

Мы уже говорили о том, что такое архитектура и распределённые системы (в уроке про CAP теорему).

**Архитектура приложения** - это комплекс решений, определяющих способы взаимодействия модулей приложения между собой и с внешним окружением. Архитектура включает подходы, ограничения и принципы, которым следует придерживаться при реализации веб-приложений

Современные веб-приложения сложны, и для того чтобы сложные системы функционировали предсказуемо, необходимо ими управлять. Архитектура помогает проектировать и управлять такими системами.

Людям сложно предсказать будущее и адаптироваться к изменяющимся требованиям и условиям (а они точно будут меняться в контексте веб-приложений, например сейчас есть тренд Mobile First, хотя раньше его не было). Хорошо спланированная архитектура облегчает расширение и модификацию системы.

Важно помнить, что архитектура и шаблоны проектирования - это средства, а не цель. Разработчики, аналитики, архитекторы должны использовать их по мере необходимости и оценивать преимущества и недостатки каждого инструмента, применяемого в проекте.

Давайте определимся с тем, что такое архитектурный паттерн.

**Паттерн, шаблон** (Pattern) — типичное решение определённой, широко распространённой проблемы. Отличается от *алгоритма* тем, что не содержит чёткой последовательности действий, а описывает общую концепцию (идею) решения, детальные реализации которой могут сильно отличаться.

Паттерны следует использовать только при необходимости, в противном случае они могут лишний раз усложнить архитектуру и реализацию приложения.

**Архитектурный паттерн** (Architectural Pattern) решает проблемы, связанные с архитектурными вопросами. (есть также паттерны эффективной разработки, паттерны оптимального тестирования и т. д.)

Мы с вами поговорим об архитектурных паттернах в контексте API. То есть, возьмем некоторые проблемы, с которыми может столкнуться проектировщик API (кто бы это ни был — аналитик, архитектор, разработчик) и как он может подойти к их решению.

На самом деле, мы уже рассказывали о некоторых архитектурных паттернах API, просто не обозначая их. Например: пагинация, rate limit, версионирование, как мы проектируем успешные ответы и ответы в случаях ошибок и т.д. Далее мы изучим важные верхнеуровневые архитектурные паттерны API.