web application was proposed for monitoring the rates of various cryptocurrencies for a selected period. The main goal of this project is to gain practical skills in working with non-relational DBMS using MongoDB as an example. Aspects such as performance and usability will be taken into account. You can find the source code and documentation here: https://github.com/moevm/nosql2h21-crypto-mongodb

Оглавление

1. Введение 7
2. Качественные требования к решению.....7
3. Сценарии использования 7
4. Модель данных 13
5. Разработанное приложение 22
6. Вывод 22
7. Приложения.....27
8. Используемая литература 28

1. Введение

Цель работы – создать высокопроизводительное и удобное решение для мониторинга курсов криптовалют.

Было решено разработать веб-приложение, которое позволит мониторить добавленные ранее криптовалюты и также добавлять новые, отображать курс токена за указанный период.

2. Качественные требования к решению

Требуется разработать приложение с использованием MongoDB в качестве системы управления базами данных.

3. Сценарии использования

Макеты UI

1. Экран с основными возможностями приложения (Рис. 1).

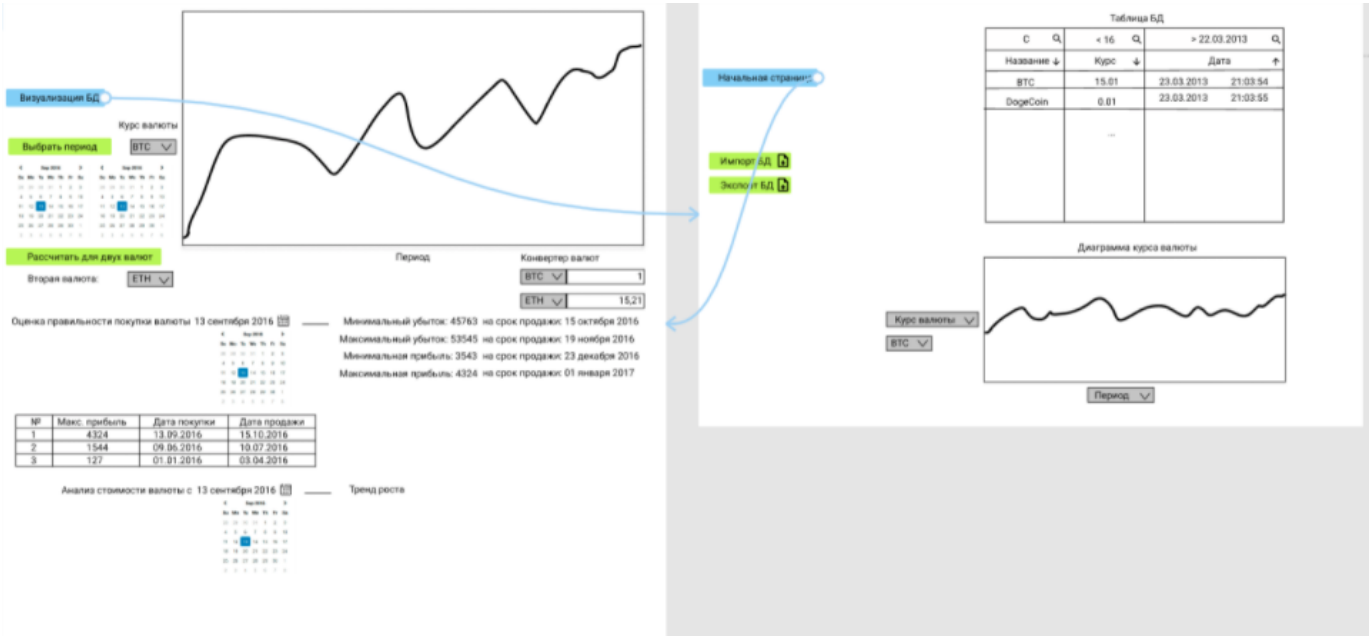

Рисунок 1. Макет приложения со всеми возможностями.

Описание сценариев использования

Use case. Основное действующее лицо - пользователь.

Просмотр истории курсов.

Основной сценарий:

1) Пользователь заходит на начальную страницу
2) Пользователь выбирает в выпадающем меню интересующую его валюту
3) Пользователь выбирает в меню интересующий его период
4) Пользователь нажимает кнопку "Выбрать период"
5) Пользователь видит график изменения курса валюты

Альтернативный сценарий:

1) Пользователь переходит на начальную страницу
2) Пользователь ничего не выбирает
3) Пользователь видит график установленной по умолчанию валюты за установленный по умолчанию период

Просмотр курса одной валюты к другой в виде графика.

Основной сценарий:

1) Пользователь заходит на начальную страницу
2) Пользователь выбирает в выпадающем меню интересующую его валюту
3) Пользователь выбирает в меню интересующий его период

4) Пользователь выбирает вторую валюты под кнопкой "Рассчитать для двух валют"

5) Пользователь нажимает кнопку "Рассчитать для двух валют"

6) Пользователь видит график курса одной валюты к другой

Альтернативный сценарий:

1)Пользователь переходит на начальную страницу

2) Пользователь ничего не выбирает

3) Пользователь видит график установленной по умолчанию валюты за установленный по умолчанию период

Импорт БД.

Основной сценарий:

1) Пользователь заходит на начальную страницу.

2) Пользователь по кнопке “Визуализация БД” переходит на страницу.

3) Пользователь по кнопке “Импорт БД” выгружает выбранную БД.

4) БД импортирована.

Альтернативный сценарий:

1) Пользователь заходит на начальную страницу.

2) Пользователь не нажимает кнопок.

3) БД не импортирована.

Экспорт БД.

Основной сценарий:

1) Пользователь заходит на начальную страницу.

2) Пользователь по кнопке “Визуализация БД” переходит на страницу.

3) Пользователь по кнопке “Экспорт БД” загружает БД.

4) БД экспортирована.

Альтернативный сценарий:

1) Пользователь заходит на начальную страницу.

2) Пользователь не нажимает кнопок.

3) БД не экспортирована.

Просмотр БД в виде таблицы.

Основной сценарий:

1) Пользователь заходит на начальную страницу.

2) Пользователь по кнопке “Визуализация БД” переходит на страницу.

3) Пользователь нажатием на название столбца таблицы сортирует его либо по возрастанию, либо по убыванию.

4) Пользователь вводом значения (для столбца с названием - набор букв, для даты и курса - выражение) фильтрует строки таблицы.

5) Пользователь видит представление БД в виде таблицы.

Альтернативный сценарий:

1) Пользователь заходит на начальную страницу.

2) Пользователь не нажимает кнопок.

3) Пользователь не может просмотреть БД в виде таблицы.

Просмотр БД в виде кастомной диаграммы.

Основной сценарий:

1) Пользователь заходит на начальную страницу.

2) Пользователь по кнопке “Визуализация БД” переходит на страницу.

3) Пользователь выбирает оси диаграммы.

4) Пользователь види представление БД в виде кастомной диаграммы

Альтернативный сценарий:

1) Пользователь заходит на начальную страницу.
2) Пользователь не нажимает кнопок.
3) Пользователь не может просмотреть БД в виде кастомной диаграммы.

Оценка правильности покупки/продажи валюты.

Основной сценарий:

1) Пользователь переходит на начальную страницу
2) Пользователь выбирает интересующую его валюту
3) Пользователь выбирает в выпадающем календаре интересующую его дату
4) Пользователь видит оценку правильности покупки выбранной валюты в выбранный день

Альтернативный сценарий:

1) Пользователь переходит на начальную страницу
2) Пользователь не выбирает интересующую его валюту и дату
3) Пользователь видит оценку правильности покупки заданной по умолчанию валюты в заданный по умолчанию день

Обмен валют.

Основной сценарий:

1) Пользователь переходит на начальную страницу
2) Пользователь выбирает в выпадающем списке валюту, которую хочет конвертировать
3) Пользователь выбирает в выпадающем списке валюту, в которую хочет конвертировать валюту из пункта 2
4) Пользователь видит результат конвертации

Альтернативный сценарий:

1) Пользователь переходит на начальную страницу

2) Пользователь не выбирает в выпадающем списке валюты для конвертации

3) Пользователь видит результат конвертации заданных по умолчанию валют

Анализ истории.

Основной сценарий:

1) Пользователь переходит на начальную страницу

2) Пользователь выбирает интересующую его валюту

3) Пользователь выбирает в выпадающем календаре интересующую его дату

4)Пользователь видит, какой тренд имеет валюта в данный момент - тренд роста или тренд спада.

Альтернативный сценарий:

1) Пользователь переходит на начальную страницу

2) Пользователь не выбирает интересующую его валюту и дату

3) Пользователь видит тренд заданной по умолчанию валюты с заданной по умолчанию даты

4. Модель данных

Нереляционные модели данных

MONGODB

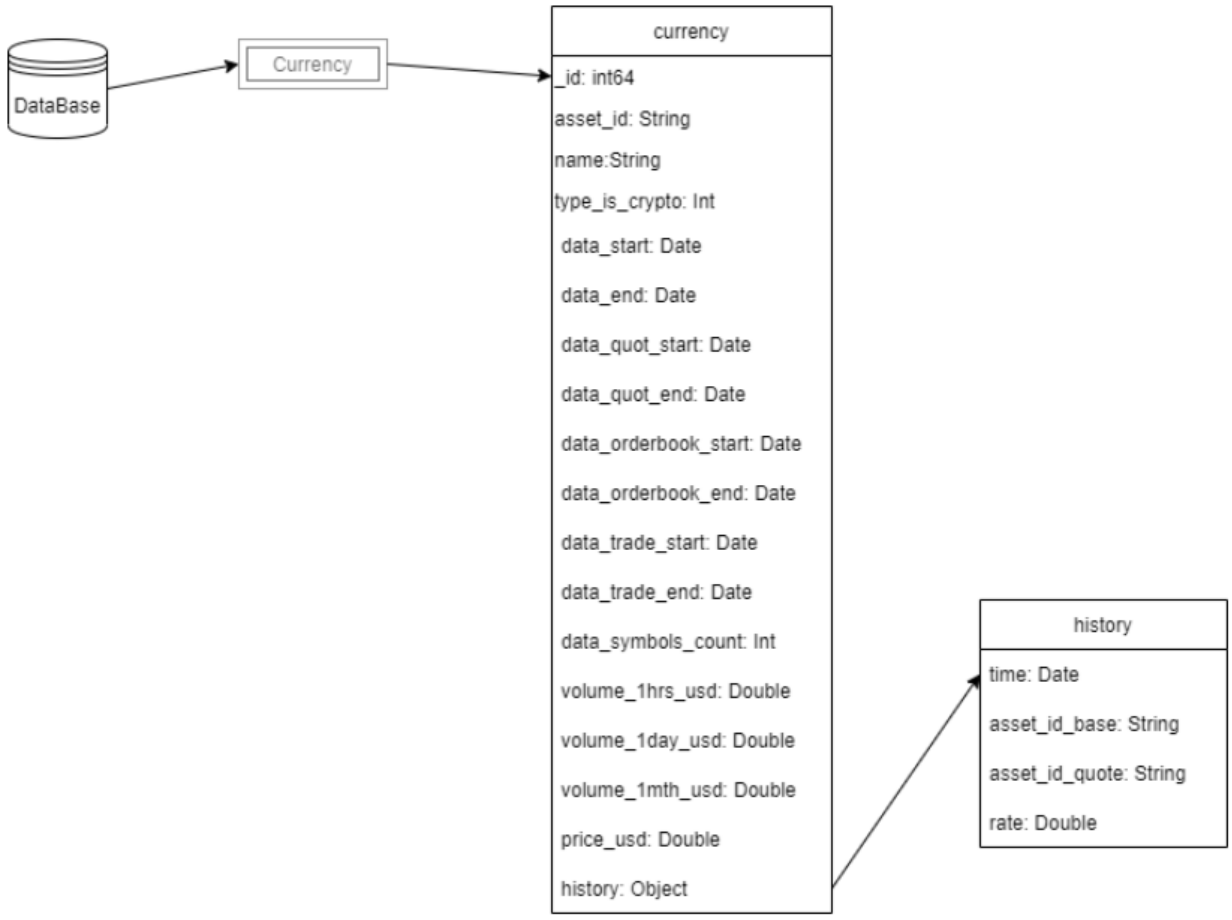

Рисунок 2. Визуальное представление модели данных MongoDB.

Модель документа:

Оценка удельного объема информации, хранимой в моделях

Оценка объема

- Коллекция "Currency"
    - _id - тип Int64 **V = 8b**
    - Asset_id – тип String **V = 5b**
    - name – тип String **V = 5b**
    - type_is_crypto – тип Int **V = 4b**
    - data_start - тип Date **V = 8b**
    - data_end - тип Date **V = 8b**
    - data_quot_start - тип Date **V = 8b**
    - data_quot_end- тип Date **V = 8b**
    - data_orderbook_start - тип Date **V = 8b**
    - data_orderbook_end - тип Date **V = 8b**
    - data_trade_start - тип Date **V = 8b**
    - data_trade_end - тип Date **V = 8b**
    - data_symbols_count - тип Int **V = 4b**
    - volume_1hrs_usd - тип Double **V = 8b**
    - volume_1day_usd - тип Double **V = 8b**
    - volume_1mth_usd - тип Double **V = 8b**
    - price_usd: Double - тип Double **V = 8b**
    - time - тип Date **V = 8b**
    - asset_id_base – тип String **V = 5b**
    - asset_id_quote – тип String **V = 5b**
    - rate: Double - тип Double **V = 8b**

- history - тип Object. V = (8 + 5 + 5 + 8) * X, где X ~ 100, среднее количество курсов. V = 2400 b

Средний объем данных для хранения одной криптовалюты V = 2540b.

Формула зависимости объема от количества валют:
N – количество валют

2540 * N

Примеры запросов

- Запрос на добавление криптовалюты:

```
db.currency.insertOne(new Document(Map.of(
"_id", _id,
"asset_id", asset_id,
"name", name,
"type_is_crypto", type_is_crypto,
"data_start", data_start,
"data_end", data_end,
"data_quot_start", data_quot_start,
"data_quot_end", data_quot_end,
"data_orderbook_start", data_orderbook_start,
"data_orderbook_end", data_orderbook_end,
"data_trade_start", data_trade_start,
"data_trade_end", data_trade_end,
"data_symbols_count", data_symbols_count,
"volume_1hrs_usd", volume_1hrs_usd,
"volume_1day_usd", volume_1day_usd,
"volume_1mth_usd", volume_1mth_usd,
"price_usd", price_usd)));
```

- Запрос об обновлении данных о криптовалюте:

```
BasicDBObject newData = new BasicDBObject();

newData.put("_id", new_id,
```

```
"asset_id", newasset_id,
"name", newname,
"type_is_crypto", newtype_is_crypto,
"data_start", newdata_start,
"data_end", newdata_end,
"data_quot_start", newdata_quot_start,
"data_quot_end", newdata_quot_end,
"data_orderbook_start", newdata_orderbook_start,
"data_orderbook_end", newdata_orderbook_end,
"data_trade_start", newdata_trade_start,
"data_trade_end", newdata_trade_end,
"data_symbols_count", newdata_symbols_count,
"volume_1hrs_usd", newvolume_1hrs_usd,
"volume_1day_usd", newvolume_1day_usd,
"volume_1mth_usd", newvolume_1mth_usd,
"price_usd", newprice_usd);

BasicDBObject searchQuery = new BasicDBObject().append("_id", _id,
"asset_id", asset_id,
"name", name,
"type_is_crypto", type_is_crypto,
"data_start", data_start,
"data_end", data_end,
"data_quot_start", data_quot_start,
"data_quot_end", data_quot_end,
"data_orderbook_start", data_orderbook_start,
"data_orderbook_end", data_orderbook_end,
"data_trade_start", data_trade_start,
"data_trade_end", data_trade_end,
"data_symbols_count", data_symbols_count,
"volume_1hrs_usd", volume_1hrs_usd,
"volume_1day_usd", volume_1day_usd,
"volume_1mth_usd", volume_1mth_usd,
"price_usd", price_usd);

currency.update(searchQuery, newData);
```

- Запрос о добавлении информации о курсах валюты:

```
BasicDBObject newDocument = new BasicDBObject();
```

```
newDocument.append("$set", new BasicDBObject().append("history", new
Document(Map.of(
"time", time,
"asset_id_base", asset_id_base,
"asset_id_qout", asset_id_quot,
"rate", rate))));

BasicDBObject searchQuery = new BasicDBObject().append("_id", old_id);

currency.update(searchQuery, newDocument);
```

- Запрос для поиска валюты по её id:

```
db.currency.find_one({'_id': node})
```

- Запрос для поиска валюты по её assert_id:

```
db.currency.find_one({'assert_id': node})
```

- Запрос для получения курса валюты по заданному времени:

```
currency.find(new Document("history", new Document("date", new
Document("$regex", date))))
```

Аналог модели данных для SQL СУБД

Вышеописанную модель данных для библиотечных карточек также можно представить виде реляционной модели с помощью таблиц:

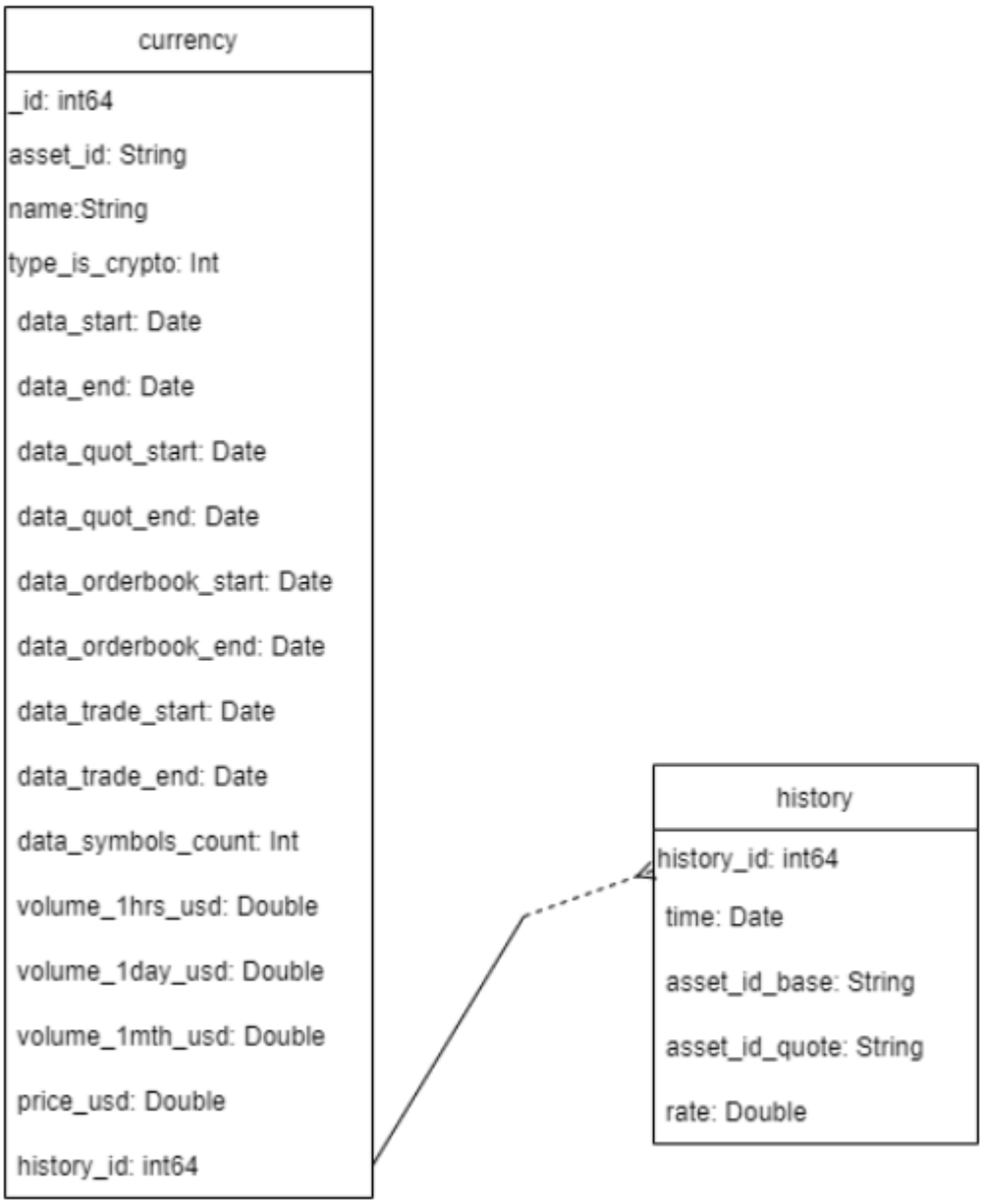

Рисунок 3. Визуальное представление модели данных реляционной базы данных.

Оценка удельного объема информации, хранимой в модели

- Таблица "Currency"
  - _id - тип Int64 **V = 8b**
  - Asset_id – тип String **V = 5b**

    - name – тип String **V = 5b**
    - type_is_crypto – тип Int **V = 4b**
    - data_start - тип Date **V = 8b**
    - data_end - тип Date **V = 8b**
    - data_quot_start - тип Date **V = 8b**
    - data_quot_end- тип Date **V = 8b**
    - data_orderbook_start - тип Date **V = 8b**
    - data_orderbook_end - тип Date **V = 8b**
    - data_trade_start - тип Date **V = 8b**
    - data_trade_end - тип Date **V = 8b**
    - data_symbols_count - тип Int **V = 4b**
    - volume_1hrs_usd - тип Double **V = **8b**
    - volume_1day_usd - тип Double **V = 8b**
    - volume_1mth_usd - тип Double V = 8b
    - price_usd: Double - тип Double **V = 8b**
    - history - тип Object. V = (8 + 8 + 5 + 5 + 8) * X, где X ~ 100, среднее количество курсов. V = 3200b
- Таблица "History"
    - History_id – уникальный идентификатор узла - тип **Int V = 4b**
    - time - тип Date **V = 8b**
    - asset_id_base – тип String **V = 5b**
    - asset_id_quote – тип String **V = 5b**
    - rate: Double - тип Double **V = 8b**

Средний объем данных для хранения одной криптовалюты V = 3352b.

Примеры запросов

- Запрос на добавление криптовалюты:

INSERT currnecy(_id, asset_id, name, type_is_crypto, data_start, data_start, data_quot_start, data_quot_end
, data_orderbook_start, data_trade_start, data_trade_end, data_symbols_count, volume_1hrs_usd,
volume_1day_usd, volume_1mth_usd, price_usd)
VALUES (_id1, asset_id1, name1, type_is_crypto1, data_start1, data_start1, data_quot_start1, data_quot_end1

, data_orderbook_start1, data_trade_start1, data_trade_end1,
data_symbols_count1, volume_1hrs_usd1,
volume_1day_usd1, volume_1mth_usd1, price_usd1)

- Запрос об обновлении данных о криптовалюте:

UPDATE currency SET
_id = _id1, asset_id = asset_id1, name = name1, type_is_crypto = type_is_crypto1,
data_start = data_start1,
data_quot_start = data_orderbook_start1, data_trade_end = data_quot_end1,
data_orderbook_start = data_orderbook_start1,
data_trade_start = data_trade_start1, data_trade_end = data_trade_end1,
data_symbols_count = data_symbols_count1,
volume_1hrs_usd = volume_1hrs_usd1, volume_1day_usd = volume_1day_usd1,
volume_1mth_usd = volume_1mth_usd1,
price_usd = price_usd1

- Запрос о добавлении информации о курсах валюты:

INSERT INTO history (rate, history_id)
VALUES (rate1, history_id1), (rate2, history_id2)
INSERT INTO rate (history_id, time, asset_id_base, asset_id_base, asset_id_qout,
rate)
VALUES (history_idtime1, asset_id_base1, asset_id_base1,
asset_id_qout1, rate1),
(history_idtime2, asset_id_base2, asset_id_base2, asset_id_qout2, rate2),

- Запрос для поиска валюты по её id:

SELECT * FROM currency WHERE _id = _id1;

- Запрос для поиска валюты по её assert_id:

SELECT * FROM currency WHERE assert_id = assert_id1;

- Запрос для получения курса валюты по заданному времени:

SELECT *

```
FROM currency
INNER JOIN history
   ON currency.history = history.rate
INNER JOIN rate
   ON history.history_id = rate.history_id
WHERE currency._id = %id%
   AND   rate.date = %date%
```

Сравнение моделей

SQL модель данных для требует незначительно больше места. Это связано с тем, что для SQl в каждой таблице необходимо хранить одно уникальное поле, которое выступает в роли ключа для связи с другими таблицами. SQL модель требует значительно больше запросов для взаимодействия с БД.

Для получения курса валюты по заданному времени в БД потребуется:

- noSQL – 1 запрос.
- SQL – 3 запроса.

Для добавления 100 новых валюты в БД потребуется:

- noSQL – 100 запрос.
- SQL – 200 запросов.

Избыточность моделей по сравнению с чистыми данными:

noSQL - 47 %.

SQL - 24 %. Объяснить такую разницу можно тем, что количество документов в не реляционной БД крайне мало.

Вывод: SQL расходует больше памяти, одни и те же операции требуют больше запросов, чем в MongoDB.

5. Разработанное приложение

Краткое описание

Back-end представляет из себя приложение на Java Spring Boot. Возможность переключения между СУБД реализована следующим образом:

Front-end – это web-приложение, которое использует API back-end приложения и отображает данные удобным образом для пользователя.

Схема экранов приложения

Экраны приложения и переходы между ними отображены на рисунке 9.

Использованные технологии

БД: MongoDB.

Back-end: Java 17, Spring Boot, thymeleaf.

Front-end: HTML, CSS, JavaScript, thymeleaf, highcharts.

Ссылки на Приложение

1. Ссылка на github: https://github.com/moevm/nosql2h21-crypto-mongodb

6. Вывод

В ходе выполнения работы было разработано приложения для мониторинга статистики по различным криптовалютам. Пользователь

может добавлять новые криптовалюты для отслеживания, анализировать актуальность покупки той или иной валюты, анализировать тренд валюты, выполнять конвертацию между валютами. Также есть возможность построения графиков курса валюты за указанный период и зависимости одной криптовалюты от другой.

Результаты

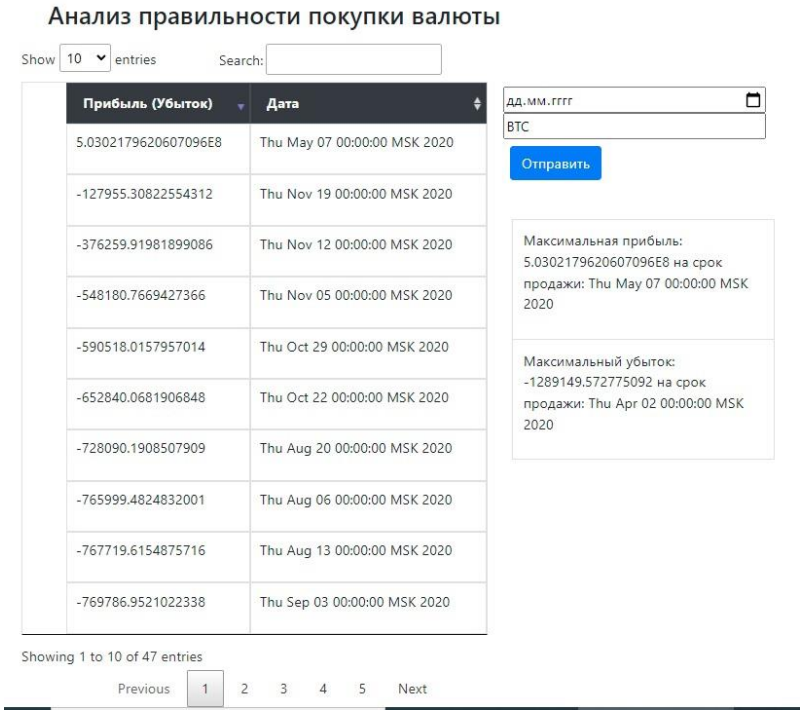

Рисунок 4. Анализ правильности покупки валюты.

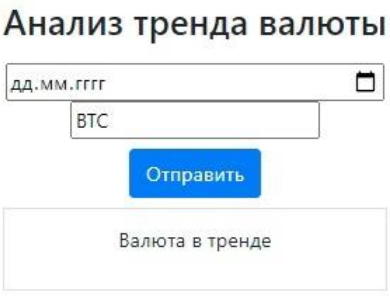

Рисунок 5. Анализ тренда валюты.

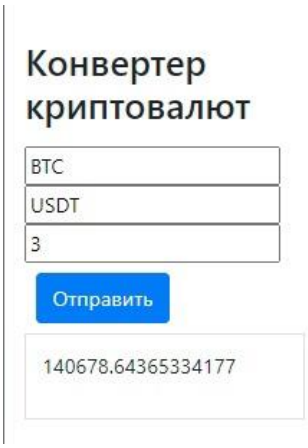

Рисунок 6. Конвертер валюты.

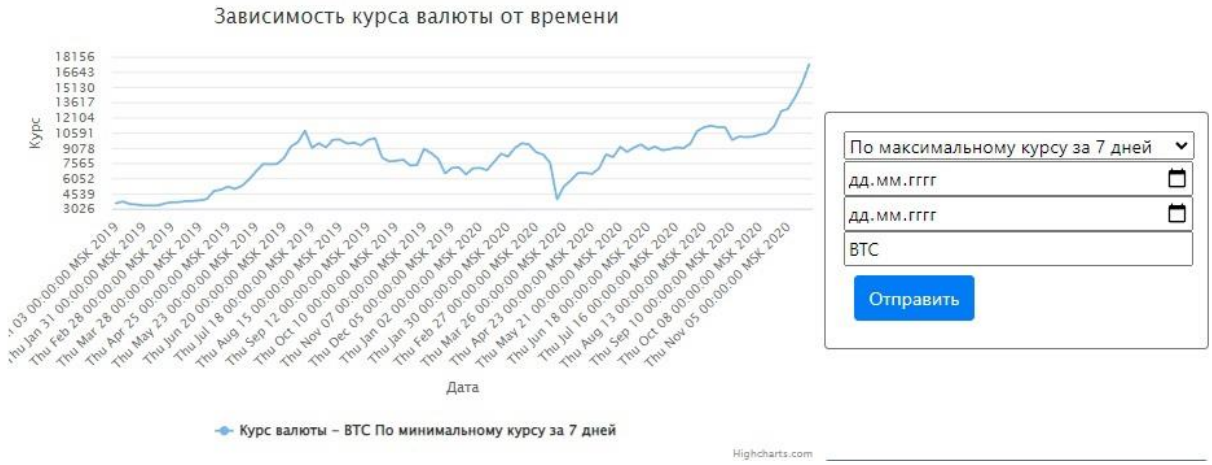

Рисунок 7. график зависимости курса валюты от времени(костюмная диаграмма).

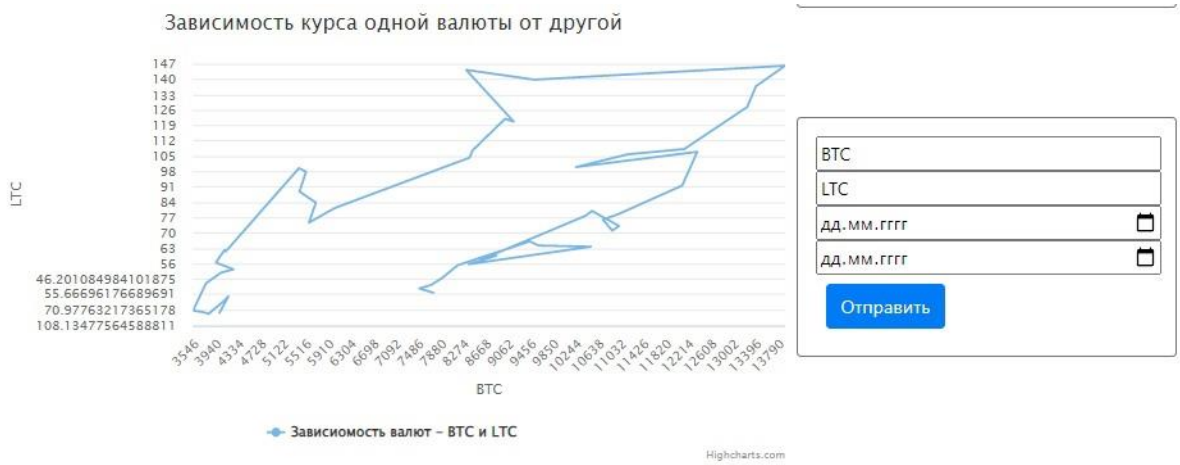

Рисунок 8. график зависимости одной валюты от другой.

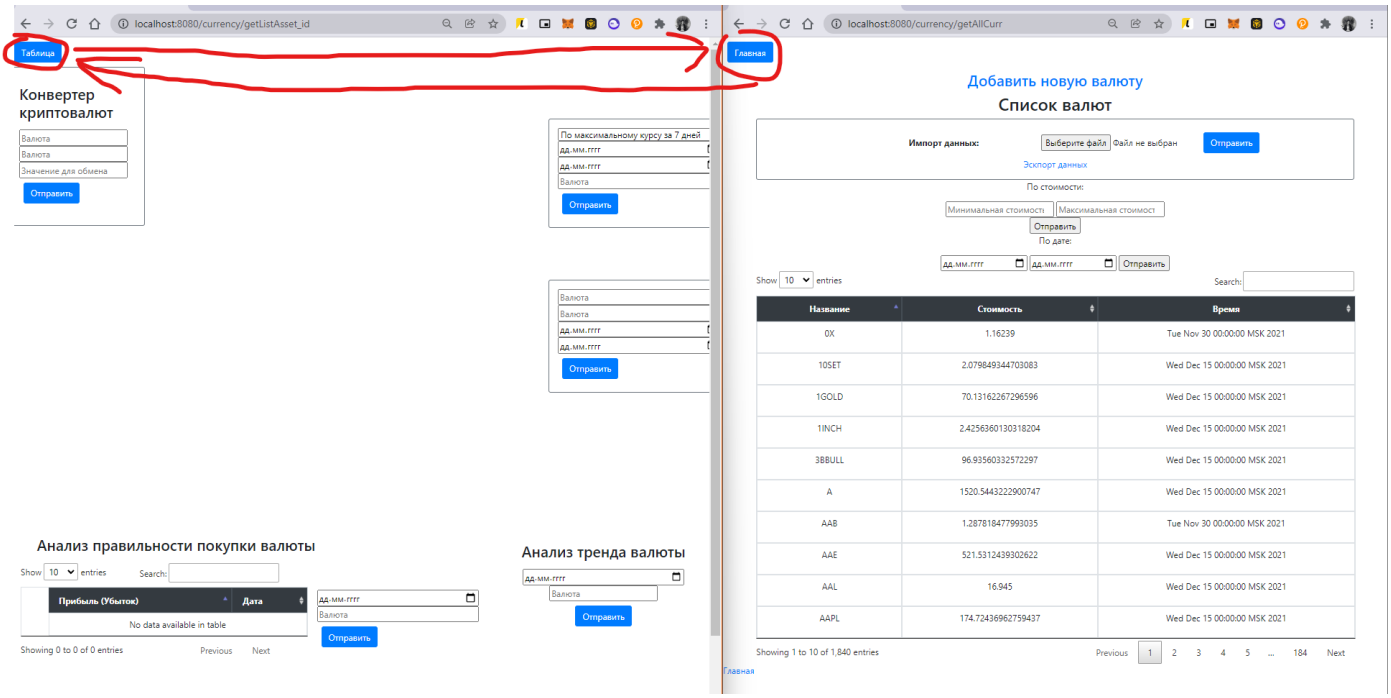

Рисунок 9. Схема экранов приложения и переходов между ними.

Страница /currency/getAllCurr позволяет добавлять новые криптовалюты в БД. Также на ней присутствует таблица с возможностью сортировки по названию, стоимости и времени добавления для имеющихся криптовалют.

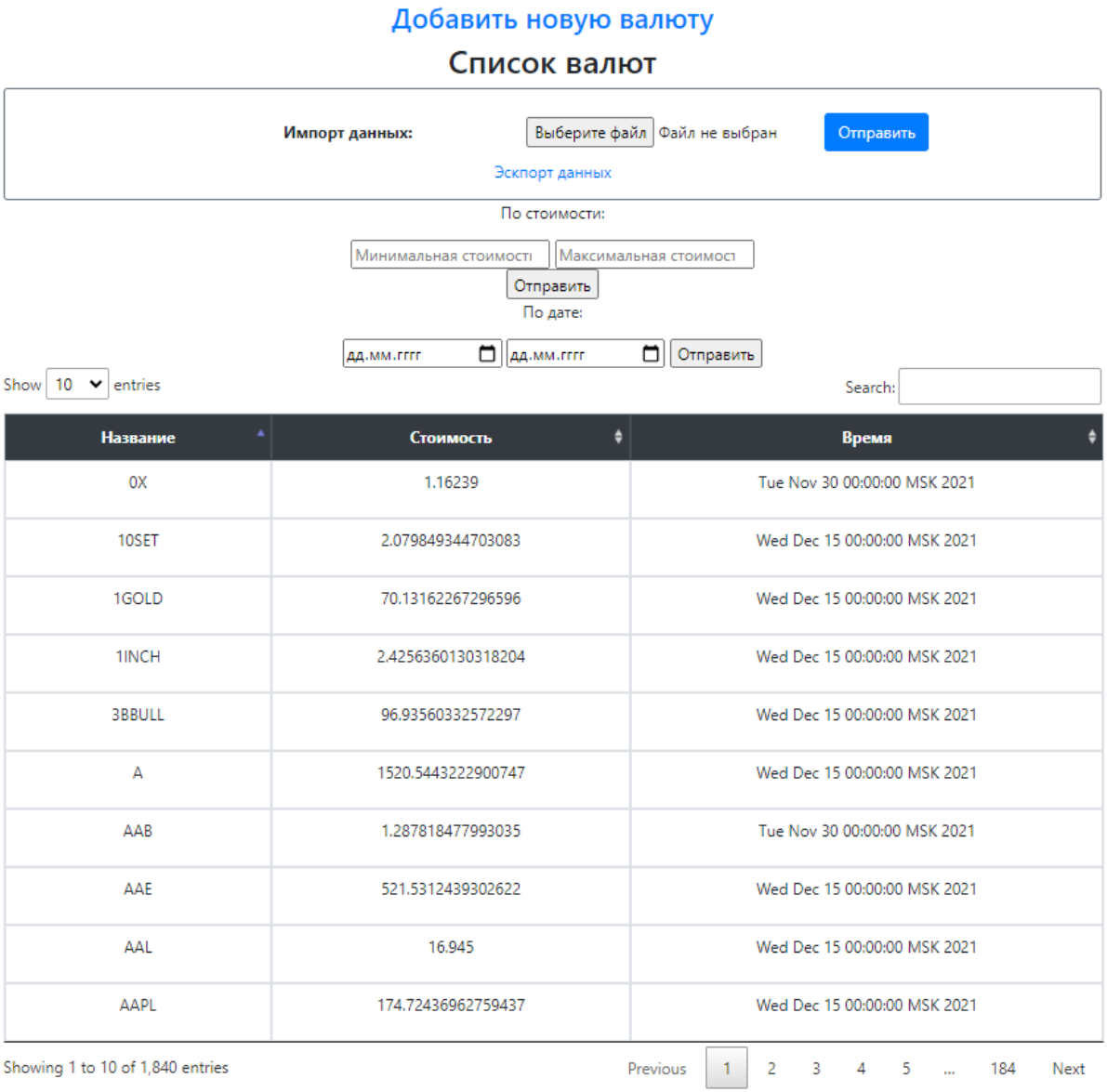

Рисунок 9. Страница, отображающая уже добавленные криптовалюты в БД.

Недостатки и пути для улучшения полученного решения

1. Бесплатная версия CoinApi позволяет делать 100 запросов в день, также CoinApi имеет ограничение на запрос к истории курса валют(за один запрос возвращает только 100 записей об истории). Соответственно, запросов не хватает, чтобы проводить анализ на большом временном интервале. Для Анализа в проекте был выбран временной промежуток с 01.01.2019 по 01.11.2020, и история по 4 валютам.

2. Выбранная технология thymeleaf создаёт html странички на стороне сервера и затем отправляет их клиенту. Соответственно, возникает трудность контроля действий пользователя, так как при большинстве действий требуется отправить страничку заново.

Будущее развитие решения

Поиск альтернативных API для получения более детальной статистики по криптовалютам.

7. Приложения

Документация по сборке и развертыванию приложения

1) Скачать проект из репозитория (указан в ссылках на приложение).

С использованием докер-контейнера:

1) Выполнить команду “docker-compose up —build -d” в корневой директории проекта.

2) Запустить веб-приложение в браузере по адресу http://localhost:8080/

Вариант без докер-контейнера:

2) Установить MongoDB.

3) Запустить CrypApplication.java

4) Запустить веб-приложение в браузере по адресу http://localhost:8080/

8. Используемая литература

1. Документация MongoDB: https://docs.mongodb.com/manual/
2. Документация Docker: https://docs.docker.com
3. Документация Highcharts https://www.highcharts.com/docs/index
4. Документация Thymeleaf https://www.thymeleaf.org/documentation.html