МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

**Лабораторна робота № 3**

**з курсу «Сучасні технології розробки WEB-застосувань на платформі Microsoft.NET»**

Викладач: Виконав:  
Бардін В. студент 3 курсу  
 групи ІП-15 ФІОТ

Плугатирьов Д.В.

Київ – 2023

**Тема:** Проектування REST веб-API

**Мета лабораторної роботи:** ознайомитися з основами створення REST веб-API та методологією C4 для відображення архітектури системи. Ознайомитися з основами створення ER-діаграм для представлення структури бази даних.

**Завдання:**

1. З дотриманням вимог REST-у спроектувати веб-API для обраної (згідно варіанту) доменної області, використовуючи методологію C4 для створення діаграми архітектури системи;
2. Створити ER-діаграму для DAL (Data Access Layer), яка відображатиме структуру бази даних веб-API;
3. Оформити спроєктоване рішення у вигляді звіту до лабораторної роботи.

|  |  |  |
| --- | --- | --- |
| **Варіант** | **Предметна галузь** | **Функціональні вимоги** |
| 1 | Ресторан. Формування замовлень | 1. Страви складаються із інгредієнтів. Інгредієнти можуть складати різні страви. Страви складають прайсліст, в якому вказано ціну для різних порцій страви.  2. Замовлення може містити в собі набір декількох порцій різних страв.  Функціональні вимоги:  1. Складання страв та меню;  2. Формування замовлень. |

**Документація:** підготувати документацію(звіт до ЛР), яка включатиме опис веб-API, а також структуру бази даних з урахуванням ER-діаграми.

# **С4 діаграма**

**Рівень 1**

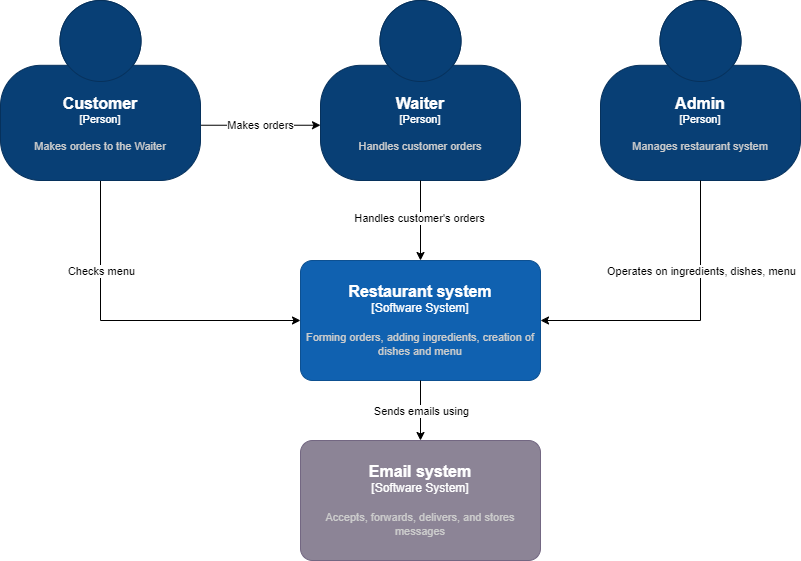


Рисунок 1 – System Context

**Рівень 2**

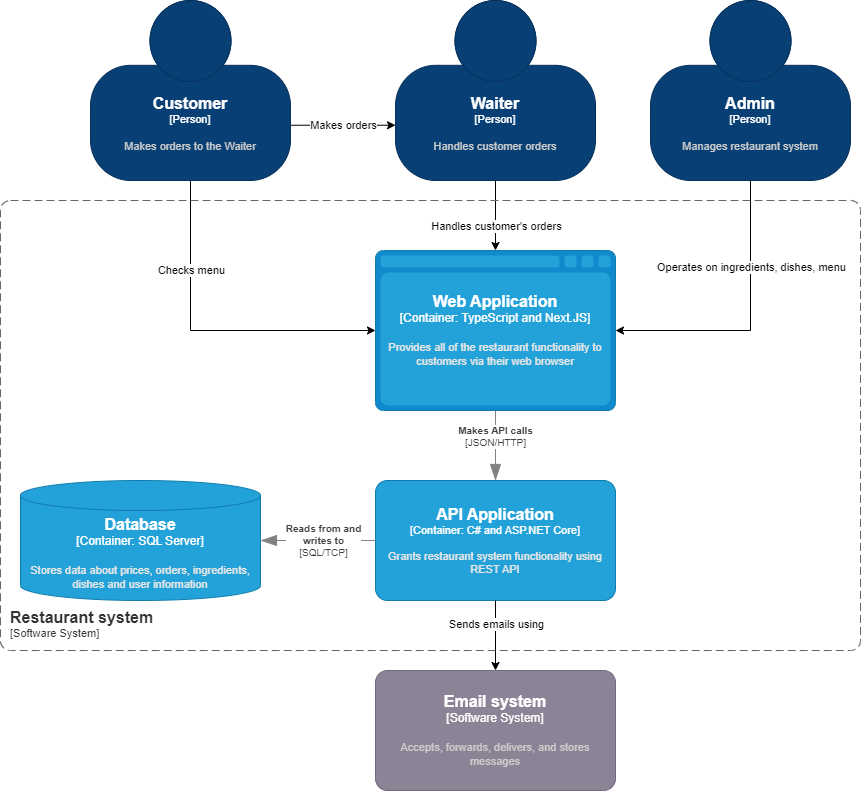


Рисунок 2 – Restaurant system Container

**Рівень 3**

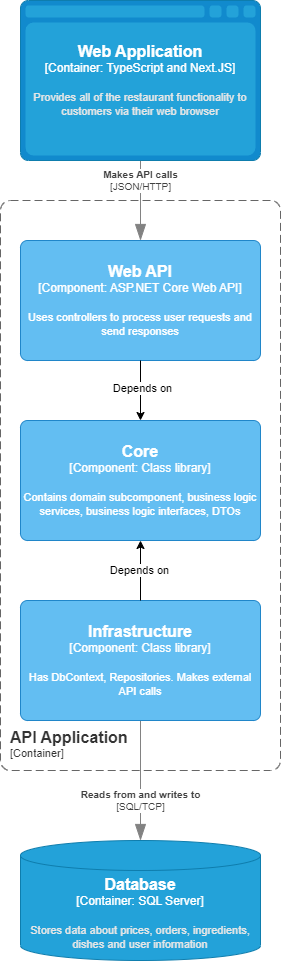


Рисунок 3 – API application Container

Для розробки API програми було використано шаблон Clean architecture, котрий передбачає існування наступних її рівнів:

1. **Presentation (Web API)** – приймає запити з браузера користувача та надсилає відповіді з сервера;
2. **Core** – бізнес-логіка застосунку та DTOs:
   1. **Domain** – класи сутностей та контракти репозиторіїв;
3. **Infrastructure** – контекст БД, репозиторії виклики до зовнішніх API.

**Рівень 4**

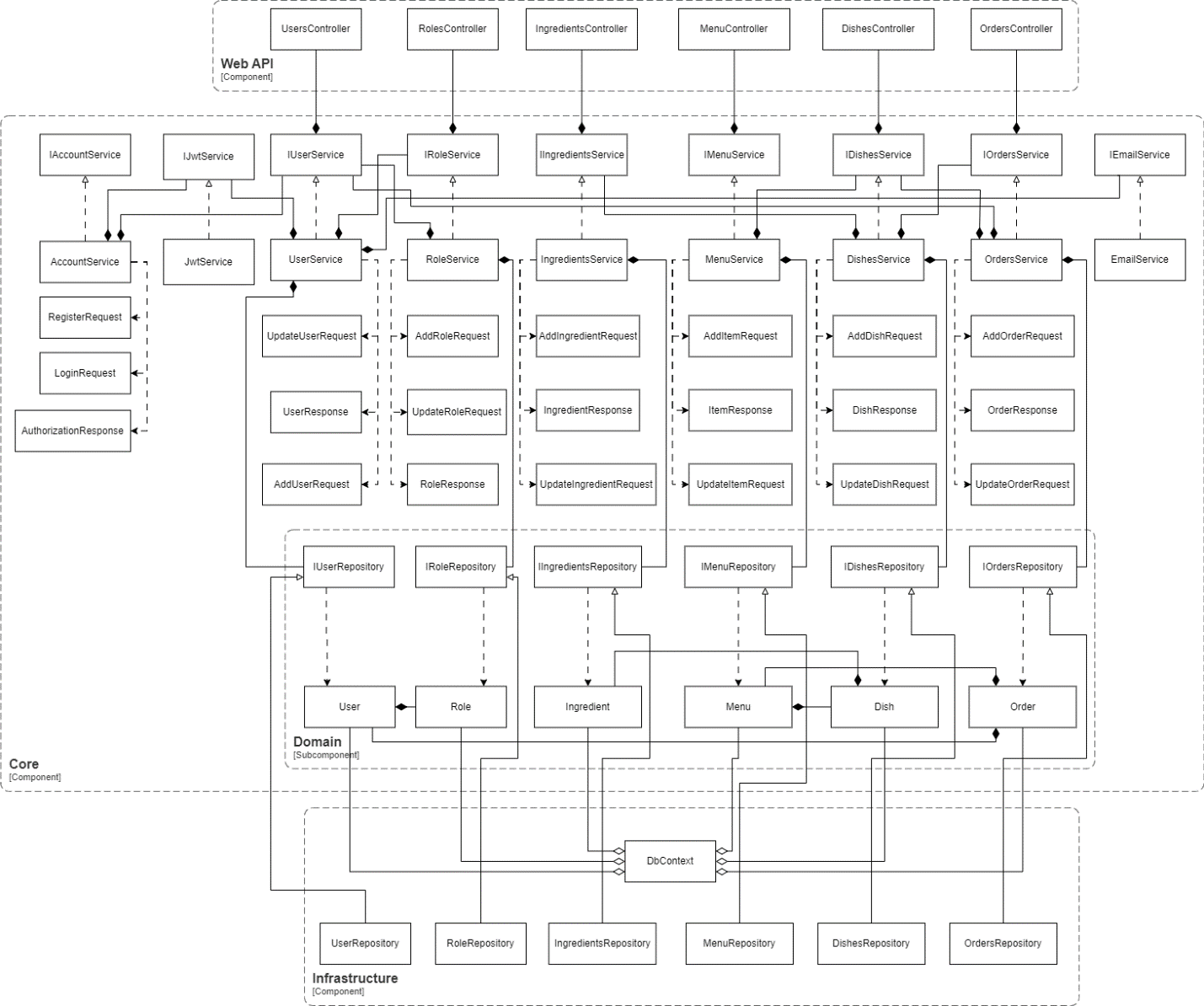


Рисунок 4 – API application Code

# **REST API Endpoints**

**Users**

* **GET**  
  **URL:** /api/v1/users  
  **Parameters:** pagination  
  **Body:** -  
  **Response:** UserResponse[]
* **GET  
  URL:** /api/v1/users/{id}  
  **Parameters:** UserId  
  **Body:** -  
  **Response:** UserResponse
* **POST  
  URL:** /api/v1/users  
  **Parameters:** -  
  **Body:** AddUserRequest  
  **Response:** UserResponse
* **DELETE  
  URL:** /api/v1/users/{id}  
  **Parameters:** UserId  
  **Body:** -  
  **Response:** -
* **PATCH  
  URL:** /api/v1/users  
  **Parameters:** -  
  **Body:** UpdateUserRequest  
  **Response:** UserResponse

**Roles**

* **GET  
  URL:** /api/v1/roles  
  **Parameters:** -  
  **Body:** -  
  **Response:** RoleResponse[]
* **GET  
  URL:** /api/v1/roles/{id}  
  **Parameters:** RoleId  
  **Body:** -  
  **Response:** RoleResponse
* **POST  
  URL:** /api/v1/roles  
  **Parameters:** -  
  **Body:** AddRoleRequest  
  **Response:** RoleResponse
* **DELETE  
  URL:** /api/v1/roles/{id}  
  **Parameters:** RoleId  
  **Body:** -  
  **Response:** -
* **PATCH  
  URL:** /api/v1/roles  
  **Parameters:** -  
  **Body:** UpdateRoleRequest  
  **Response:** RoleResponse

**Account**

* **POST  
  URL:** /api/v1/login  
  **Parameters:** -  
  **Body:** LoginRequest  
  **Response:** AuthorizationResponse
* **POST  
  URL:** /api/v1/signin  
  **Parameters:** -  
  **Body:** RegisterRequest  
  **Response:** AuthorizationResponse

**Ingredients**

* **GET  
  URL:** /api/v1/ingredients  
  **Parameters:** pagination  
  **Body:** -  
  **Response:** IngredientResponse[]
* **GET  
  URL:** /api/v1/ingredients/{id}  
  **Parameters:** IngredientId  
  **Body:** -  
  **Response:** IngredientResponse
* **POST  
  URL:** /api/v1/ingredients  
  **Parameters:** -  
  **Body:** AddIngredientRequest  
  **Response:** IngredientResponse
* **DELETE  
  URL:** /api/v1/ingredients/{id}  
  **Parameters:** IngredientId  
  **Body:** -  
  **Response:** -
* **PATCH  
  URL:** /api/v1/ingredients  
  **Parameters:** -  
  **Body:** UpdateIngredientRequest  
  **Response:** IngredientResponse

**Menu**

* **GET  
  URL:** /api/v1/menu  
  **Parameters:** pagination  
  **Body:** -  
  **Response:** ItemResponse[]
* **GET  
  URL:** /api/v1/menu/{id}  
  **Parameters:** ItemId  
  **Body:** -  
  **Response:** ItemResponse
* **POST  
  URL:** /api/v1/menu  
  **Parameters:** -  
  **Body:** AddItemRequest  
  **Response:** ItemResponse
* **DELETE  
  URL:** /api/v1/menu/{id}  
  **Parameters:** ItemId  
  **Body:** -  
  **Response:** -
* **PATCH  
  URL:** /api/v1/menu  
  **Parameters:** -  
  **Body:** UpdateItemRequest  
  **Response:** ItemResponse

**Dish**

* **GET  
  URL:** /api/v1/dishes  
  **Parameters:** pagination  
  **Body:** -  
  **Response:** DishResponse[]
* **GET  
  URL:** /api/v1/dishes/{id}  
  **Parameters:** DishId  
  **Body:** -  
  **Response:** DishResponse
* **POST  
  URL:** /api/v1/dishes  
  **Parameters:** -  
  **Body:** AddDishRequest  
  **Response:** DishResponse
* **DELETE  
  URL:** /api/v1/dishes/{id}  
  **Parameters:** DishId  
  **Body:** -  
  **Response:** -
* **PATCH  
  URL:** /api/v1/dishes  
  **Parameters:** -  
  **Body:** UpdateDishRequest  
  **Response:** DishResponse

**Order**

* **GET  
  URL:** /api/v1/orders  
  **Parameters:** pagination  
  **Body:** -  
  **Response:** OrderResponse[]
* **GET  
  URL:** /api/v1/orders/{id}  
  **Parameters:** OrderId  
  **Body:** -  
  **Response:** OrderResponse
* **POST  
  URL:** /api/v1/orders  
  **Parameters:** -  
  **Body:** AddOrderRequest  
  **Response:** OrderResponse
* **DELETE  
  URL:** /api/v1/orders/{id}  
  **Parameters:** OrderId  
  **Body:** -  
  **Response:** -
* **PATCH  
  URL:** /api/v1/orders  
  **Parameters:** -  
  **Body:** UpdateOrderRequest  
  **Response:** OrderResponse

# **ER-діаграма**

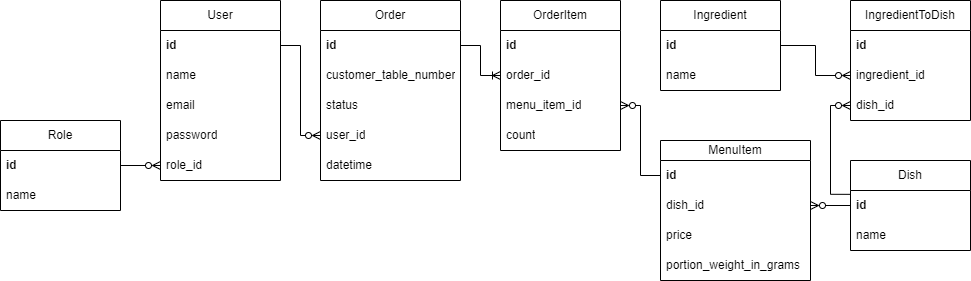


Рисунок 5 – ER-діаграма БД системи ресторану

**Tables**

**Role**

* id (PK): INT
* name: NVARCHAR(30)

**User**

* id (PK): INT
* name: NVARCHAR(255)
* email: NVARCHAR(255)
* password: NVARCHAR(255)
* role\_id (FK): INT

**Order**

* id (PK): INT
* customer\_table\_number: INT
* status: NVARCHAR(255)
* user\_id (FK): INT
* datetime: DATETIME

**OrderItem**

* id (PK): INT
* order\_id (FK): INT
* menu\_item\_id (FK): INT
* count: INT

**Ingredient**

* id (PK): INT
* name: NVARCHAR(50)

**MenuItem**

* id (PK): INT
* dish\_id (FK): INT
* price: MONEY
* portion\_weight\_in\_grams: INT

**Dish**

* id (PK): INT
* name: NVARCHAR(50)

**IngredientToDish**

* id (PK): INT
* ingredient\_id (FK): INT
* dish\_id (FK): INT

**Висновок**

Під час виконання цієї лабораторної роботи я ознайомився з архітектурним шаблоном Clean architecture, використав методологію C4 для розробки діаграми архітектурної системи. Замість Data Access Layer мною було зображено Infrastructure layer, котрий має посилання на Core layer та відповідає за взаємодію з БД і ER-діаграма БД системи з описом її елементів. Також мною був спроєктований REST API.