**2. ВВЕДЕНИЕ В ASP.NET WEB API**

2.1. Организация структуры проекта по Clean Architecture

Организация структуры проекта по Clean Architecture обеспечивает модульность, чистоту и независимость компонентов приложения. Вот примерная организация структуры проекта по принципам Clean Architecture:

Domain Layer (Доменный слой):

В доменном слое содержится бизнес-логика приложения и определяются основные сущности и правила их взаимодействия. Этот слой должен быть независимым от инфраструктурных деталей и представления данных.

Сущности (Entities): Определяют основные объекты предметной области и их атрибуты. Например, для аниме-сайта это могут быть объекты "Анимационный сериал", "Манга", "Персонаж" и т.д.

Бизнес-правила (Business Rules): Включают в себя правила и ограничения, которые определяют допустимые состояния и операции над сущностями. Например, правила валидации данных, логика обработки заказов или рассчета скидок.

Infrastructure Layer (Инфраструктурный слой):

Инфраструктурный слой содержит реализации абстракций из доменного слоя и обеспечивает взаимодействие приложения с внешними системами и ресурсами. Этот слой включает в себя репозитории для работы с базой данных, клиентов API, а также другие инструменты и компоненты, необходимые для взаимодействия с внешними ресурсами.

Примеры компонентов в инфраструктурном слое:

Репозитории (Repositories): Реализации интерфейсов репозиториев, определенных в доменном слое. Они обеспечивают доступ к данным из базы данных или других источников данных.

Сервисы доступа к данным (Data Access Services): Компоненты, которые реализуют логику доступа к данным, такую как выполнение SQL-запросов, работа с ORM или обращение к внешним API.

Внешние сервисы (External Services): Клиенты для взаимодействия с внешними системами, такими как сервисы сторонних API или другие микросервисы.

Такая организация позволяет отделить бизнес-логику от инфраструктурных деталей, обеспечивая высокую гибкость, тестируемость и переносимость приложения.

2.2. Создание моделей приложения

Создание моделей приложения для аниме сайта включает в себя определение основных объектов данных, которые будут использоваться в приложении. Ниже приведен пример возможных моделей для аниме сайта:

1. Модель Anime (Аниме):

- Название аниме

- Описание

- Год выпуска

- Жанр

- Рейтинг

2. Модель Genre (Жанр):

- Название жанра

3. Модель Studio (Студия):

- Название студии

- Описание

4. Модель Viewing (Просмотр):

- Текст обзора

- Оценка

- Пользователь, оставивший обзор

- Аниме, к которому относится обзор

После определения моделей необходимо создать соответствующие миграции для базы данных, чтобы таблицы для каждой модели были созданы. Далее можно начать работу с данными, добавляя новые записи, редактируя существующие и выводя их на страницы сайта.

Также важно учитывать связи между моделями, например, связь "многие ко многим" между аниме и жанрами или "один ко многим" между аниме и персонажами. В зависимости от функциональности сайта можно добавить дополнительные модели и поля для более полного описания данных.

Не забывайте также обеспечить безопасность данных, например, с помощью проверки прав доступа к определенным действиям или данным пользователями.

*Заключение:*

*Организация структуры проекта по принципам Clean Architecture позволяет создать модульное, чистое и независимое приложение. Доменный слой содержит бизнес-логику и определяет основные сущности и правила их взаимодействия, в то время как инфраструктурный слой обеспечивает взаимодействие приложения с внешними системами и ресурсами.*

*Создание моделей приложения для аниме сайта играет ключевую роль в определении основных объектов данных, используемых в приложении. Примеры моделей включают аниме, жанры, студии и обзоры. Важно не только создать модели, но и учесть их связи и обеспечить безопасность данных.*

*Применение принципов Clean Architecture и создание соответствующих моделей позволяют разрабатывать гибкие, масштабируемые и легко поддерживаемые приложения, отвечающие потребностям пользователей и бизнес-задачам.*