Уважаемые Кандидаты!

При решении задач просьба проявить все свои знания в области ИТ.

Мы ждем от Вас решения, которые покажут ваши знания, опыт и понимание того, как решаются проблемы и создается хороший программный код. Можно прислать только часть решения, обязательно указав, что именно у вас получилось, а что нет.

* Для решения задачи Вам необходимо использовать Java.

При выполнении важно обратить внимание на следующее:

* Ваша первоочередная задача сделать рабочий вариант задачи. От кода который не запускается пользы мало.
* Алгоритм может быть сколько угодно сложным, но не забывайте про п.1 — лучше более простой, но рабочий вариант.
* Старайтесь написать код максимально "красиво", это не олимпиадная задача и читаемость тоже оценивается.

Удачи Вам!

**Биржа**

Требуется разработать **RESTfull API service** — биржу для проведения торгов криптовалютами.

**Обязательные требования:**

\* Spring Boot;

\* возвращаемые данные в формате Json;

Приложение должно реализовывать **упрощенный** функционал **криптобиржи**.

В системе предполагается две роли: **пользователь** и **администратор**. В упрощенном варианте достаточно при получении запроса проверять полученный secret\_key — если данный метод доступен только администратору и secret\_key тоже принадлежит администратору, то выполнять операцию, а иначе возвращать http статус — 403 Forbidden (вариант «со звёздочкой» предполагает подключение Spring Security).

**Пользователь** должен иметь возможность зарегистрироваться, пополнить кошелёк, вывести деньги, купить/продать крипту.

**Администратор** должен иметь возможность изменить курс крипты, посмотреть статистику по всем кошелькам пользователей, статистику по торгам за определенный промежуток времени.

Все данные по кошелькам и истории сделок должны храниться в постоянной памяти компьютера (не обнуляться при перезапуске приложения) — в обычных текстовых файлах (и потом подгружаться в приложение в HashMap, например) или в базе данных (вариант «со звёздочкой»). Далее по тексту, будет использоваться слово «*хранилище*» -- подразумевается, что это может быть как база данных, так и текстовый файл.

Приложение должно иметь REST API (принимать REST запросы и возвращать ответы).

**Примерный** набор **API** (названия методов, т. е. URI на которые будут отправляться запросы — нужно придумать самому). Функционал может быть расширен/дополнен кандидатом на своё усмотрение, но обязательно все доработки должны быть описаны в сопутствующей документации.

**API ДЛЯ ПОЛЬЗОВАТЕЛЯ.**

**1. Регистрация нового пользователя.**

**POST запрос.**

Должна осуществляться проверка, что username и email уникальны (их еще нет в хранилище) — а иначе возвращать отказ в регистрации.

{

"username": "vasya\_vezunchik",

"email": "vasyu\_kolbasit@mail.ru"

}

**Ответ.**

В ответ должна генерироваться уникальная хеш-строка (случайный набор символов и букв) фиксированной длинны, которая будет служить секретным ключом для доступа пользователя к своему аккаунту (кошельку). Строка будет закрепляться за конкретным пользователем и по ней будет происходить поиск нужного кошелька в системе.

{

"secret\_key": "AAFeyWzOnlD-9G4i662PdKn2B-b4BwrCNA",

}

**2. Просмотр баланса своего кошелька.**

**GET запрос.**

Пользователь может запросить баланс своего кошелька. Для этого в запросе потребуется передать секретный ключ, полученный при регистрации (для идентификации пользователя).

{

"secret\_key": "AAFeyWzOnlD-9G4i662PdKn2B-b4BwrCNA"

}

**Ответ.**

В ответ придёт выгрузка данных по его кошелькам. Например, у него на счёту 0.0031589 биткоинов, 254.87 TON и 3500 рублей.

{

"BTC\_wallet": "0.0031589",

"TON\_wallet": "254.87",

"RUB\_wallet": "3500"

}

**3. Пополнение кошелька.**

**POST запрос.**

Пользователь может пополнить кошёлок. Предполагаем, что для этого достаточно просто отправить REST-запрос и ему на счёт зачислятся деньги из пустоты :)

{

"secret\_key": "AAFeyWzOnlD-9G4i662PdKn2B-b4BwrCNA",

"RUB\_wallet": "1000"

}

**Ответ.**

В случае успешного запроса — в ответ придёт обновленный баланс кошелька.

{

"RUB\_wallet": "4500"

}

**4. Вывод денег с биржи.**

**POST запрос.**

Пользователь должен иметь возможность вывести валюту с биржи. Для этого он должен отправить номер кредитной карты (для вывода рублей) или адрес кошелька (для вывода крипты) — это, как правило, строка из случайного набора символов. Также, потребуется указать тип валюты в запросе.

{

"secret\_key": "AAFeyWzOnlD-9G4i662PdKn2B-b4BwrCNA",

"currency": "RUB",

"count": "1500",

"credit\_card": "1234 5678 9012 3456"

}

ИЛИ

{

"secret\_key": "AAFeyWzOnlD-9G4i662PdKn2B-b4BwrCNA",

"currency": "TON",

"count": "15",

"wallet": "AsS5A2SASd2as3q5sd2asd53a1s5"

}

**Ответ.**

Должна выполняться проверка, что на счёте достаточно денег. В случае успешного запроса — в ответ придёт обновленный баланс кошелька (из которого происходил вывод денег).

{

"RUB\_wallet": "3000"

}

**5. Просмотр актуальных курсов валют (данный запрос будет доступен и пользователю и администратору).**

**GET запрос.**

Пользователь может запросить актуальные курсы валют. Для этого он в запросе должен указать тип валюты, относительно которой он хочет узнать курс.

{

"secret\_key": "AAFeyWzOnlD-9G4i662PdKn2B-b4BwrCNA",

"currency": "TON"

}

**Ответ.**

Например, для валюты TON он получит в ответ, что 1 TON можно приобрести за 180 руб. или 0.00009564 BTC.

{

"BTC": "0.00009564",

"RUB": "180"

}

**6. Обмен валют по установленному курсу.**

**POST запрос.**

Пользователь может обменять валюту, которая хранится у него на счетах по установленным курсам. Для этого в запросе нужно указать валюту, которую меняем, её количество, и валюту, которую хотим получить. Например, мы хотим купить TON`ы на 2000 рублей.

{

"secret\_key": "AAFeyWzOnlD-9G4i662PdKn2B-b4BwrCNA",

"currency\_from": "RUB",

"currency\_to": "TON",

"amoun": "2000"

}

**Ответ.**

Перед проведением обмена должна выполниться проверка, что у пользователя достаточно денег на счету. В случае успеха, сообщить сколько денег списано и сколько зачислено.

{

"currency\_from": "RUB",

"currency\_to": "TON",

"amount\_from": "2000",

"amount\_to": "11.11"

}

**API ДЛЯ АДМИНИСТРАТОРА.**

**1. Изменить курс валют.**

**POST запрос.**

Администратор может указать в запросе тип базовой валюты и цену за неё в других валютах. Например, он хочет установить курс для TON: 1 TON = 184 руб. и 1 TON = 0.000096 BTC. Обратите внимание, что у администратора свой собственный secret\_key, который уже внесён в систему (и ему не нужно регистрироваться) и дает доступ к данному API. Обычный пользователь со своим secret\_key, естественно, не должен иметь доступ к данному методу.

{

"secret\_key": "1071daaabf1cda35d207030c898d07ff16c934b7"

"base\_currency": "TON",

"BTC": "0.000096",

"RUB": "184"

}

**Ответ.**

В ответ должен возвращаться актуальные курс валют относительно базовой валюты.

{

"BTC": "0.000096",

"RUB": "184"

}

**2. Посмотреть общую сумму на всех пользовательских счетах для указанной валюты.**

**GET запрос.**

Администратор может запросить общую сумму на всех пользовательских кошельках в определенной валюте. Например, он хочет узнать, сколько всего на бирже хранится RUB.

{

"secret\_key": "1071daaabf1cda35d207030c898d07ff16c934b7",

"currency": "RUB"

}

**Ответ.**

{

"RUB": "15006813",

}

**3. Посмотреть количество операций, которые были проведены за указанный период (например, за последние сутки).**

**GET запрос.**

Администратор может посмотреть статистику по количеству операций на бирже. Для этого он должен отправить запрос с указанием временного отрезка.

{

"secret\_key": "1071daaabf1cda35d207030c898d07ff16c934b7"

"date\_from": "18.02.2023",

"date\_to": "19.02.2023"

}

**Ответ.**

{

"transaction\_count": "154"

}

**Будет плюсом:**

* подключить базу данных Sqlite или PostgreSQL для хранения данных о балансе пользовательских кошельков и истории операций;
* сервис по запросу может возвращать данные в json ИЛИ xml.
* Формат может быть изменен добавлением header
* **accept:application/json** или **accept:application/xml** к HTTP-запросу.
* подключить и настроить swagger;
* использовать Spring Security для разграничения ролей (admin/user);
* реализовать подтверждение регистрации на бирже посредством перехода по ссылке, которая будет отправляться в письме на указанный при регистрации эл. адрес;

**При отправке выполненной работы требуется:**

* снять короткий видео-ролик (до 60 сек) с демонстрацией работы приложения (запись экрана), загрузить его на диск-хранилище и ссылку добавить в письме;
* залить код с выполненным заданием в git репозиторий и ссылку добавить в письмо;
* добавить в корень репозитория README (в формате md или txt) с примерами REST-запросов и ответов и кратким описанием работы приложения (обязательно нужно указать, если были выполнены какие-то доп.задания).