**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

# **Дисциплина:** Бэк-энд разработка

Отчет

Лабораторная работа №1

Выполнил:

Крохин Владимир

БР1.1

Проверил:

Добряков Д. И.

Санкт-Петербург

2025 г.

**Задача**

Нужно написать свой boilerplate на express + TypeORM + typescript.

Должно быть явное разделение на:

* модели
* контроллеры
* роуты

**Ход работы**

В ходе выполнения лабораторной работы были переработаны контроллеры, роутеры и модели-сущности ORM из ДЗ№1.

Был применен подход, взаимствующий элементы концепций “Чистой архитектуры” (Clean Architecture) и предметно-ориентированного проектирования (Domain Driven Design (DDD)).

Структура бойлерплейта выглядит следующим образом:

src/

├── application

│ ├── domain

│ └── services

├── app.ts

├── bootstrap.ts

├── config

├── dtos

├── infrastructure

│ ├── repositories

│ │ ├── fakes

│ │ ├── interfaces

│ │ └── typeorm

│ │ ├── models

│ └── services

├── presentation

│ └── expressjs

│ ├── controllers

│ ├── middlewares

│ ├── routes

│ └── types

├── swagger.ts

├── tests

│ ├── integration

│ │ ├── controllers

│ │ └── repositories

│ │ └── typeorm

│ ├── setup.ts

│ └── unit

│ ├── business-services

│ ├── controllers

│ ├── models

│ └── services

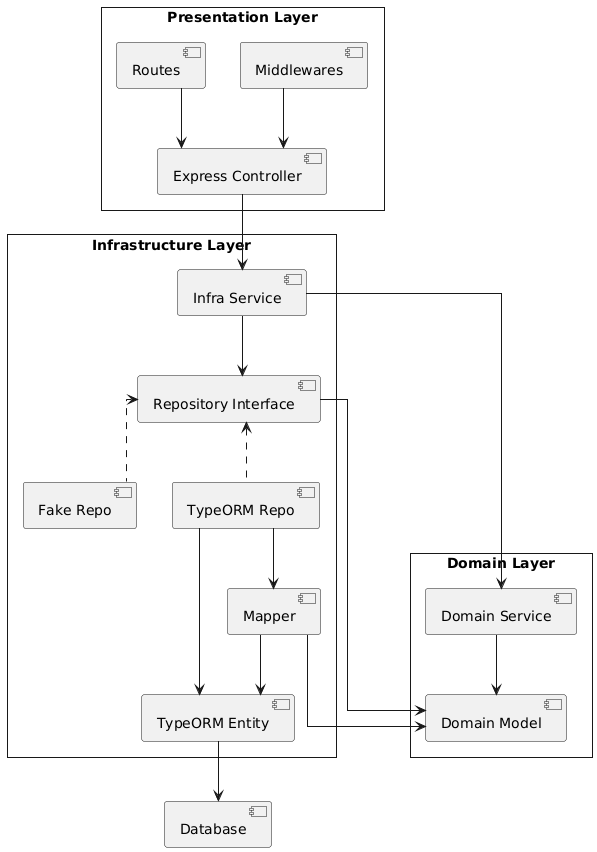
В проекте выделены следующие слои:

* Domain Layer (application/domain/):
  + Содержит бизнес-модели (domain) и бизнес-правила (services)
  + Это ядро приложения, которое по Clean Architecture не зависит от других слоев.
* Infrastructure Layer (Доступ к данным)
  + Содержит реализацию репозиториев и инфраструктурных сервисов (сервисов, которые связывают бизнес-правила с репозиторием)
  + Включает:
    - Services - инфраструктурные сервисы (взаимодействуют с репозиторием)
    - Repositories: Работа с базой данных
      * Эти репозитории отличаются от репозиториев TypeORM. Они представляют собой репозиторий для бизнес-моделей.
      * Interfaces: Определение контрактов для репозиториев.
      * Fakes: “поддельные репозитории” для быстрых тестов (без реальной записи в БД)
      * TypeORM: Репозитории с использованием TypeORM
        + Models: Модели (сущности ORM)

Mappers: мапперы для отображения бизнес-сущностей в сущности ORM

* Presentation Layer (API)
  + Express JS
    - Controllers: Обработка HTTP-запросов (взаимодействуют с инфраструктурными сервисами)
    - Routes: Определение маршрутов API
    - Middlewares: Промежуточные обработчики (auth)

Взаимодействие слоев между собой представлено на рис. 1

Рис. 1. Взаимодействие слоев в бойлерплейте

В файле [bootstrap.py](http://bootstrap.py) происходит проброс зависимостей в инфраструктурные сервисы: им пробрасывается репозиторий TypeORM. При желании, в любой момент можно написать любой другой репозиторий, который будет хранить модели другим способом. При этом, нужно будет лишь изменить код в 1 месте - [bootstrap.py](http://bootstrap.py). Такой способ уже выстрое с “поддельными” репозиториями.

**Вывод**

Написанный бойлерплейт представляет собой структурированное backend-приложение на базе Express + TypeORM + TypeScript, организованное с учетом принципов Clean Architecture и Domain-Driven Design (DDD).

Главным результатом работы является реализация четкого разделения ответственности между слоями.

Каждый слой зависит только от слоев, расположенных "ниже", что делает архитектуру устойчивой к изменениям и легко расширяемой. Использование интерфейсов репозиториев и мапперов позволяет свободно подменять реализацию (например, на фейковые репозитории в тестах), не затрагивая остальной код.

В результате, данное приложение легко масштабируется, тестируется и сопровождается.

* Модели (TypeORM) расположены в src/infrastructure/repositories/typeorm/models
* Контроллеры расположены в src/presentation/expressjs/controllers
* Роуты расположены в src/presentation/expressjs/routes