SimbirSoft

Волга IT — Задание финального этапа Backend WebApi

Здравствуй, друг, и добро пожаловать на финальный этап конкурса Волга IT! Ты уже прошел через множество испытаний, и сейчас тебя ждет последний этап, который станет настоящим вызовом для твоих навыков программирования.

Ты выбрал для работы одну из доступных технологий: C#, Java, Python или GO. Теперь твоя задача — разработать полноценную систему для покупок в интернете под названием Simbir.Market, используя выбранный язык программирования.

Simbir.Market — это платформа для покупок в интернете. Ты должен создать систему, которая позволяет регистрировать новых пользователей, создавать и управлять продуктами, и самое главное создавать заказ на платформе. Безопасность и конфиденциальность данных — приоритетные задачи, а также система должна быть готова к масштабированию для работы с большим продуктов и платежей.

Ты готов к этому вызову? Удачи в разработке!

SimbirSoft

Общее описание задачи

В данном задании вы будете работать над созданием микросервисного приложения, которое охватывает различные аспекты программной инженерии, включая разработку микросервисов, конфигурацию API, использование баз данных и многое другое. Ваша цель — разработать ряд микросервисов, обеспечивающих функциональность для моделирования работы интернет магазина.

Архитектура приложения

Account microservice отвечает за авторизацию и данные о пользователях. Все остальные сервисы зависят от него, ведь именно он выпускает JWT токен и проводит интроспекцию.

Market microservice отвечает за данные о товарах. Отправляет запросы в микросервис аккаунтов для интроспекции токена.

Purchase microservice отвечает за покупку определенных продуктов на маркете. Отправляет запросы в микросервис аккаунтов для интроспекции токена и проверки существования связанных сущностей. Отправляет запросы в микросервис маркета для проверки количества определенных товаров на складе. Отправляет запрос в микросервис оплаты для создания нового платежа.

Payment microservice отвечает за оплату определенных продуктов на маркете. Отправляет запрос в микросервис покупки для завершения платежа.

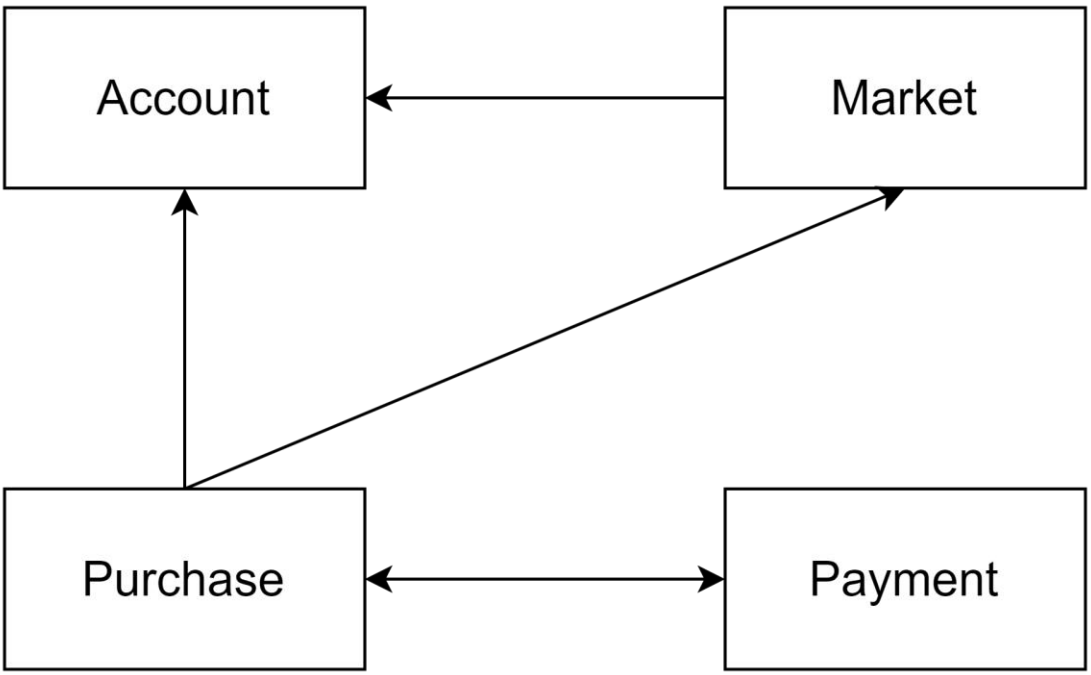

Рис.1. Диаграмма зависимостей микросервисов друг от друга.

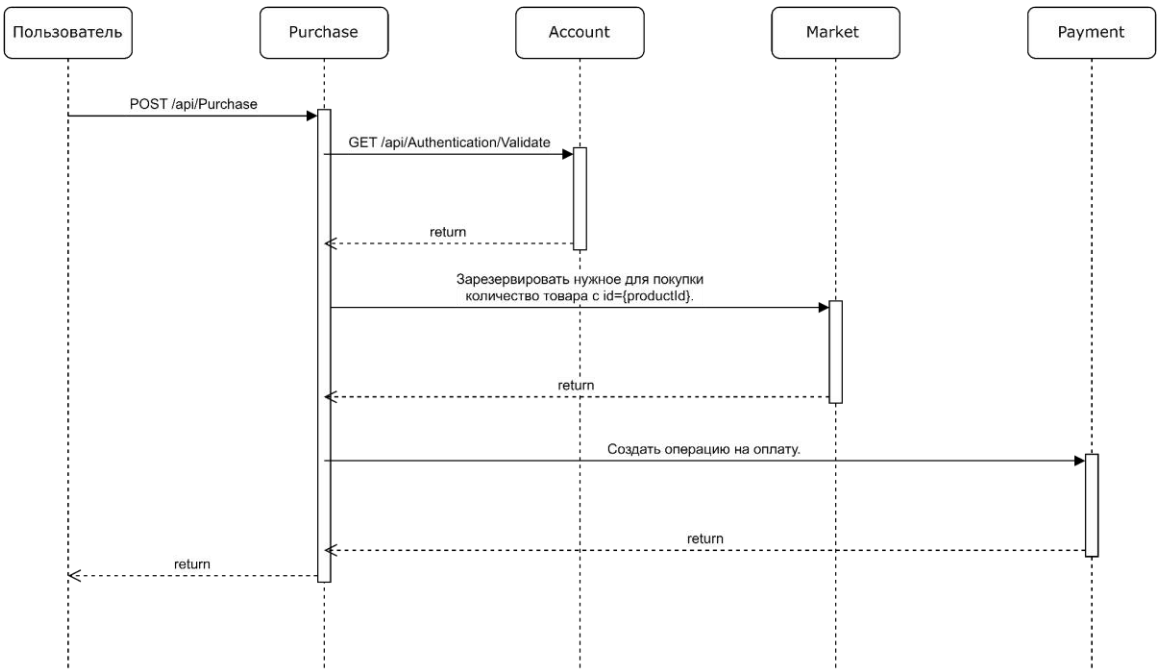

Рис.2. Пример межсервисных запросов в api.

SimbirSoft

Разработка приложения

Порядок выполнения разработки основного задания:

1. Необходимо разработать микросервис аккаунтов.
2. Необходимо разработать микросервис маркета.
3. Необходимо разработать микросервис покупок.
4. Необходимо разработать микросервис оплаты.

Ограничения основного задания:

1. Используйте базу данных PostgreSQL.
2. Для авторизации между сервисами используйте JWT.
3. Выберите один из методов взаимодействия микросервисов на свое усмотрение. Доступные методы: HTTP, gRPC, RabbitMQ или Apache Kafka.
4. Все API должны быть задокументированы с помощью Swagger с возможностью авторизации через JWT, ссылки на Swagger разместите в файле README.md.
5. При реализации сервера авторизации вы не должны использовать готовые решения по типу (identity server, keycloak и т.п.)
6. При выполнении команды “docker-compose up -d” ваши микросервисы не должны скачиваться с docker-hub, а должны билдится из исходников.

Развертывание приложения

Приложение должно быть контейнеризировано и запускаться через Docker. При проверке мы будем запускать единственную команду и ожидаем что все сервисы будут иметь предустановленные настройки и пользователей. Команда запуска:

```
docker-compose up -d
```

Предустановленные аккаунты по умолчанию:

№	Логин	Пароль	Роль
1	admin	admin	Admin
2	seller	seller	Seller
3	user	user	User

Валидные карты:

№	Номер	Имя	Дата	CVV
1	0000 0000 0000 0000	FOMA KINIAEV	08/69	007
2	0001 0002 0003 0004	DANILA BAGROV	13/37	420

SimbirSoft

Отправка задания

1. Выполните основную часть задания.
2. Оформите документацию в файле README.md с инструкциями по запуску.
3. Разместите ваш проект на одной из платформ: Github, Gitlab, Bitbucket, GitVerse.
4. Создайте папку source и скопируйте туда весь код приложения.
5. Создайте папку release и скопируйте туда рабочее приложение с файлом docker-compose.yml.
6. Запакуйте все папки в .zip архив с названием “Номер_Имя_Фамилия.zip” и переместите его на рабочий стол.
7. Обязательно сообщите организаторам о своем завершении задания.

Важно! Ваше приложение должно запускаться из папки release с помощью одной команды “docker-compose up -d”.

Ответы на вопросы

Если у вас возникли вопросы по заданию, то вы можете его задать непосредственно организаторам. Если ваше решение отличается от задания, то обязательно напишите все изменения в файл Readme.md.

Пример README.md

```
# Основное задание:
1. Account URL: http://localhost:8081/ui-swagger
2. Market URL: http://localhost:8082/ui-swagger
3. Purchase URL: http://localhost:8083/ui-swagger

# Дополнительная информация которую вы захотите указать
...
```

SimbirSoft

Account microservice

POST /api/Authentication/SignUp

описание: Регистрация нового аккаунта
body:
{
  "username": "string",
  "password": "string"
}
ограничения: Нет

POST /api/Authentication/SignIn

описание: Получение новой пары jwt пользователя
body:
{
  "username": "string",
  "password": "string"
}
ограничения: Нет

PUT /api/Authentication/SignOut

описание: Выход из аккаунта
ограничения: Только авторизованные пользователи

GET /api/Authentication/Validate

описание: Интроспекция токена
параметры:
accessToken: string
ограничения: Нет

POST /api/Authentication/Refresh

описание: Обновление пары токенов
body:
{
  “refreshToken”: “string”
}
ограничения: Нет

SimbirSoft

GET /api/Account/Me

описание: Получение данных о текущем аккаунте
ограничения: Только авторизованные пользователи

PUT /api/Account/Update

описание: Обновление своего аккаунта
body:
```
{
  "password": "string"
}
```
ограничения: Только авторизованные пользователи

GET /api/Accounts

описание: Получение списка всех аккаунтов
параметры:
from: int //Начало выборки
count: int //Размер выборки
ограничения: Только администраторы

POST /api/Accounts

описание: Создание администратором нового аккаунта
body:
```
{
  "username": "string",  //имя пользователя
  "password": "string",  //пароль
  "roles": [
    "string"  //массив ролей пользователя
  ]
}
```
ограничения: Только администраторы

PUT /api/Accounts/{id}

описание: Изменение администратором аккаунта по id
body:
```
{
  "username": "string",  //имя пользователя
  "password": "string",  //пароль
  "roles": [
```

```
    "string"                          //массив ролей пользователя
  ]
}
```

ограничения: Только администраторы

DELETE /api/Accounts/{id}

описание: Мягкое удаление аккаунта по id
ограничения: Только администраторы

SimbirSoft

Market microservice

POST /api/Products

описание: Создание новой карточки товара
body:
```
{
  “name”: “string”,
  “description”: “string”,
  “tags”: [
    “string”
  ]
}
```
ограничения: Только администраторы и продавцы

PUT /api/Products/{id}

описание: Изменение карточки товара
body:
```
{
  “name”: “string”,
  “description”: “string”,
  “tags”: [
    “string”
  ]
}
```
ограничения: Только администраторы и продавцы

GET /api/Products/{id}

описание: Получение карточки товара
ограничения: Только авторизованные пользователи

DELETE /api/Products/{id}

описание: Мягкое удаление карточки товара
ограничения: Только администратор или владелец товара

POST /api/Products/{id}/Increase

описание: Добавление на склад определенного количества товара.
детали: Метод доступен пользователю для облегчения тестирования.

параметры:
body:
{
  “amount”: int // Количество товара добавленного на склад.
}
ограничения: Только администратор или владелец товара. amount > 0

POST /api/Products/{id}/Decrease

описание: Удаление определенного количества товара со склада.
детали: Метод доступен пользователю для облегчения тестирования.
body:
{
  “amount”: int // Количество товара удалённого со склада.
}
ограничения: Только администратор или владелец товара. amount > 0

GET /api/Products

описание: Поиск товаров в маркете
параметры:
name: string // Фильтр имени (LIKE ‘%{*name*}%’) (может отсутствовать)
tags: array strings // Массив тегов каждый из которых должен быть у искомого товара. Если массив пустой, фильтрация по этому параметру не применяется.
ограничения: Только авторизованные пользователи

SimbirSoft

Purchase microservice

POST /api/Purchase

описание: Создание операции покупки товара.
детали: Покупка товара разделена на 2 этапа.
1) Начало покупки (резервирование товара)
2) Получение информации о покупке от сервиса оплаты (для упрощения задания это просто эндпоинт который будет описан ниже).
body:
{
“productId”: long // Идентификатор товара.
“amount”: int // Количество товара удалённого со склада.
}
ограничения: Только авторизованный пользователь. amount > 0. Количество товара на складе >= amount.

POST /api/Purchase/Cancel

описание: Отмена операции покупки.
детали: Отменяет незавершенную операцию. Возврат зарезервированного товара.
body:
{
“purchaseId”: long // Идентификатор транзакции.
}
ограничения: Только администратор и пользователь совершающий покупку.

GET /api/Purchase/{id}

описание: Получение статуса покупки.
ограничения: Только администратор и пользователь совершающий покупку.

GET /api/Purchase/History

описание: Получение истории покупок.
параметры:
userId: long //Идентификатор пользователя для которого получаем историю
from: int //Начало выборки
count: int //Размер выборки
ограничения: Только администратор и пользователь, чей id указан в запросе.

SimbirSoft

Payment microservice

POST /api/Payment/Pay/{paymentId}

описание: Оплата.
детали: Эмулирует списание средств с указанной карты. Отправляет подтверждение платежа в сервис покупки.
body:

```
{
  “cardNumber”: “string”,
  “expireDate”: “string”
  “cvv”: “string”
  “name”: “string”
}
```

ограничения: Принимает только карты описанные в документе.