**Архитектурные подходы и паттерны проектирования**

**Распределение архитектурных подходов по:**

1. **Ответственность между модулями**

1.1 MVC

1. **Насколько каждый модуль близок к бизнес-логике**

2.1 Трёхслойная архитектура

1. **Управление потоками данных в приложении**

3.1 Flux

1. **Компоновка модулей**
   1. Монолит
   2. Микросервисы

**Паттерны проектирования**

1. **Порождающие**
   1. Фабрика
   2. Фабричный метод
   3. Абстрактная фабрика
   4. Строитель
2. **Структурные**
   1. Адаптер
   2. Декоратор
   3. Фасад
   4. Прокси
3. **Поведенческие**
   1. Цепочка ответственности
   2. Стратегия
   3. Команда
   4. Наблюдатель

Архитектура приложения ­­– это набор решений о том, как модули приложения будут общаться друг с другом и с внешним миром. Архитектура включает в себя подходы: ограничения, правила которым надо следовать при написании кода.

Паттерн проектирования – шаблонное решение частой архитектурной проблемы. Паттерны проектирования помогают решать проблемы на более низком уровне, ближе к коду, а архитектура же решает проблемы проектирования всей системы в целом.

**Распределение архитектурных подходов по:**

1. **Ответственность между модулями**
   1. **MVC**

Самый известный из таких подходов Model-View-Controller. Он выделяет 3 «типа» задач, в соответствии с которыми назначает модулям роли. Элементарный пример – калькулятор, где Model – модуль, отвечающий за логику вычислений, View – UI калькулятора, Controller – модуль, валидирующий передаваемые данные с View на Model.

1. **Насколько каждый модуль близок к бизнес-логике**

Таким подходам важно, какая часть кода занимается непосредственно задачей приложения, а какая — инфраструктурными задачами. Например, в приложении для обработки фотографий бизнес-логикой были бы функции фильтров, а инфраструктурными задачами — обращение к API камеры телефона. В зависимости от степени близости к бизнес-логике такие подходы делят код на «слои». Самый распространённый подход среди таких — это [трёхслойная архитектура](https://doka.guide/tools/clean-architecture/).

* 1. **Трёхслойная архитектура (чистая архитектура?)**

1. **Управление потоками данных в приложении**

Определяет межмодульное взаимодействие: напрямую, опосредованно или с помощью специальных сервисов типа шины событий.

Известный подход – Flux и его распространённая реализация Redux.

* 1. **Flux**

1. **Компоновка модулей**
   1. **Монолит**
   2. **Микросервисы**

**Паттерны проектирования**

1. **Порождающие**

Данные паттерны проектирования помогают решать задачи с созданием сущностей или групп похожих сущностей. Примеры порождающих паттернов:

* 1. Фабрика
  2. Фабричный метод
  3. Абстрактная фабрика
  4. Строитель

1. **Структурные**

Помогают решать задачи по совмещению и сочетанию сущностей, определяют как сущности могут использовать друг друга. Пример структурных паттернов:

* 1. Адаптер
  2. Декоратор
  3. Фасад
  4. Прокси

1. **Поведенческие**

Распределяют ответственность между модулями и определяют, как именно будет происходить взаимодействие между модулями. Пример поведенческих паттернов проектирования:

* 1. Цепочка ответственности
  2. Стратегия
  3. Команда
  4. Наблюдатель