**Отчет по большому домашнему заданию №3 по курсу «Конструирование программного обеспечения»**

**Каспари Давид Эрвинович.**

В этом отчете я отражу результаты своей работы, опишу архитектуру написанной программы, приведу инструкции.

1. **Функционал.**

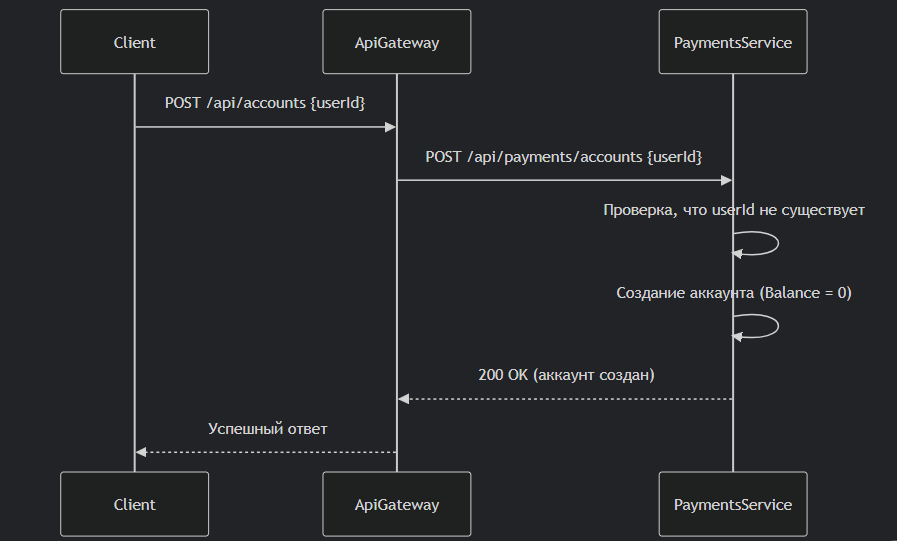
PaymentsService содержит функции создания, пополнения, просмотра баланса счета. OrdersService занимается обработкой заказа. ApiGateway отвечает за маршрутизацию.

1. **Микросервисная архитектура.**
2. API Gateway – отвечает только за routing запросов
3. Orders Service – отвечает за создание заказа и обработку
4. Payments Service – отвечает за создание, пополнение и просмотра баланса счета.
5. **Архитектура программы.**

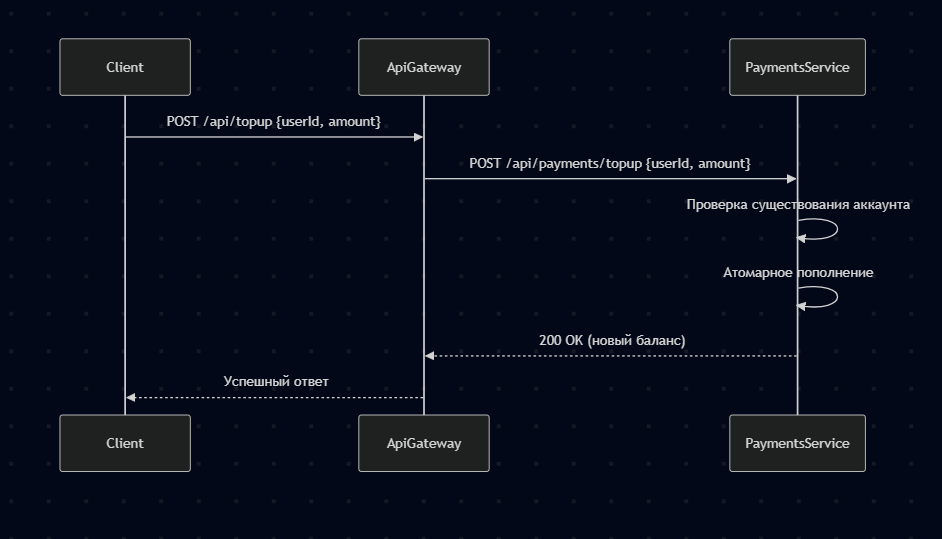
В Visual Studio созданы Веб-API Asp.Net.Core с названиями ApiGateway, OrdersService и PaymentsService.

Приведем некоторые схемы работы:

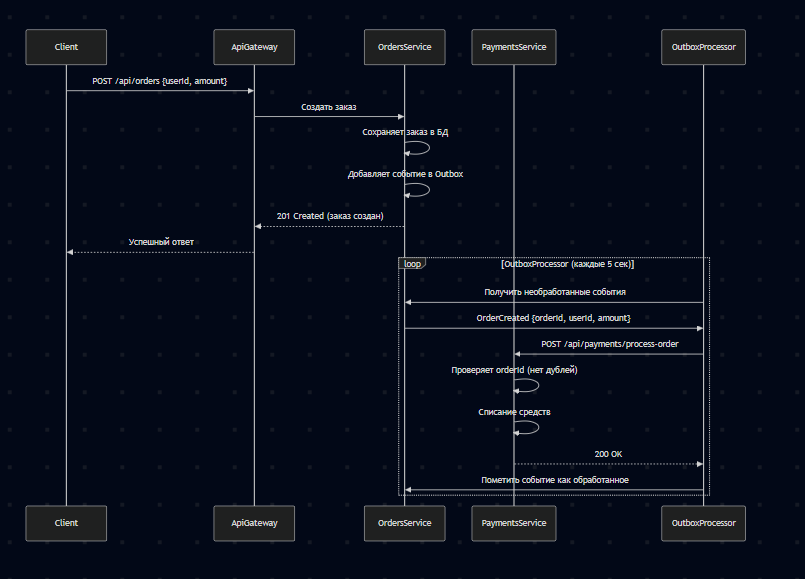
Создание аккаунта:



Пополнение баланса.



Процесс обработки заказа.

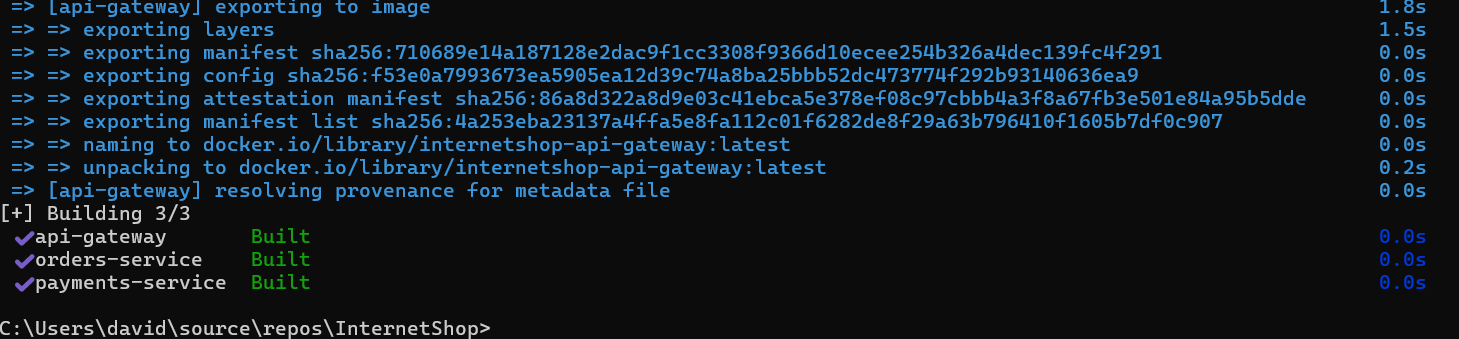
****

С конкретной реализацией алгоритма работы можно ознакомиться в файлах программы с помощью комментариев.

1. **Пример работы программы.**

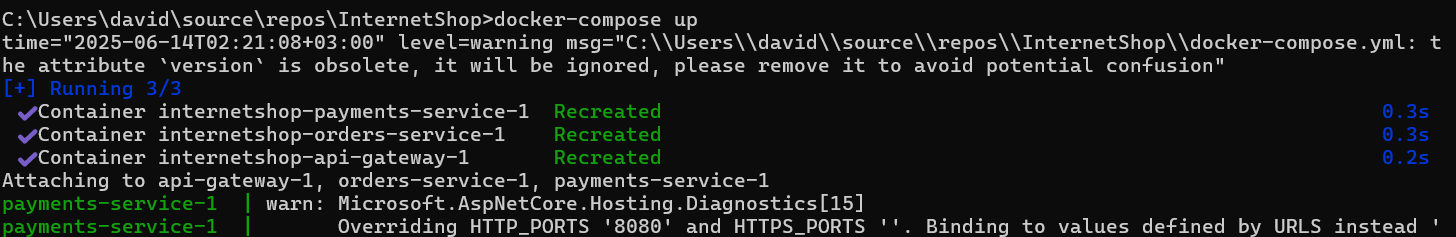
Приведем несколько примеров работы нашей программы.

Собираем ее с помощью Docker. После docker-compose build видим, что программа собирается.

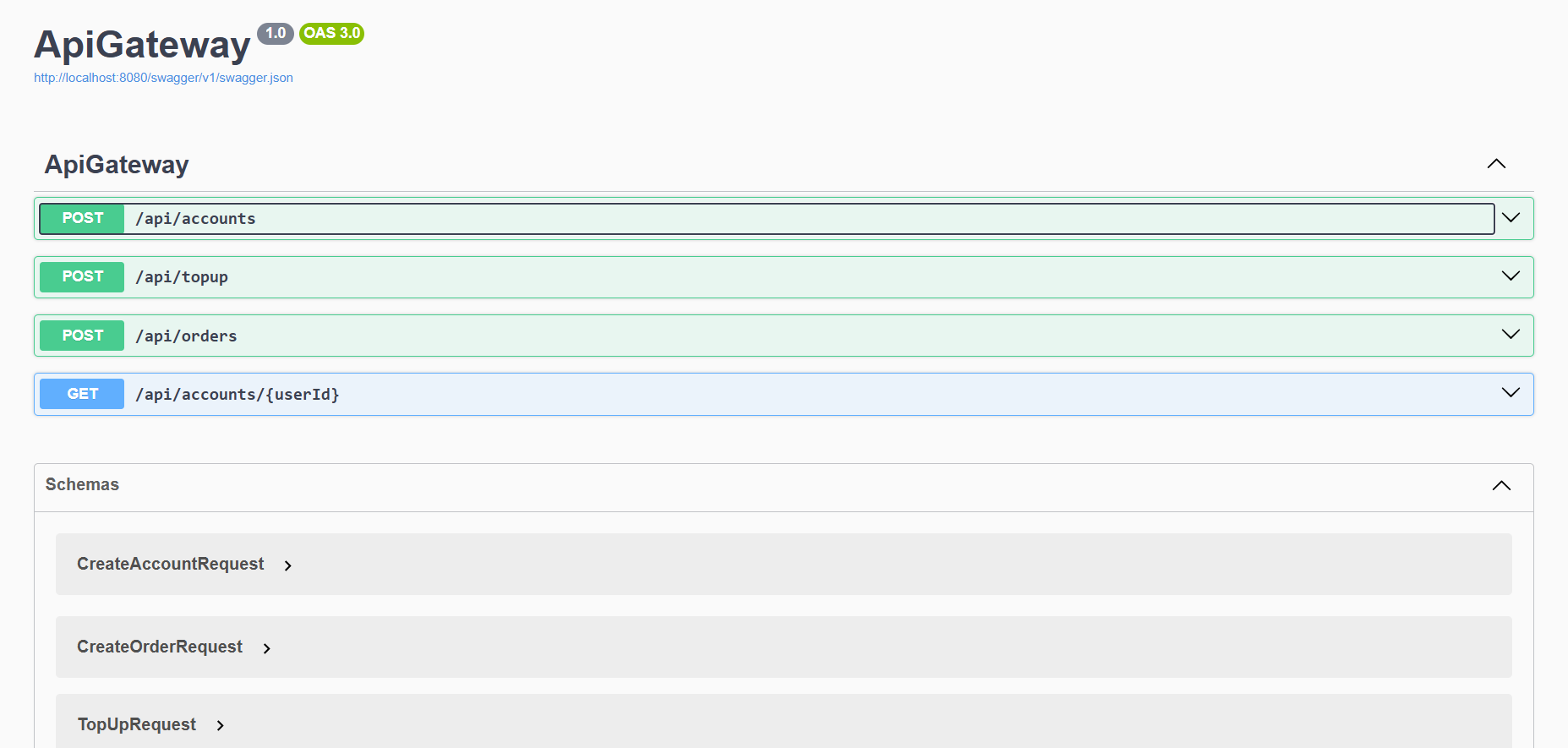


Запускаем с помощью docker-compose up.

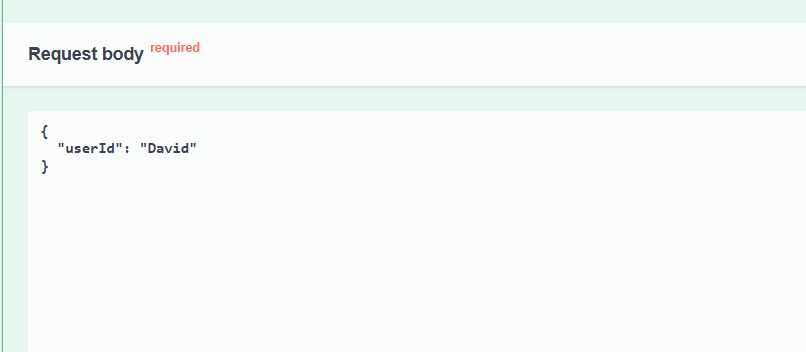
Здесь можно протестировать: <http://localhost:8080/swagger>

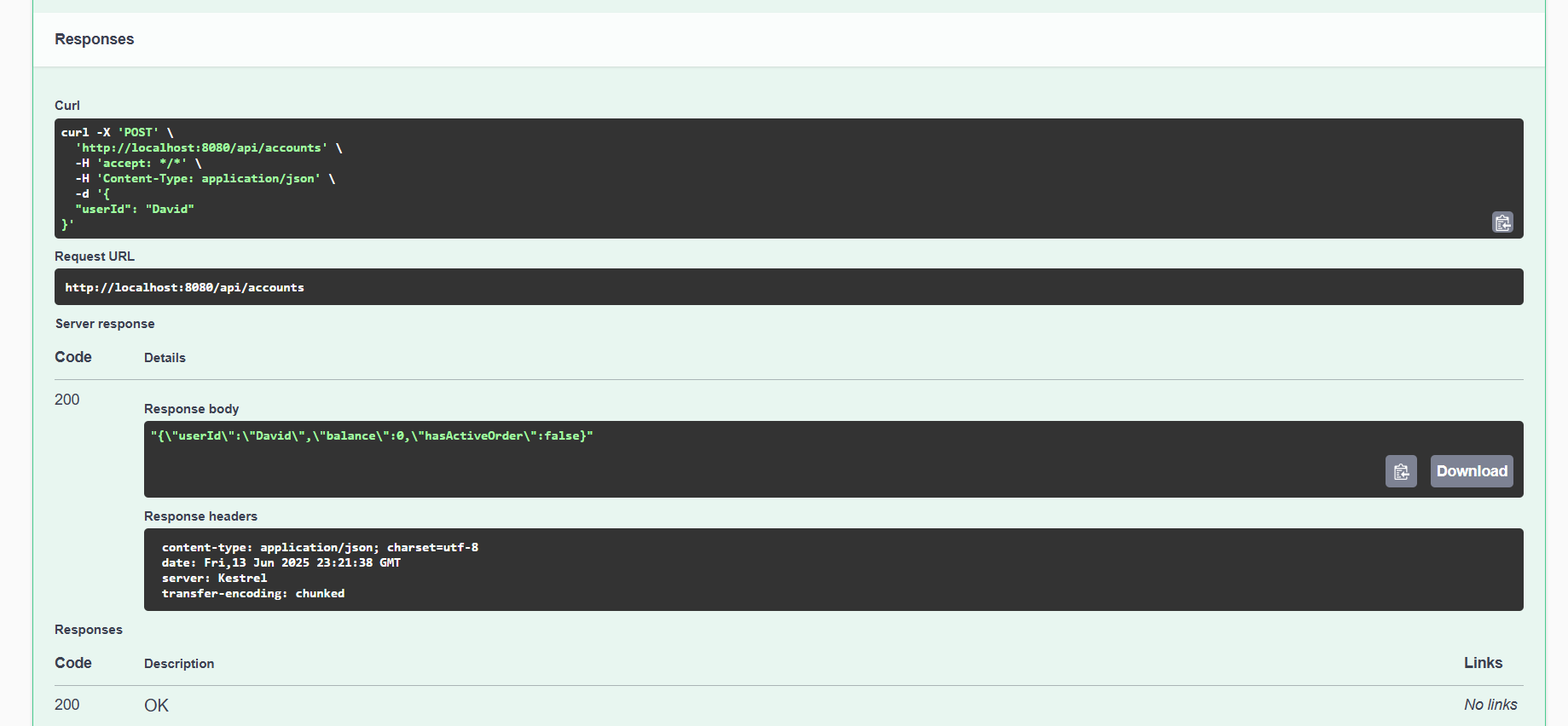


Открывается следующее окно:



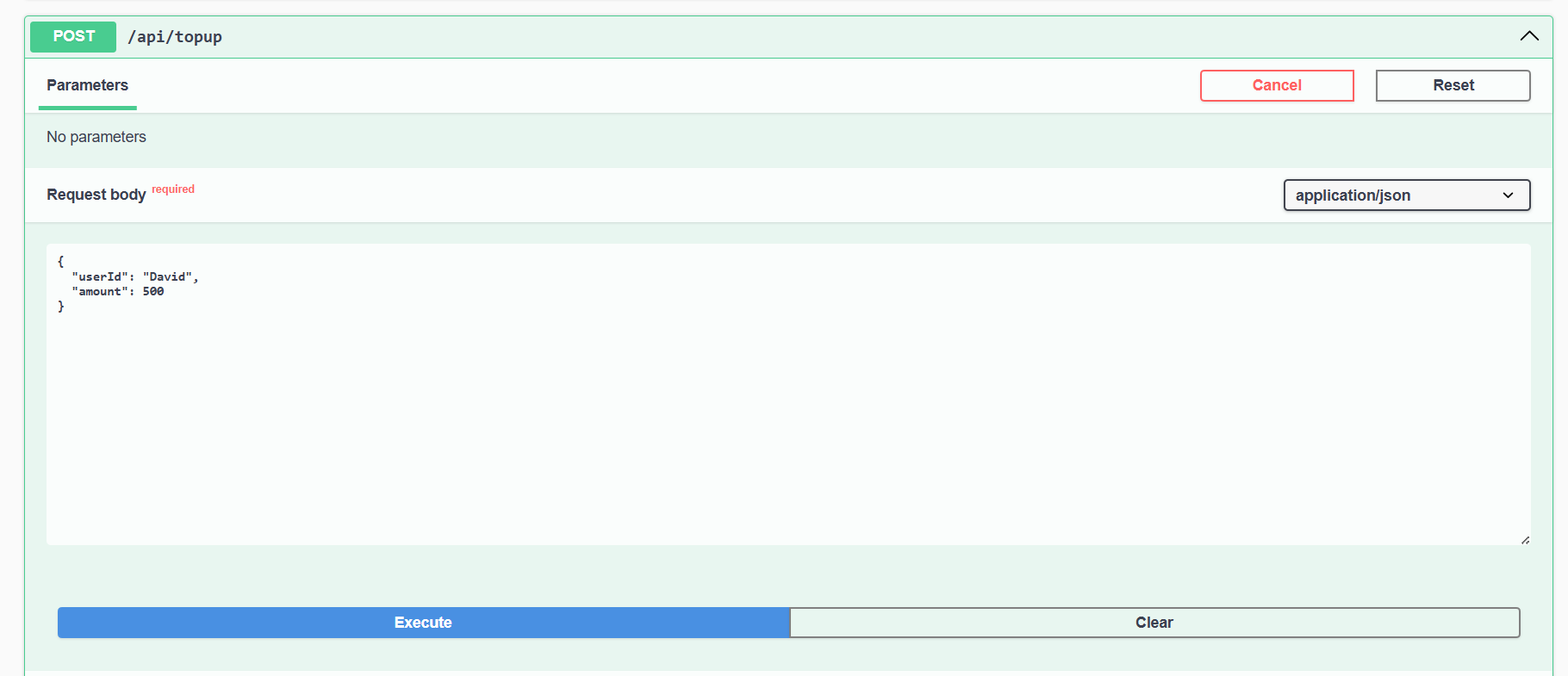
Введем первого user’а.



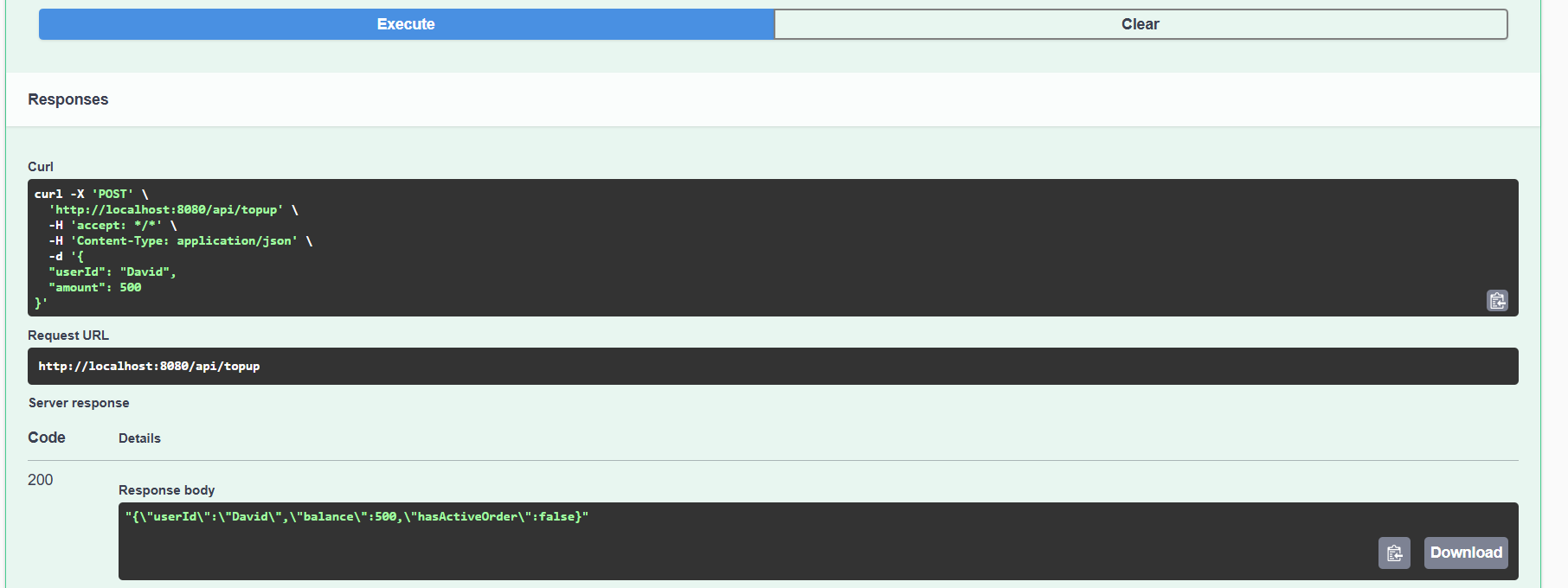


Видим, что пользователь зарегистрировался.

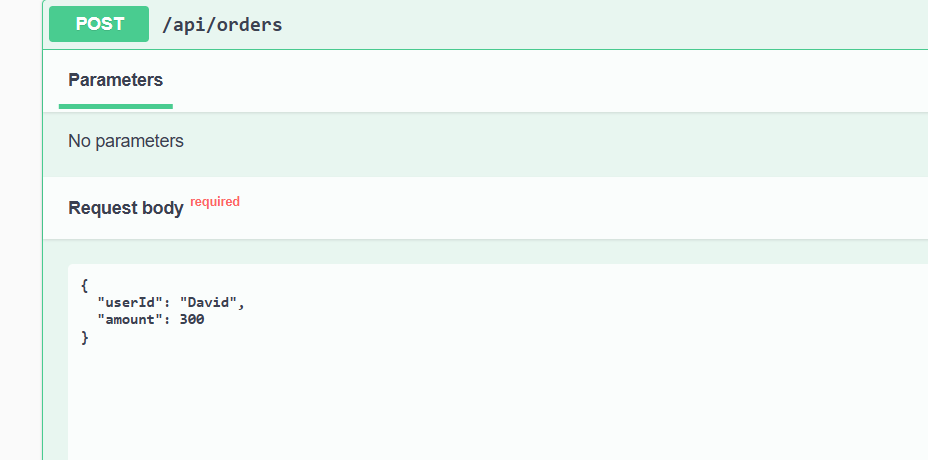
Далее сделаем пополнение на определенную сумму.



Видим, что аккаунт пополнен.



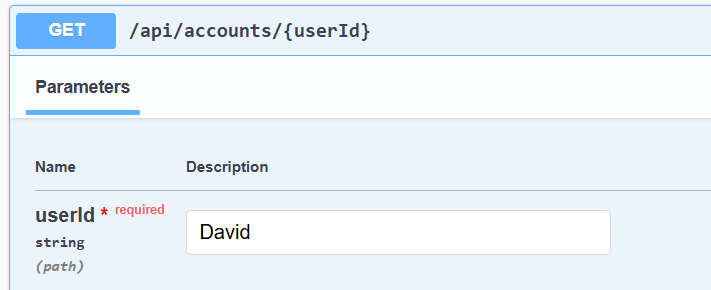
Далее формируем заказ на сумму 300.



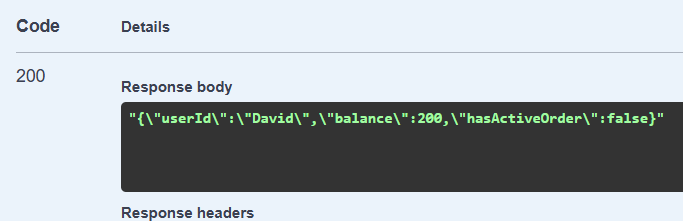
Видим, что заказ в процессе.



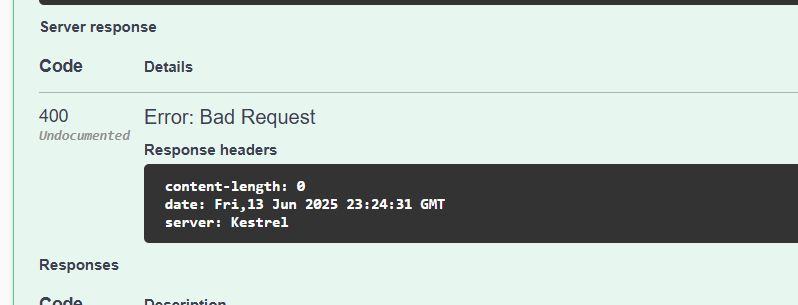
Далее спрашиваем, сколько осталось денег.



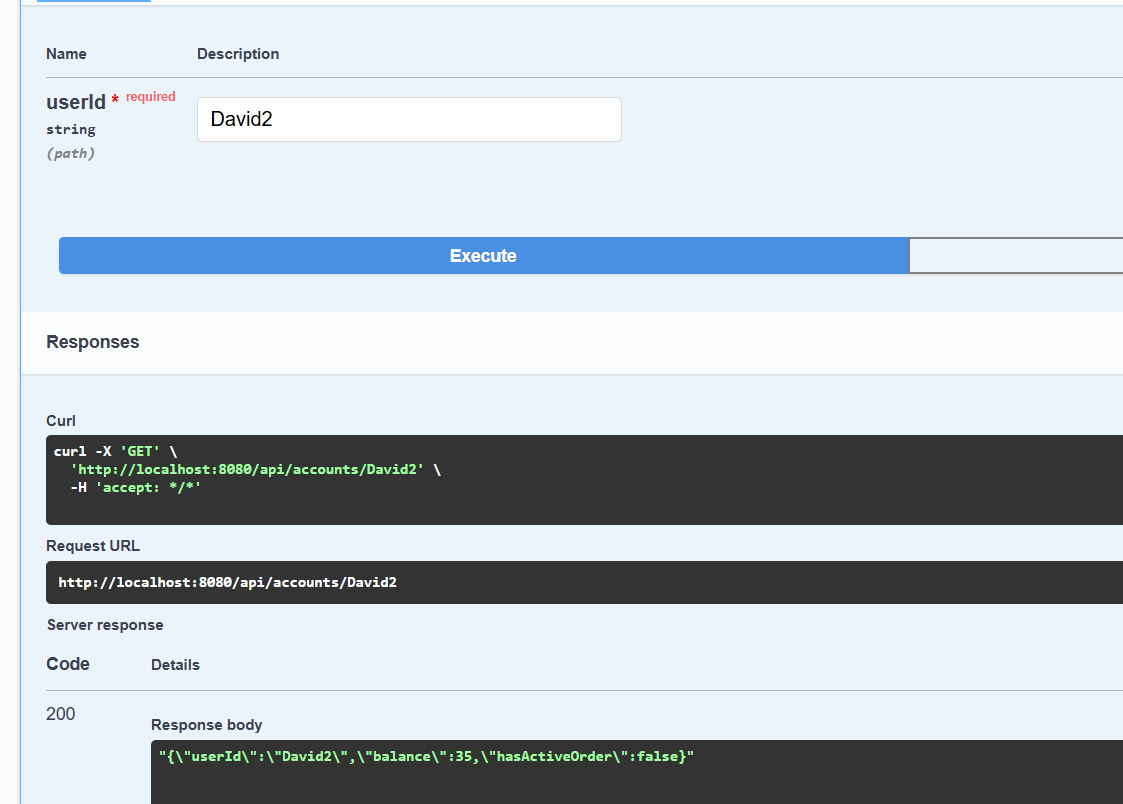
Видим, что 300, то есть заказ выполнился.

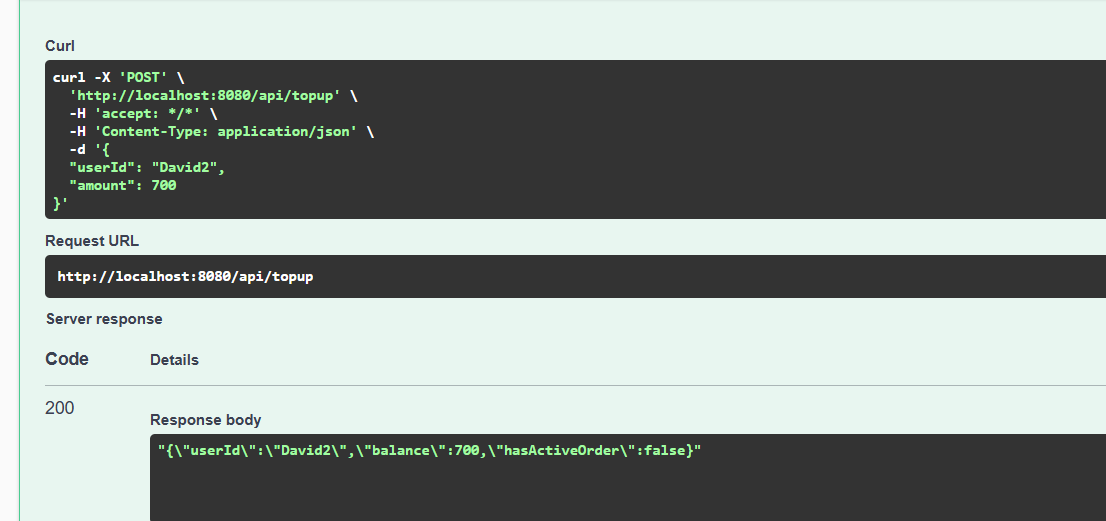


Если попытаться потратить больше, чем есть на аккаунте, то заказ не выполнится. Если создать аккаунт с таким же Id, то будет ошибка.



Создадим второго пользователя и потратим сумму 665. Видим, что осталось 35 при балансе 700.





1. **Выполнение требований к заданию.**

At most once – в PaymentsService если запрос приходит повторно, платеж не выполняется. Используется ConcurrentDictionary для проверки заказов.

Создание заказа работает асинхронно, процесс оплаты происходит через Outbox.

Реализованы требования к функционалу, есть четкое разделение на сервисы (описано выше).

В OrdersService используется Transactional Outbox. При создании заказа идет сообщение в Outbox. Подробнее в программе.

Swagger демонстрирует функциональность, работа с Docker корректна и описана выше.