**1. API Gateway (Go)**

**Описание:** API Gateway будет принимать все входящие запросы от клиентов и направлять их соответствующим микросервисам. Он также будет обеспечивать аутентификацию и маршрутизацию.

**Технические требования:**

* **Язык:** Go
* **Основной функционал:**
  + Маршрутизация запросов между микросервисами.
  + Проверка JWT токенов для аутентификации.
  + Поддержка CORS.
  + Логирование запросов.
  + Ответы на ошибки с кодами HTTP.
  + Интеграция с сервисами стриминга, авторизации и каталогом фильмов.
* **Технологии:**
  + **JWT аутентификация** (с использованием библиотеки для обработки JWT).
  + **HTTP/REST** (использование стандартных библиотек Go или фреймворков вроде Gin).
  + **Docker** для контейнеризации.

**2. Сервис стриминга (Go)**

**Описание:** Этот сервис отвечает за передачу видеофайлов пользователям через протокол HLS.

**Технические требования:**

* **Язык:** Go
* **Основной функционал:**
  + Обработка запросов на стриминг видео.
  + Протокол HLS для передачи видео.
  + Логирование ошибок и производительности.
  + Поддержка сессий пользователя (например, история просмотров).
  + Возможность масштабирования для поддержания нагрузки.
  + Работа с хранилищем видеофайлов (локальные или облачные сервисы, например, S3).
* **Технологии:**
  + **HLS** (HTTP Live Streaming).
  + **gRPC** или **REST API** для взаимодействия с другими сервисами (например, авторизация).
  + **Docker**.

**3. Сервис авторизации (Java)**

**Описание:** Сервис обработки регистрации, входа в систему и управления сессиями пользователей. Этот сервис будет генерировать JWT токены для безопасного взаимодействия с другими сервисами.

**Технические требования:**

* **Язык:** Java
* **Основной функционал:**
  + Регистрация нового пользователя.
  + Авторизация (вход по логину и паролю).
  + Генерация JWT токенов для аутентификации пользователей.
  + Валидация JWT токенов.
  + Возможность смены пароля, восстановления доступа.
  + Связь с PostgreSQL для хранения данных пользователей.
  + Логирование с подробным выводом ошибок.
* **Технологии:**
  + **Spring Boot** для быстрого создания REST API.
  + **JWT** для токенов.
  + **PostgreSQL** для хранения данных.
  + **Docker**.

**4. Сервис каталога фильмов (Java)**

**Описание:** Этот сервис будет управлять списком фильмов, предоставлять информацию о фильмах и их категориях.

**Технические требования:**

* **Язык:** Java
* **Основной функционал:**
  + CRUD операции для фильмов (создание, чтение, обновление, удаление).
  + Возможность поиска и фильтрации по жанрам, году выпуска, рейтингу и т.д.
  + Хранение информации о фильмах в базе данных PostgreSQL.
  + Предоставление метаданных о фильмах (описания, постеры, длительность).
  + Поддержка пагинации для больших объемов данных.
  + Логирование запросов и ошибок.
* **Технологии:**
  + **Spring Boot**.
  + **PostgreSQL**.
  + **Docker**.

**5. Сервис управления подписками (Java)**

**Описание:** Этот сервис управляет подписками пользователей, их статусами и интеграцией с платежной системой.

**Технические требования:**

* **Язык:** Java
* **Основной функционал:**
  + Регистрация новых подписок.
  + Проверка статуса подписки (активна ли, срок действия).
  + Интеграция с внешними платежными провайдерами (например, Stripe или PayPal).
  + Отправка уведомлений пользователям о сроках окончания подписки.
  + Хранение данных о подписках в PostgreSQL.
  + Логирование взаимодействий с внешними сервисами.
* **Технологии:**
  + **Spring Boot**.
  + **PostgreSQL**.
  + **Docker**.

**6. Сервис очередей (Go)**

**Описание:** Сервис будет использоваться для обработки асинхронных задач, таких как обновление данных, отправка уведомлений и другие фоновые операции.

**Технические требования:**

* **Язык:** Go
* **Основной функционал:**
  + Работа с **Redis** или **RabbitMQ** для обработки очередей.
  + Обработка фоновых задач (например, генерация отчетов, отправка уведомлений).
  + Логирование работы с очередями.
  + Обработка ошибок и повторение задач при сбоях.
* **Технологии:**
  + **Redis** или **RabbitMQ** для очередей.
  + **Go**.
  + **Docker**.

**Общие технические требования:**

* **Документация API:** Каждый сервис должен быть документирован с помощью **Swagger**.
* **Интеграция с Docker:** Все микросервисы должны быть контейнеризованы.
* **База данных:** Используется **PostgreSQL** для всех микросервисов, которые работают с данными.
* **Кэширование:** Использование **Redis** для кэширования часто запрашиваемых данных.
* **CI/CD:** Для автоматизации сборки и деплоя использовать **GitLab CI** или **GitHub Actions**.

Эти требования и задачи помогут вам правильно организовать и спланировать разработку каждого микросервиса. Если что-то нужно уточнить или добавить, сообщи!