**Руководство по запуску**

Проект состоит из трех отдельных сервисов: mailing, fething, finnhub.

**Mailing** – сервис рассылки уведомлений пользователям и проверки срока годности подписки.

**Fetching** – сервис обновления информации с finnhub.io, хранения и передачи главному приложению.

**Finnhub** – главное приложение, ответственное за регистрацию, авторизацию пользователей; администрацию пользователей; передачу пользователям информации о компаниях; подписки пользователей.

Для запуска сервисов необходимы maven, java 11, docker, и postgres базы данных(одна для fetching, вторая для mailing и finnhub).

**Настройка приложений в исходном коде**

Настройка приложений происходит в файлах с названием application.properties (../<project\_name>/src/main/resources/application.properties)

**Общие настройки для всех приложений**

Настройка доступа к базам данных

* После *“spring.datasource.url=”* необходимо указать к базе данных, в которой будет храниться информация конкретного приложения.
* После *“spring.datasource.username=”* необходимо указать имя пользователя вашей базы данных.  
  После *“spring.datasource.password=”* необходимо указать пароль к пользователю базы данных.
* После *“spring.datasource.driver-class-name=”* указывается драйвер для работы с базой данных (для postgres базы это org.postgresql.Driver).
* После *“server.port=”* указывается порт, который будет слушать приложение.

**Настройка отдельных сервисов**

**Fetching** (finnhub/fetching/src/main/resources/application.properties)

* После *“finnhub.token=”*  необходимо указать ваш токен для finnhub.io.
* В базе данных выполнить скрипт database.sql для создания таблиц базы данных (finnhub/fetching/src/main/resources/database.sql)

**Mailing** (finnhub/mailing/src/main/resources/application.properties)

Рассылка происходит через gmail (настройка аккаунта для произведения рассылки [тут](https://www.hostinger.ru/rukovodstva/kak-ispolzovat-smtp-server)).

* После *“spring.mail.username=”* указываем имя пользователя почты.
* После *“spring.mail.password=”* указываем пароль от почты.

**Finnhub** (finnhub/src/main/resources/application.properties)

* Для получения оплаты необходимо зарегистрировать аккаунт на paypal, получить clientId и secret и вставить их после соответствующих параметров *“paypal.clientId=”* и *“paypal.secret=”*
* В параметре *“paypal.returnLink=”* изменить *<localhost>* на адрес, на котором развернуто главное приложение(finnhub).
* В параметре *“mailing.path”* изменить *<localhost>* на адрес, на котором развернут сервис рассылки (mailing).
* В параметре *“fetching.path”* изменить *<localhost>* на адрес, на котором развернут сервис получения информации (fetching).
* По желанию отредактировать *“mailing.message...”* для установки желаемых сообщений, отправляемых пользователям.
* В базе данных выполнить скрипт database.sql для создания таблиц базы данных (finnhub/src/main/resources/database.sql)

**Запуск приложений**

Каждое приложение запускается из корневого каталога из командной строки (папка содержит /src, /target, pom.xml. Пример: mailing - ../mailing)

В командной строке после перехода в папку с приложением выполняются следующие команды

1. **mvn test** – выполнение тестов.
2. **mvn package** – создание .jar файла (создается в папке /target).
   1. Запуск приложения через консоль: **java -jar target\<project\_name>-2.4.4.jar**, где <project\_name> - название проекта: finnhub/fetching/mailing.
   2. Запуск приложения через doker:

# “docker build --tag=<project\_name> .”, где <project\_name> - название проекта: finnhub/fetching/mailing (команда содержит все символы между скобками без самих скобок, перед tag два знака “минус”).

* + 1. docker run --name <name> -p<port>:<port> <project\_name>.
* Вместо <name> задаем имя запущенному образу в docker(пример: “fetching”).
* Вместо <port> задается порт, который указан в параметрах сервиса (application.properties) (пример: … -p8081:8081 ...).
* Вместо где <project\_name> - название проекта: finnhub/fetching/mailing.

Перед запуском приложения необходимо создать минимум один аккаунт администратора.

Для этого необходимо зарегистрироваться в главном приложении и выдать роль зарегистрированному пользователю вручную.

Это можно сделать через HTTP-клиент postman:

* Создается новый запрос в postman и конфигурируется в соответствии с параметрами запроса, которые принимает контроллер в приложении(работает со всеми контроллерами в приложении).

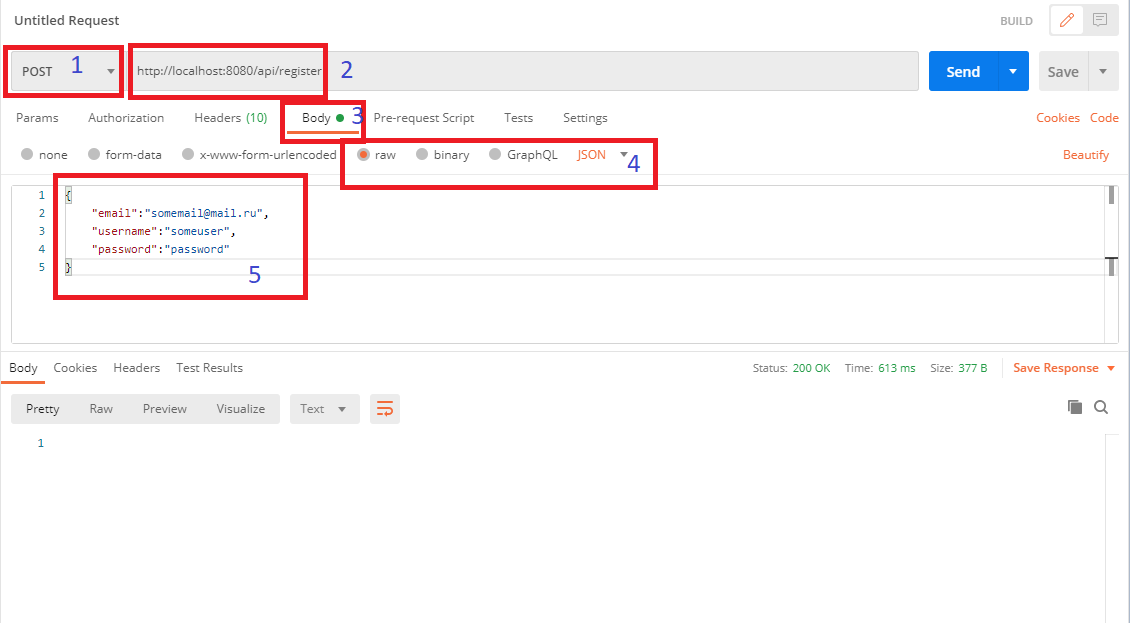


Рис. 1

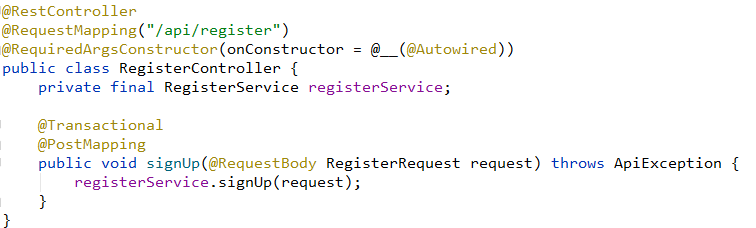


Рис. 2

Пример регистрации находится на картинке 1.

1. Указывается тип запроса(аннотация(*@PostMapping*) на методе контроллера(Рис. 2), ответственного за регистрацию).
2. url запроса к методу контроллера. (в случае изменения адреса сервера *localhost*  нужно заменить на новый адрес сервера, в случае изменения порта, прослушиваемого сервером – изменить порт(число, находящееся после “:”)). Через /указывается оставшийся адрес к методу (складывается путь из аннотации *@RequestMapping* над контроллером и аннотации над методом)
3. Настраивается тело запроса(только в случае с параметром @RequestBody в методе, обслуживающем запрос).
4. В настройках тела запроса выбирается тип*“raw”*  и формат *“json”*
5. В формате json записывается объект, который должен быть передан в теле запроса (в нашем случае этот тип – класс RegisterRequest, содержащий информацию о регистрируемом пользователе. Через запятую перечисляются пары название : значение параметров (название параметра должно соответстовать названию поля в классе) значение почты, пароля и имени пользователя можно менять; все значения должны находиться в кавычках).

При успешной регистрации в таблице *“user”* появится пользователь строка с переданными параметрами.

* После этого нужно выдать подписку с ролью “администратор” пользователю.

**Способ 1:**

В таблицу *“subscription”*  вставляется строка со следующими значениями:

* Столбец id должен быть таким же, как у зарегистрированного пользователя
* Столбец role имеет значение 4 (соотвествует роли “администратор”)

Далее небходимо сохранить изменения в таблице.

**Способ 2:**

Выполнить скрипт по вставке значений в таблицу

*INSERT INTO subscription (id, role)  
VALUES (<id>, 4);*

Где <id> - значение id зарегистрированного пользователя.

После этого созданный пользователь будет обладать правами администратора. Данный пользователь сможет создавать администраторов с помощью api для администратора.

**Рекомендации к действиям при первом запуске**

**с использванием Postman**

Сервисы fetching и mailing работают в автоматическом режиме, основная работа будет происходить с главным приложением (finnhub).

После запуска сервисов, fetching-сервису необходимо с помощью Postman отправить запросы на получение списка компаний и информации о них:

1. http://localhost:8082/api/companies/getall
2. http://localhost:8082/api/companies/refreshAllCompaniesInfo

(localhost:8082 – адрес и порт, указанные в конфигурации fetching-сервиса)

В постмане создаются и выполняются следующие запросы без параметров и тела запроса:

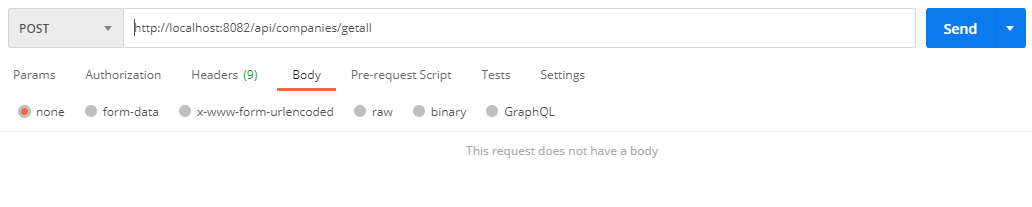
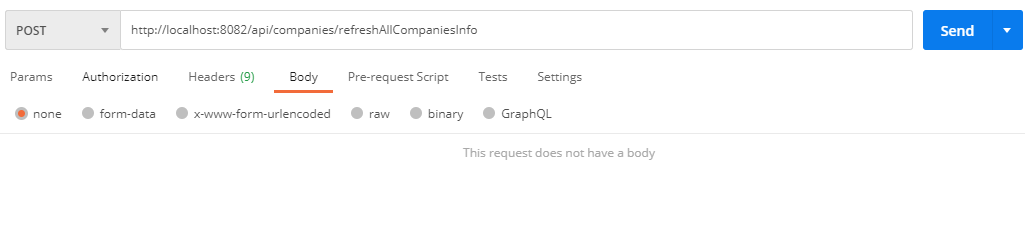


Рис. 3

(1 запрос: получение списка компаний)

Рис. 4

(2 запрос: обновление информации о каждой компании)

После этого нужно скопировать список компаний в главное приложение (finnhub).

Для этого администратору необходимо выполнить вход в приложение:

Нужно отправить запрос к главному приложению (*http://localhost:8080/login* с именем пользователя и паролем в теле запроса(по примеру регистрации Рис. 1; *localhost:8082* – адрес и порт, указанные в конфигурации главного приложения)

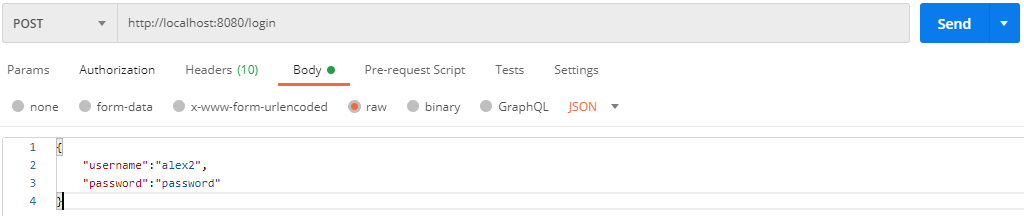


Рис. 5 (авторизация в приложении)

После успешной авторизации в ответе с сервера (нижняя половина окна с запросом в Postman) в списке *headers* можно найти хедер *Authorization.* (Рис. 6)

Необходимо скопировать его значение, так как этот хедер будет использоваться для каждого запроса к приложению для идентификации пользователя и предоставления ему прав доступа.

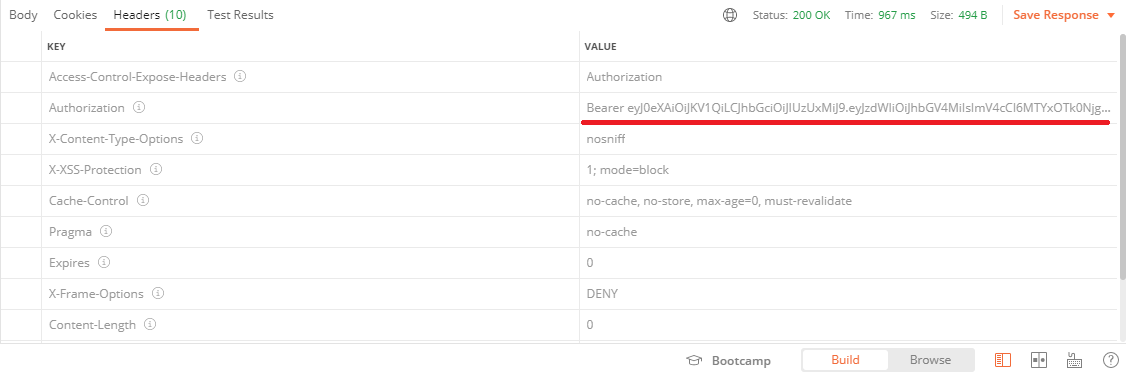


Рис. 6

После авторизации администратор может приступить к копированию списка компаний в базу данных главного приложения. Для этого ему необходимо отправить запрос в Postman по url *http://localhost:8080/api/admin/fetchAllCompanies*

К запросу необходимо прикрепить ранее полученный хедер, для этого в параметрах запроса нужно выбрать вкладку хедер и в новой строке в первую ячейку вставить имя хедера – Authorization, а во вторую сам хедер. (Рис. 7)

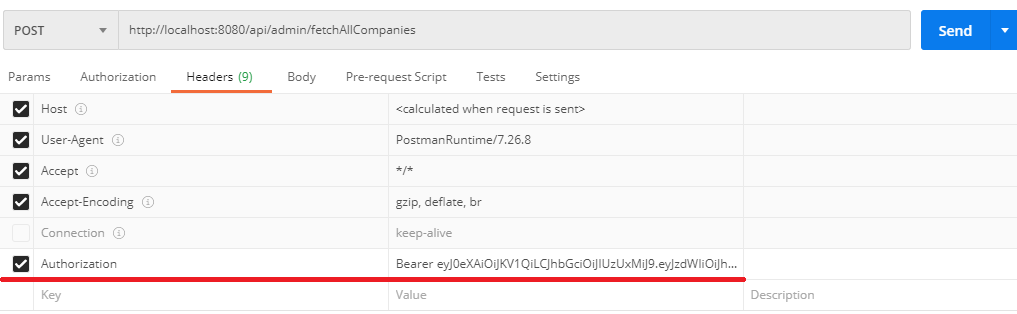


Рис. 7

В случае успешного завершения операции в таблицу *company* в главном приложении появится список компаний.

Приложение полностью готово к работе.

**Примеры работы с приложением**

**с помощью Postman**

Выше показаны примеры регистрации, входа в приложение и работа с некоторыми функциями приложение.

Рассмотрим еще несколько примеров.

**API администратора:**

Все запросы должны содержать хедер Authorization,

Запросы отправляются на адрес *http://localhost:8080/api/admin/<method\_address>* где localhost и порт – адрес и порт указанные в конфигурации, */api/admin* адрес контроллера администрации, *<method\_address>* – адрес, указанный над используемым методом в исходном коде.

**Пример совершения рассылки:**

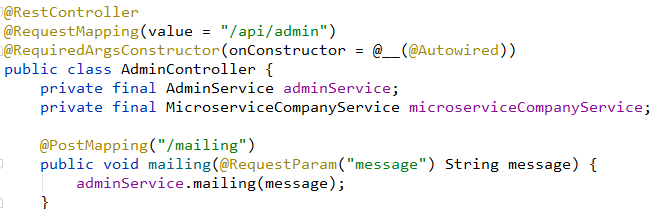


Рис. 8

Тип запроса – Post, адрес метода /mailing, параметр message – отправляемое пользователям сообщение. Вкладка *body* – выбираем тип *none* так как метод не принимает тело запроса(@RequestBody).

Для этого создаем новый запрос в Postman (см. выше требования к организации запроса)(Рис. 9)

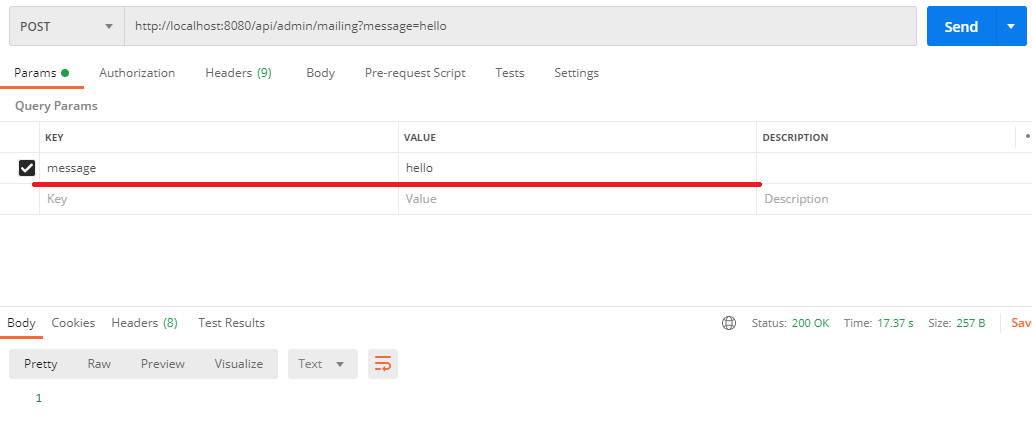


Рис. 9

Если в теле ответа (нижнаяя часть в окне запроса, вкладка *body*) нет информации об ошибке – значит запрос прошел успешно.

**Пример создания аккаунта для администратора:**

Тип запроса – Post, адрес метода /createAdminAccount, параметр message – отправляемое пользователям сообщение. Вкладка *params* пустая так как метод не принимает параметры запроса(@RequestParam).(Рис. 11)

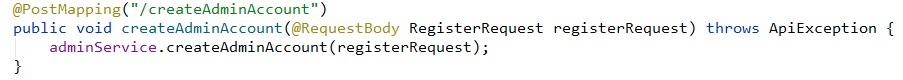


Рис. 10

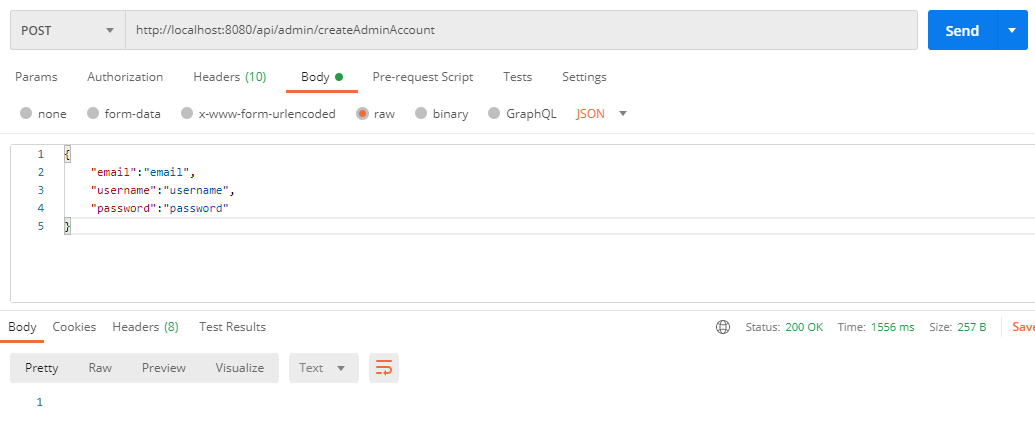


Рис. 11

В теле ответа нет информации об ошибке – запрос прошел успешно. Убедиться можно просмотрев таблицы *user*  и *subscription* в базе данных. Должен быть создан пользователь и подписка со значением *id* равным *id* пользователя и *role* со значением 4.

Остальные запросы аналогичны.

**API пользователя**

**Оплата подписки**

Отплата подписки происходит в несколько этапов:

* Отправка запроса на оплату подписки
* Перенаправление на сайт оплаты
* Подтверждение оплаты

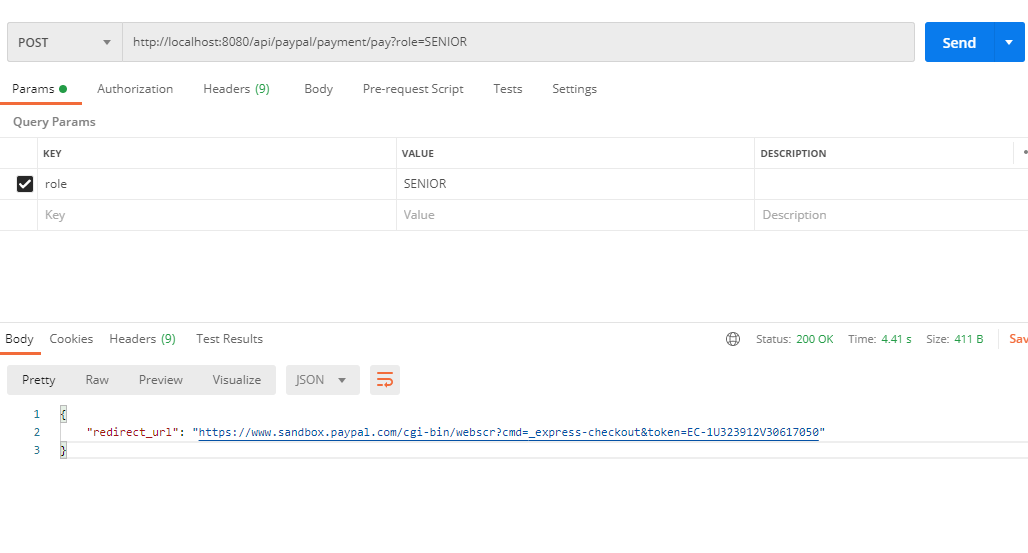
**Пример совершения оплаты:**

*http://localhost:8080/api/paypal/payment/pay*

Тип запроса – Post, адрес метода */payment/pay*, параметр role – тип подписки. HttpServletRequest request заполнится автоматически (Рис. 13).

****

Рис. 12

Рис. 13

В теле ответа пришла ссылка для перенаправления для оплаты.

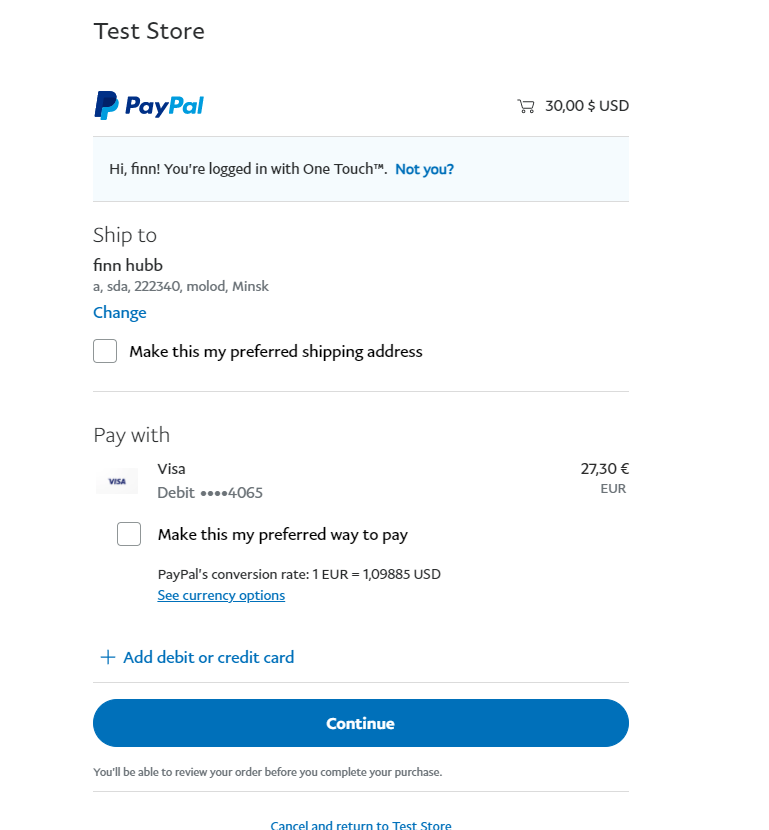


Рис. 14 (подтверждение оплаты)

В случае успешной оплаты произойдет перенаправление на http://<localhost>:<8080>/api/paypal/payment/complete?paymentId=\*\*\*.

**Пример начала отслеживания определенной компании:**

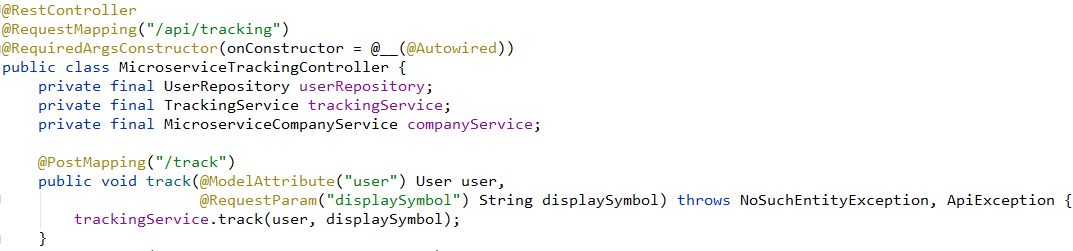
****

Рис. 15

*http://localhost:8080/api/tracking/track*

Тип запроса – Post, адрес метода */track*, параметр displaySymbol – тикер комании, которую нужно отслеживать. user заполнится автоматически исходя из хедера Authorization (Рис. 16).

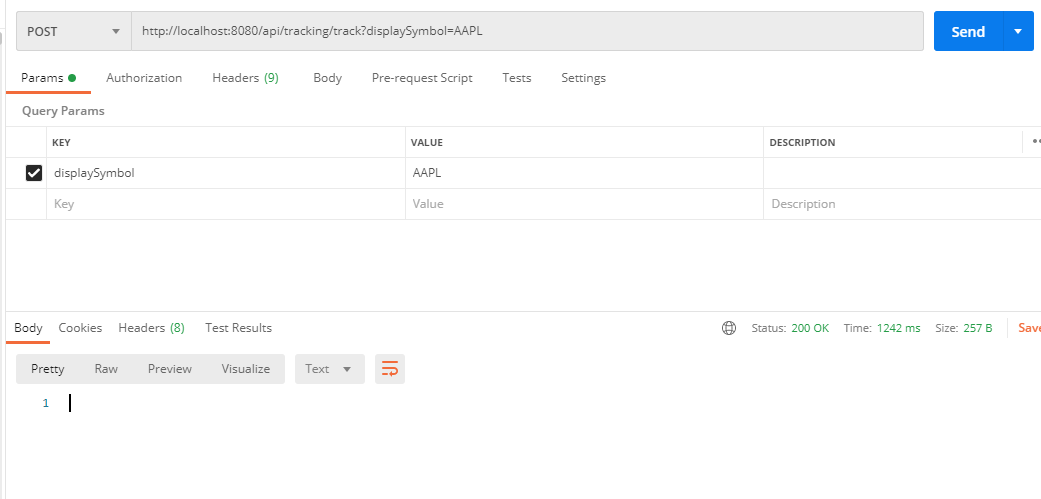


Рис. 16

**Пример получения данных о последних ценах акций отслеживаемой компании:**

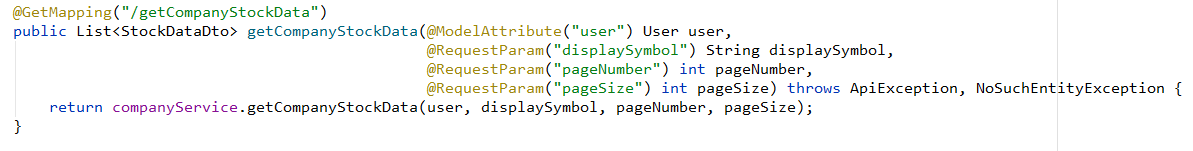


Рис. 17

*http://localhost:8080/api/tracking/getCompanyStockData*

Тип запроса – Get, адрес метода */getCompanyStockData*, параметр displaySymbol – тикер комании, акции которой необходимо получить. user заполнится автоматически исходя из хедера Authorization (Рис. 18).

В базе данных хранится много строк с данными об акциях компании, так как сохраняется каждое состояние цен в определенный момент, поэтому используются параметры *pageNumber* и *pageSize*, для получения конкретной части этих акций. *pageNumber* – номер страницы (первая страница - 0), которую нужно получить, *pageSize* количество строк, считываемых из базы данных в одну страницу.

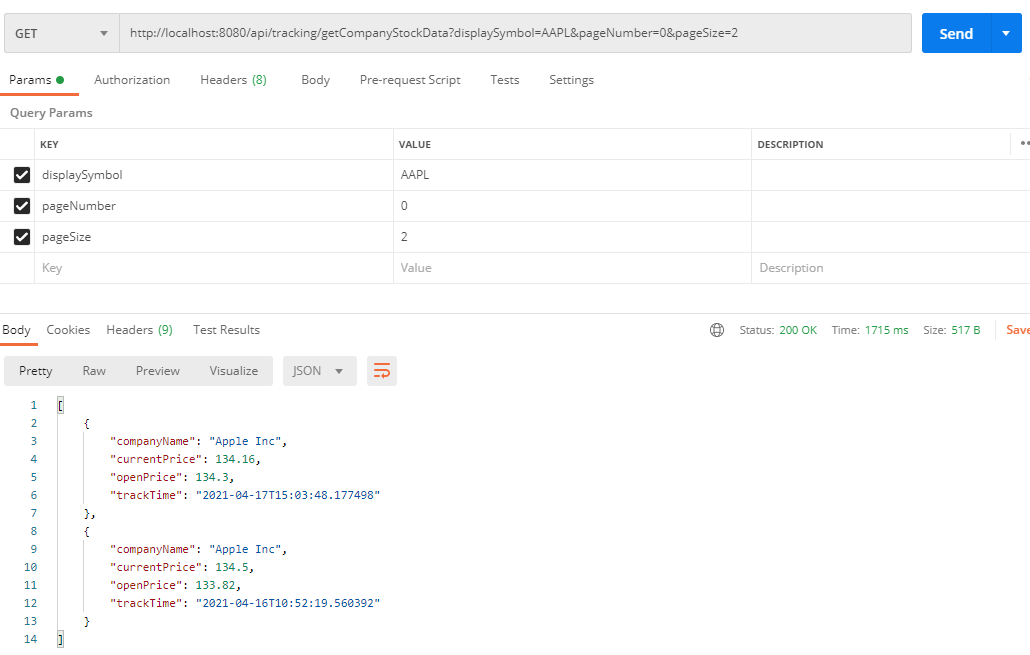


Рис. 18

Доступ к остальным функциям приложения аналогичен.