## ADR - Выбор архитектурного стиля для фотохостинга

Участники: Зыкин Алексей  
Дата: 5 декабря 2024  
Статус: -

**Контекст**

Разрабатывается приложение для фотохостинга, аналогичное Instagram, которое должно поддерживать большое количество пользователей и обеспечивать высокую доступность, масштабируемость и производительность. Приложение должно включать такие функции, как загрузка и обработка изображений, лайки, комментарии, подписки и лента новостей. Учитывая растущий объем данных и потенциальное увеличение числа пользователей, необходимо выбрать архитектурный стиль, который позволит эффективно справляться с этими задачами.

**Рассматриваемые варианты**

**Монолитная архитектура**  
Плюсы: Простота разработки и развертывания, легче тестировать и отлаживать.  
Минусы: Сложности с масштабированием, трудности при внесении изменений, риск возникновения «большого шара» (big ball of mud) при увеличении кода.

**Микросервисная архитектура**  
Плюсы: Высокая масштабируемость, возможность независимого развертывания и обновления сервисов, возможность использования различных технологий для разных сервисов.  
Минусы: Усложнение разработки и тестирования, необходимость в управлении распределенными системами, сложность в обеспечении безопасности и мониторинга.

**Модульная архитектура**  
Плюсы: Более структурированный подход, возможность выделить модули для улучшения тестирования и поддержки.  
Минусы: В некоторых случаях может быть сложно масштабировать отдельные модули независимо.

**Решение**

Выбрана **микросервисная архитектура**.

**Обоснование**

**Микросервисная архитектура** была выбрана из-за её способности масштабироваться и адаптироваться к изменениям в требованиях, что критично для приложения с потенциально большим количеством пользователей и данных.

**Последствия**

* Упрощение масштабирования отдельных компонентов
* Улучшение устойчивости системы
* Увеличение сложности управления
* Необходимость в более продвинутых инструментах для мониторинга и управления сервисами

**Риски**

Коммуникация: Потенциальные проблемы с коммуникацией между сервисами. Решением может быть внедрение API Gateway для управления взаимодействием между сервисами.  
Целостность: Сложность в обеспечении целостности данных. Для решения можно использовать паттерны для обеспечения согласованности данных (например, Saga), регулярно тестировать интеграций.

**Затронутые области**

* Архитектура приложения
* Процесс разработки
* Управление конфигурацией и развертыванием
* Системы мониторинга и логирования

**История**

Обсуждение началось с анализа требований к системе и возможных архитектурных стилей. Рассматривались плюсы и минусы каждого подхода. В результате было принято решение о выборе **микросервисной архитектуры**, так как она наилучшим образом соответствовала требованиям по масштабируемости и гибкости.