## ADR - Выбор языка программирования для серверной части

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

**Контекст**

В процессе разработки приложения для фотохостинга нам необходимо выбрать язык программирования для серверной части. Основные задачи включают создание высокопроизводительного, масштабируемого и безопасного API для обработки запросов пользователей, хранения и обработки изображений, а также интеграции с различными сервисами. Ключевыми ограничениями являются необходимость в быстром времени отклика, легкость в поддержке и наличие библиотек для работы с изображениями и базами данных.

**Рассматриваемые варианты**

**C++** - Высокопроизводительный язык, но сложен в поддержке.

**JavaScript (TypeScript)** - Хорош для веб-разработки, но может быть менее производительным для серверной части.

**Rust** - Высокая производительность и безопасность, но требует времени на изучение и внедрение.

**Kotlin** - Хорош для Android, но не так популярен для серверной части.

**C** - Высокая производительность, но низкий уровень абстракции и сложность в поддержке.

**C#** - Хорошая поддержка для веб-приложений через ASP.NET, высокая производительность и удобство разработки.

**Python** - Легок в изучении и разработке, но может быть менее производительным для высоконагруженных систем.

**Решение**

Выбран язык программирования **C#**.

**Обоснование**

**C#** был выбран по следующим причинам:  
Производительность: C# обеспечивает высокую производительность благодаря компиляции в промежуточный код и оптимизации выполнения.  
Поддержка: Существует множество библиотек и фреймворков (например, ASP.NET), что ускоряет разработку и упрощает интеграцию с другими сервисами.  
Сообщество: Большое сообщество разработчиков и множество ресурсов для решения проблем.  
Безопасность: C# предлагает встроенные механизмы для работы с безопасностью и управлением памятью.

Альтернативы, такие как **Rust** и **C++**, имеют высокую производительность, но требуют больше времени на обучение и могут усложнить процесс разработки. **Python**, хотя и прост в освоении, может не обеспечить необходимую производительность в условиях высокой нагрузки.

**Последствия**

* Ускорение разработки благодаря наличию готовых библиотек и фреймворков
* Упрощение интеграции с Microsoft Azure и другими облачными сервисами
* Потребность в обучении, если у команды недостаточно опыта работы с C#
* Высокая производительность
* Хорошая поддержка
* Удобные инструменты разработки

**Риски**

Нехватка опыта: Если у команды недостаточно опыта в C#, это может замедлить разработку. Планы по обучению помогут снизить этот риск.  
Зависимость от Microsoft: Использование технологий Microsoft может привести к зависимостям от их экосистемы. Необходимо следить за изменениями и оценивать альтернативы при необходимости.

**Затронутые области**

* Серверная часть приложения (API)
* Интеграция с базами данных
* Обработка изображений и работа с файловой системой

**История**

Данное решение было принято после рассмотрения плюсов и минусов различных языков. В результате, **C#** был выбран как наиболее подходящий язык для решения задач, стоящих перед проектом.