**3. # ADR - Выбор архитектурного стиля**

Участники: Геворгян Арсен, Афанасьев Никита, Сахабутдинов Тимур

Дата: 25.09.24

Статус:  принят

**## Контекст**

Наш сервис должен поддерживать потоковое воспроизведение музыки с минимальными задержками, большое количество пользователей и одновременных подключений и высокую отказоустойчивость и безопасность данных пользователей. Необходимо выбрать архитектурный стиль, который будет соответствовать требованиям масштабируемости, управляемости и надежности.

**## Рассматриваемые варианты**

1. Монолитная архитектура

Плюсы: Простота разработки и развертывания, высокая производительность, легкость в отладке.

Минусы: Трудности с масштабированием отдельных частей, сложность поддержки при росте приложения.

1. Микросервисная архитектура

Плюсы: Легкость в масштабировании отдельных компонентов, независимость разработки и развертывания, высокая гибкость и адаптивность.

Минусы: Сложность в управлении и координации, потенциальные проблемы с производительностью из-за сетевых задержек.

1. Модульная архитектура

Плюсы: Упрощение разработки и поддержки, возможность независимой разработки и тестирования модулей, легкость в обновлении и замене модулей

Минусы: Необходимость в тщательном планировании и координации, возможные проблемы с интеграцией модулей.

**## Решение**

Мы выбрали микросервисную архитектуру.

**## Обоснование**

Микросервисная архитектура предоставляет высокую масштабируемость и гибкость, что является ключевым требованием для нашего музыкального стриминг-сервиса. Возможность независимого развертывания и обновления компонентов позволяет нам быстро адаптироваться к изменениям и новым требованиям. Легкость в интеграции новых технологий и языков программирования также является важным преимуществом.

**## Последствия**

**Плюсы**

* Высокая масштабируемость и гибкость.
* Возможность независимого развертывания и обновления компонентов.
* Легкость в интеграции новых технологий и языков программирования.

**Минусы**

* Сложность в управлении и координации.
* Необходимость в сложной инфраструктуре для взаимодействия между сервисами.
* Возможные проблемы с производительностью и задержками.

**## Риски**

Микросервисная архитектура требует сложной инфраструктуры для управления и координации сервисов.

Меры: Использование оркестраторов контейнеров (например, Kubernetes) и инструментов для управления микросервисами (например, Istio).

Взаимодействие между микросервисами может привести к задержкам и проблемам с производительностью.

Меры: Оптимизация сетевых запросов, использование кэширования и асинхронного взаимодействия.

Микросервисная архитектура может быть подвержена атакам и уязвимостям.

Меры: Использование современных методов шифрования, регулярное обновление программного обеспечения и проведение аудитов безопасности.

**##  Затронутые области**

* Инфраструктура: Необходимость в разработке и поддержке микросервисной инфраструктуры.
* Безопасность: Реализация мер безопасности для защиты данных пользователей.
* Масштабируемость: Обеспечение возможности горизонтального масштабирования.
* Отказоустойчивость: Реализация механизмов для обеспечения высокой доступности.

**## История**

25.09.24: Первоначальное обсуждение вариантов архитектурногостиля**.**