/

├── api-gateway/ # Микросервис API Gateway (Go)

│ ├── cmd/ # Главная точка входа

│ │ └── main.go

│ ├── pkg/ # Основные бизнес-логики, маршруты и обработчики

│ │ ├── auth/ # Логика аутентификации

│ │ ├── gateway/ # Основная маршрутизация и обработка запросов

│ │ ├── middleware/ # Middleware для аутентификации, логирования

│ │ └── utils/ # Утилиты, такие как JWT

│ └── Dockerfile # Dockerfile для контейнеризации

│

├── streaming-service/ # Микросервис стриминга видео (Go)

│ ├── cmd/ # Главная точка входа

│ │ └── main.go

│ ├── pkg/ # Логика стриминга

│ │ ├── stream/ # Логика передачи видео (HLS)

│ │ └── utils/ # Утилиты для обработки видео

│ └── Dockerfile # Dockerfile для контейнеризации

│

├── subscription-service/ # Микросервис управления подписками (Java)

│ ├── src/ # Основной исходный код

│ │ ├── main/ # Основная точка входа

│ │ ├── controllers/ # Контроллеры (API эндпоинты)

│ │ ├── services/ # Логика управления подписками

│ │ ├── models/ # Модели данных (например, подписки)

│ │ ├── repository/ # Доступ к базе данных (JPA, Hibernate)

│ │ └── utils/ # Утилиты

│ ├── Dockerfile # Dockerfile для контейнеризации

│ └── application.properties # Настройки Spring Boot

│

├── movie-catalog-service/ # Микросервис каталога фильмов (Java)

│ ├── src/

│ │ ├── main/

│ │ ├── controllers/ # API для каталога фильмов

│ │ ├── services/ # Логика поиска, фильтрации фильмов

│ │ ├── models/ # Модели фильмов, жанров

│ │ ├── repository/ # Доступ к базе данных (JPA)

│ │ └── utils/ # Утилиты

│ ├── Dockerfile

│ └── application.properties

│

├── auth-service/ # Микросервис авторизации (Java)

│ ├── src/

│ │ ├── main/

│ │ ├── controllers/ # Контроллеры для регистрации, входа

│ │ ├── services/ # Логика аутентификации

│ │ ├── models/ # Модели пользователей

│ │ ├── repository/ # Доступ к базе данных (JPA)

│ │ └── utils/ # Утилиты (JWT)

│ ├── Dockerfile

│ └── application.properties

│

├── analytics-service/ # Микросервис аналитики (Java)

│ ├── src/

│ │ ├── main/

│ │ ├── controllers/ # API для получения аналитики

│ │ ├── services/ # Логика сбора и анализа данных

│ │ ├── models/ # Модели данных аналитики

│ │ └── utils/ # Утилиты

│ ├── Dockerfile

│ └── application.properties

│

├── docker-compose.yml # Конфигурация для запуска всех сервисов в Docker

├── .gitignore # Игнорируемые файлы для Git

├── README.md # Документация проекта

├── swagger/ # Документация API в формате Swagger

│ ├── swagger.yaml # Основной файл спецификации API

└── postman/ # Коллекция Postman для тестирования API

└── collection.json

**Объяснение структуры:**

* **api-gateway/** — Микросервис, который управляет всеми входящими запросами и маршрутизирует их на соответствующие микросервисы.
* **streaming-service/** — Микросервис для стриминга видео, возможно, с использованием протокола HLS.
* **subscription-service/** — Управляет подписками пользователей, проверяет активность подписок и взаимодействует с внешними платёжными провайдерами.
* **movie-catalog-service/** — Сервис для работы с каталогом фильмов, включая поиск, фильтрацию и сортировку.
* **auth-service/** — Сервис для аутентификации пользователей (регистрация, вход, создание JWT).
* **analytics-service/** — Сервис для сбора и обработки данных аналитики (например, количество просмотров, пользовательская активность и т.д.).

**Дополнительные файлы:**

* **docker-compose.yml**: Файл для настройки и запуска всех микросервисов с помощью Docker.
* **swagger/**: Содержит документацию для API всех микросервисов.
* **postman/**: Коллекция Postman для тестирования API.

**Основные аспекты:**

1. **Микросервисы на Go и Java**:
   * Go будет использоваться для сервисов с высокой нагрузкой, таких как API Gateway и стриминг видео.
   * Java для сервисов с бизнес-логикой, таких как авторизация, каталог фильмов и управление подписками.
2. **Docker и Docker Compose**:
   * Все сервисы будут контейнеризованы с помощью Docker, и для упрощения разработки и тестирования будет использоваться Docker Compose.
3. **Swagger**:
   * Документация API будет поддерживаться через Swagger, чтобы каждый сервис был хорошо документирован и легко интегрируем в проект.

Эта структура позволит разделить задачи между двумя разработчиками, с учётом их опыта, и будет достаточно гибкой для добавления новых сервисов в будущем.