App architecture

Определение

Архитектура приложения — это как план строения для вашего дома, только для программы.

Она помогает определить, как разные части программы соединяются между собой, как они общаются, и что каждая часть должна делать.

Это важно, потому что, как и в строительстве дома, без плана все может превратиться в большую кучу неразберихи, где ничего не работает как надо.

Зачем:

- Понятная структура и код
- Простота изменения
- Эффективность программы
- Эффективность команды
- Разделение ответственности между слоями
- Как следствие - более простая отладка

Многослойная архитектура

В настоящее время в разработке ПО достаточно часто применяется многоуровневая архитектура или многослойная архитектура (n-tier architecture), в рамках которой компоненты проекта разделяются на уровни (или слои).

Классическое приложение с многоуровневой архитектурой, чаще всего, состоит из 3 или 4 уровней, хотя их может быть и больше, учитывая возможность разделения некоторых уровней на подуровни.

Слои

1 - Слой представления, с которым взаимодействует пользователь

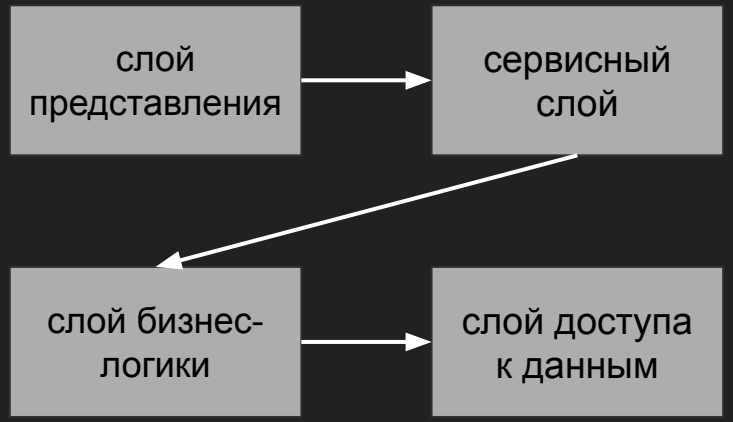
2 - Сервисный слой, реализующий взаимодействие между слоями представления и бизнес-логики.

3 - Слой бизнес-логики, в котором реализуется основная логика проекта. Компоненты, реализующие бизнес-логику, обрабатывают запросы, поступающие от компонентов сервисного слоя, а так же используют компоненты слоя доступа к данным для обращения к источникам данных.

4 - Слой доступа к данным — набор компонентов для доступа к хранимым данным. В качестве источников данных могут выступать различного рода базы данных, ВЕБ-сервисы, файлы на файловой системе и т.д.

Принципы

- Направление зависимостей между слоями идёт от слоя представления к слою доступа к данным
- В идеальной ситуации каждый слой зависит только от следующего слоя (компоненты бизнес-слоя могут зависеть от других компонентов бизнес-слоя)



MVC

MVC расшифровывается как model, view, controller.

Это способ организации кода, который предполагает выделение блоков, отвечающих за решение разных задач.

Один блок отвечает за данные приложения, другой отвечает за внешний вид, а третий контролирует работу приложения.

Модель — этот компонент отвечает за данные, а также определяет структуру приложения. Например, если вы создаете To-Do приложение, код компонента model будет определять список задач и отдельные задачи.

Представление — этот компонент отвечает за взаимодействие с пользователем. То есть код компонента view определяет внешний вид приложения и способы его использования.

Контроллер — этот компонент отвечает за связь между model и view. Код компонента controller определяет, как сайт реагирует на действия пользователя. По сути, это мозг MVC-приложения.

MVC на примере

Рассмотрим как выглядела бы реализация
первой проектной работы первой команды
при использовании MVC (см. запись)

Контрольная точка

- Понятно ли?

Если все ясно, ставим плюсы, иначе - задаем вопросы.

Домашнее задание

-

The end