**Карта разработки онлайн-кинотеатра**

**Этап 1: Подготовка и планирование**

1. **Обсудить архитектуру системы:**
   * Разделить бэкенд на микросервисы с балансом Go и Java (50/50).
   * Основные микросервисы:
     + **Go:** API Gateway, сервис стриминга, обработка очередей.
     + **Java:** сервис авторизации, каталог фильмов, управление подписками, аналитика.
2. **Определить взаимодействие между микросервисами:**
   * Выбрать протоколы общения между сервисами: REST или gRPC.
   * Спланировать структуру запросов и ответов API (использовать JSON).
3. **Настроить окружение:**
   * Установить необходимые инструменты:
     + Go и Java (JDK).
     + Docker для контейнеризации.
     + PostgreSQL (основная база данных).
     + Redis (для кэша и очередей).
     + Swagger для документирования API.
   * Создать репозитории для каждого микросервиса в Git.

**Этап 2: Реализация бэкенда**

**2.1: Базовая структура и настройка сервисов**

1. **Go:**
   * Разработать API Gateway:
     + Реализовать маршрутизацию запросов к микросервисам.
     + Добавить аутентификацию JWT.
   * Подготовить шаблон сервиса стриминга (заглушка без сложной логики).
2. **Java:**
   * Создать сервис авторизации:
     + Реализовать регистрацию, вход и выдачу JWT токенов.
     + Настроить связь с базой данных PostgreSQL.
   * Подготовить шаблон сервиса каталога фильмов (без поиска и фильтрации).
3. **Общие задачи:**
   * Настроить Docker Compose для запуска всех микросервисов.
   * Создать базовую структуру базы данных (миграции).
   * Протестировать взаимодействие между сервисами через Postman.

**2.2: Расширение функционала**

1. **Go:**
   * Реализовать базовую логику сервиса стриминга:
     + Передача видеофайлов через протокол HLS.
   * Настроить сервис для обработки очередей (с использованием Redis или RabbitMQ).
2. **Java:**
   * Добавить функционал поиска и фильтрации в сервис каталога фильмов.
   * Разработать сервис управления подписками:
     + Проверка активных подписок пользователя.
     + Интеграция с внешним платежным провайдером (заглушка).
3. **Документирование:**
   * Добавить описания всех API в Swagger.
   * Протестировать интеграцию всех сервисов (end-to-end тестирование).

**Этап 3: Тестирование и оптимизация**

**3.1:**

1. Провести нагрузочное тестирование:
   * Проверить производительность сервиса стриминга на Go.
   * Тестировать авторизацию и работу с базой данных для сервиса на Java.
2. Оптимизировать базу данных:
   * Настроить индексы в PostgreSQL.
   * Добавить кэширование с помощью Redis для популярных запросов.
3. Написать автоматические тесты для основных сценариев (unit и интеграционные).

**Этап 4: Разработка фронтенда**

**4.1:**

1. **Подготовка фронтенда:**
   * Настроить окружение React.
   * Разработать минимальные страницы:
     + Авторизация (вход/регистрация).
     + Главная страница с каталогом фильмов.
     + Страница фильма с плеером.
2. **Интеграция с API:**
   * Настроить взаимодействие фронтенда с бэкендом через Axios.
   * Подключить аутентификацию через JWT.

**4.2:**

1. Доработка интерфейса:
   * Улучшить навигацию (например, меню, ссылки на профиль).
   * Добавить фильтры и поиск на странице каталога фильмов.
2. Написать простые тесты для фронтенда (например, с помощью Jest).

**Этап 5: Финальные доработки и запуск**

**5.1:**

1. Провести интеграционное тестирование (фронтенд + бэкенд).
2. Настроить деплой:
   * Настроить CI/CD для автоматической сборки и развёртывания микросервисов.
   * Развёртывание приложения на облачной платформе (например, AWS, GCP или DigitalOcean).
3. Обратная связь и исправление багов.

**Итоговый результат**

* Полностью работающий онлайн-кинотеатр с балансом 50/50 между Java и Go.
* Реализованная авторизация, каталог фильмов, подписки и стриминг.
* Фронтенд с минимальным, но удобным интерфейсом.
* Задокументированный API для дальнейших улучшений.

Если понадобится помощь с любым этапом, обращайся!