## ADR - Выбор языка программирования для клиентской части приложения фотохостинга

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

**Контекст**

При разработке клиентской части приложения фотохостинга необходимо учитывать высокие требования к производительности, удобству разработки и поддержке. Приложение должно быть кроссплатформенным, поддерживать современные веб-стандарты и обеспечивать плавный пользовательский интерфейс. Основные задачи включают:

* Разработка отзывчивого и интерактивного интерфейса
* Поддержка различных устройств (мобильные и десктопные)
* Возможность интеграции с серверной частью
* Легкость в поддержке и расширении функционала

**Рассматриваемые варианты**

**C++ (C)** - Высокая производительность, но сложность в разработке и поддержке, а также отсутствие кросс-платформенности для веб-приложений.

**JavaScript** - Низкая безопасность типов, что может привести к большему количеству ошибок в коде.

**TypeScript** - Надстройка над JavaScript, позволяет использовать удобные библиотеки и фреймворки, наличие статической типизации.

**Rust** - Высокая производительность, но сложность в освоении и низкая поддержка для веб-разработки.

**Kotlin** - Хороший выбор для Android, но не так распространен для веб-разработки.

**Python** - Удобен для разработки, но не оптимален для клиентской части веб-приложений.

**WASM** - Требует дополнительных усилий для интеграции с JavaScript.

**Решение**

Выбран язык программирования **TypeScript** для разработки клиентской части приложения.

**Обоснование**

**TypeScript** был выбран по следующим причинам:  
Типизация: TypeScript предлагает статическую типизацию, что уменьшает количество ошибок на этапе разработки и упрощает поддержку кода.  
Совместимость с JavaScript: TypeScript является надстройкой над JavaScript, что позволяет использовать существующие библиотеки и фреймворки, такие как React и Angular.  
Поддержка современных стандартов: TypeScript активно поддерживает новые возможности ECMAScript, что позволяет использовать современные подходы к разработке.  
Сообщество и экосистема: У TypeScript большое и активное сообщество, что обеспечивает доступ к множеству библиотек и инструментов.

**Последствия**

* Упрощение процесса разработки благодаря статической типизации
* Увеличение производительности команды за счет использования современных инструментов и библиотек
* Возможность легкой миграции кода на JavaScript при необходимости
* Уменьшение количества ошибок
* Улучшенная читаемость и поддерживаемость кода
* Необходимость обучения, если у команды недостаточно опыта работы с TypeScript

**Риски**

Обучение команды: Если у команды недостаточно опыта в TypeScript, это может замедлить разработку. Планы по обучению помогут снизить этот риск.  
Интеграция с существующими библиотеками: Возможны проблемы совместимости с некоторыми старыми библиотеками JavaScript. Необходимо тщательно проверять используемые библиотеки на совместимость.

**Затронутые области**

* Клиентская часть приложения, включая интерфейс и взаимодействие с серверной частью
* Инструменты сборки и конфигурация проекта

**История**

Были рассмотрены плюсы и минусы каждого из вариантов. Результат обсуждения различных языков программирования подтвердили целесообразность выбора **TypeScript** как оптимального решения для данного проекта.