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

Практикум №3

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

на тему: «Проектування REST веб-API»

Викладач:
Бардін В.

Виконав:
Хільчук А.В.
студент 3 курсу
групи ІП-14 ФІОТ

Київ-2023

Практична робота №3

Тема: Проектування REST веб-API
Завдання:

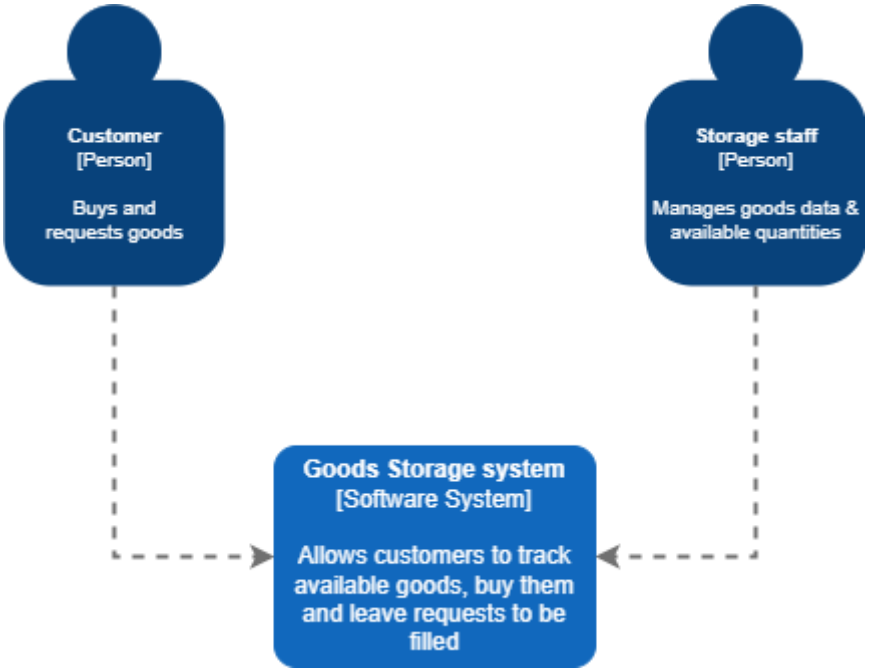
1. З дотриманням вимог REST-у спроектувати веб-API для обраної(згідно варіанту) доменної області, використовуючи методологію C4 для створення діаграми архітектури системи.
2. Створити ER-діаграму для DAL (Data Access Layer), яка відображатиме структуру бази даних веб-API.
3. Оформити спроєктоване рішення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

5	Склад. Облік товарів	1. На складі зберігаються товари, які можуть бути відпущені замовнику зі складу. 2. Товари можливо заказати до продажу навіть при відсутності його на складі. При відсутності товару потрібно записати його у чергу на придбання та завезення на склад. Після доставки відсутніх товарів на склад відвантаження за замовленням може бути виконано. **Функціональні вимоги:** 1. Облік товарів на складі; 2. Реалізація товарів зі складу.

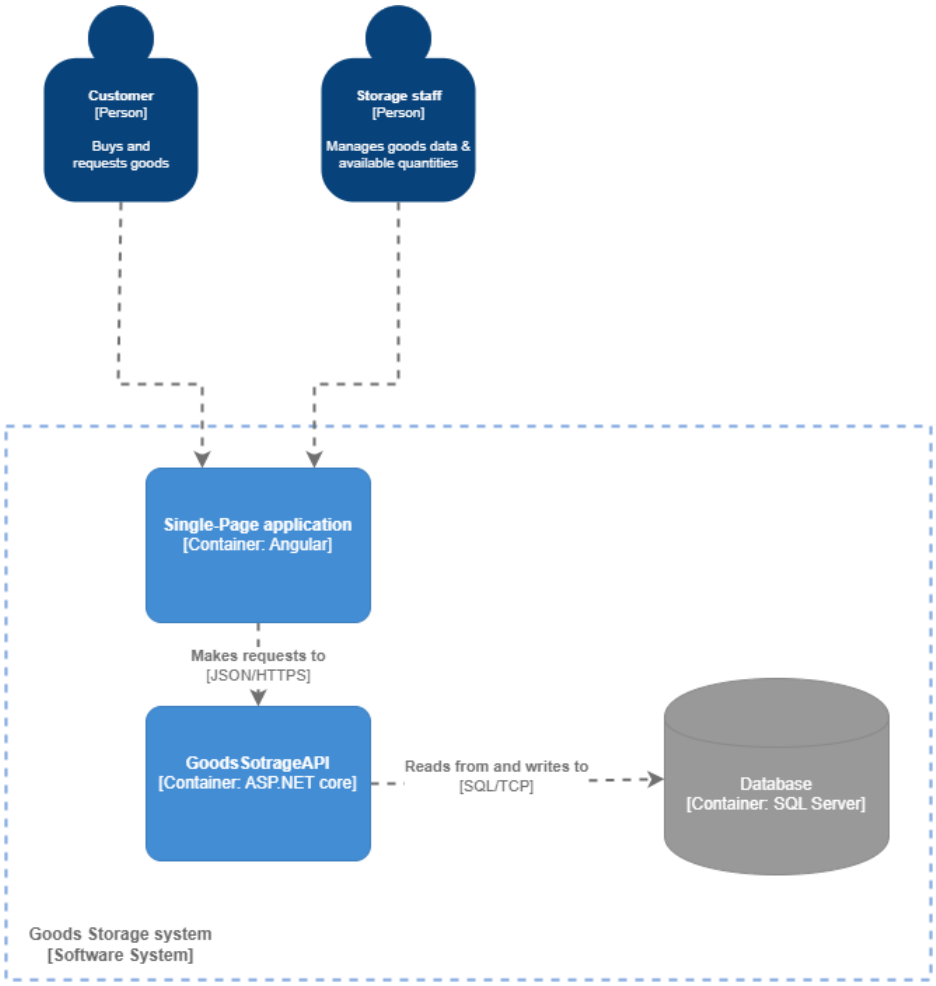
Виконання:

1. З дотриманням вимог REST-у спроектувати веб-API для обраної(згідно варіанту) доменної області, використовуючи методологію C4 для створення діаграми архітектури системи.

Отож, із завдання випливає, що цільовими типами користувачів застсосунку будуть замовники та працівники складу. Інтеграції із зовнішніми системами немає, тому контекстний рівень C4 матиме наступний вигляд:

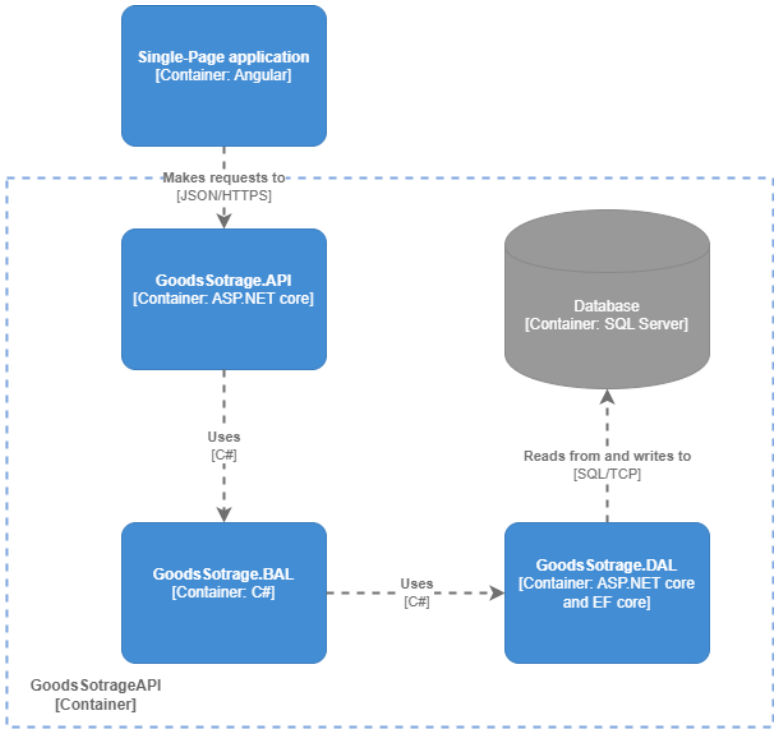


Заглилблюємося в систему. Зрозуміло, вона буде складатися з односторінкового додатку або іншого клієнту, що надсилатиме запити, а також власне API:

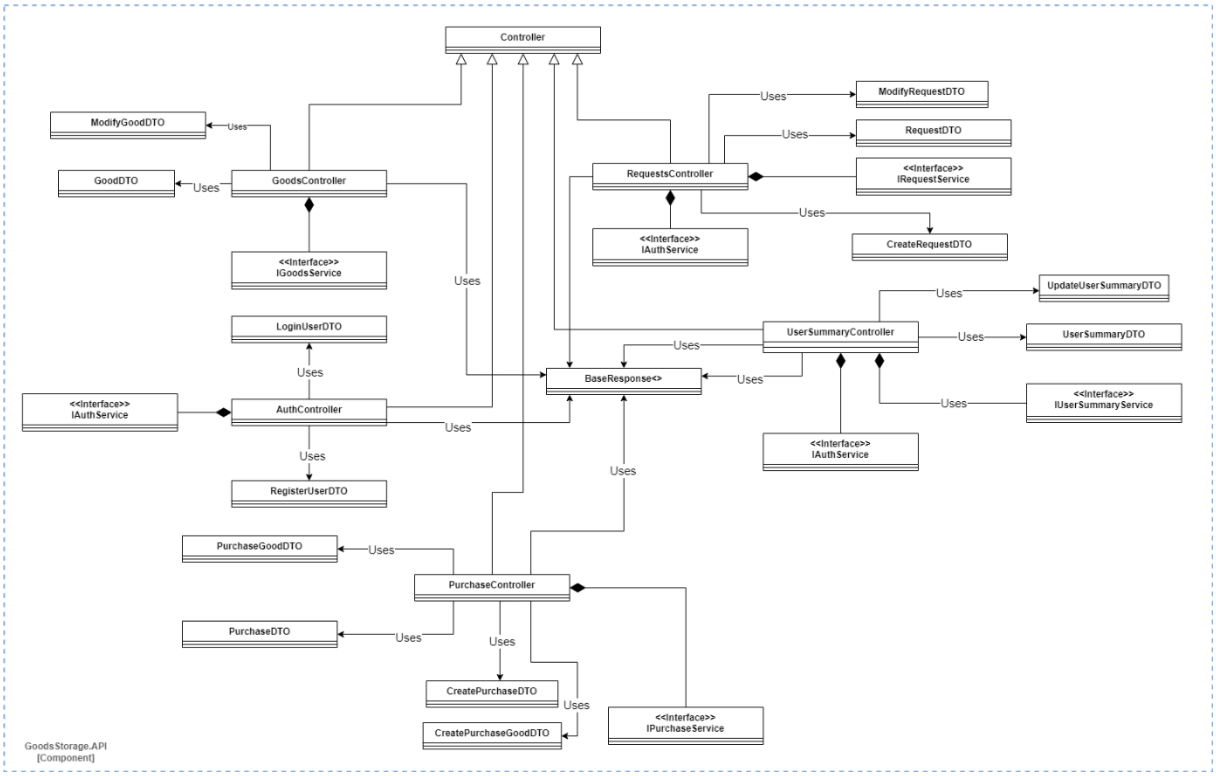


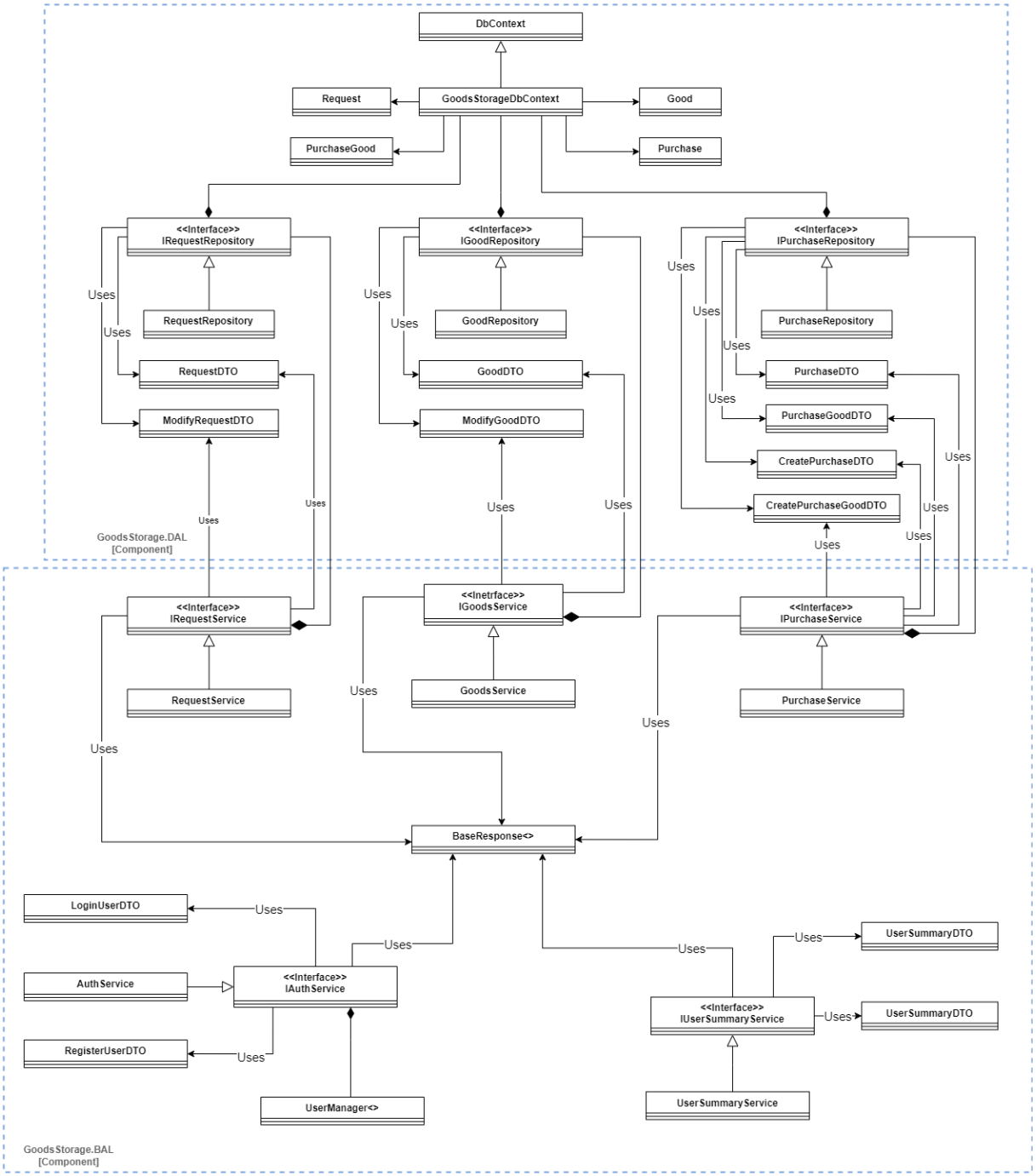
Заглиблюємося в GoodsStorageApi. Даний контейнер складається з бази даних та 3 компонентів, що представляють собою шари додатку:

1. Шар презентації – містить у собі контроллери, що надсилатимуть відповіді на запити
2. Шар бізнес логіки – містить у собі сервіси, що проводять необхідні комплексні обчислення та операції
3. Шар доступу до даних – відповідальний за доступ до даних, а також конвертацію моделей зберігання в бізнес моделі



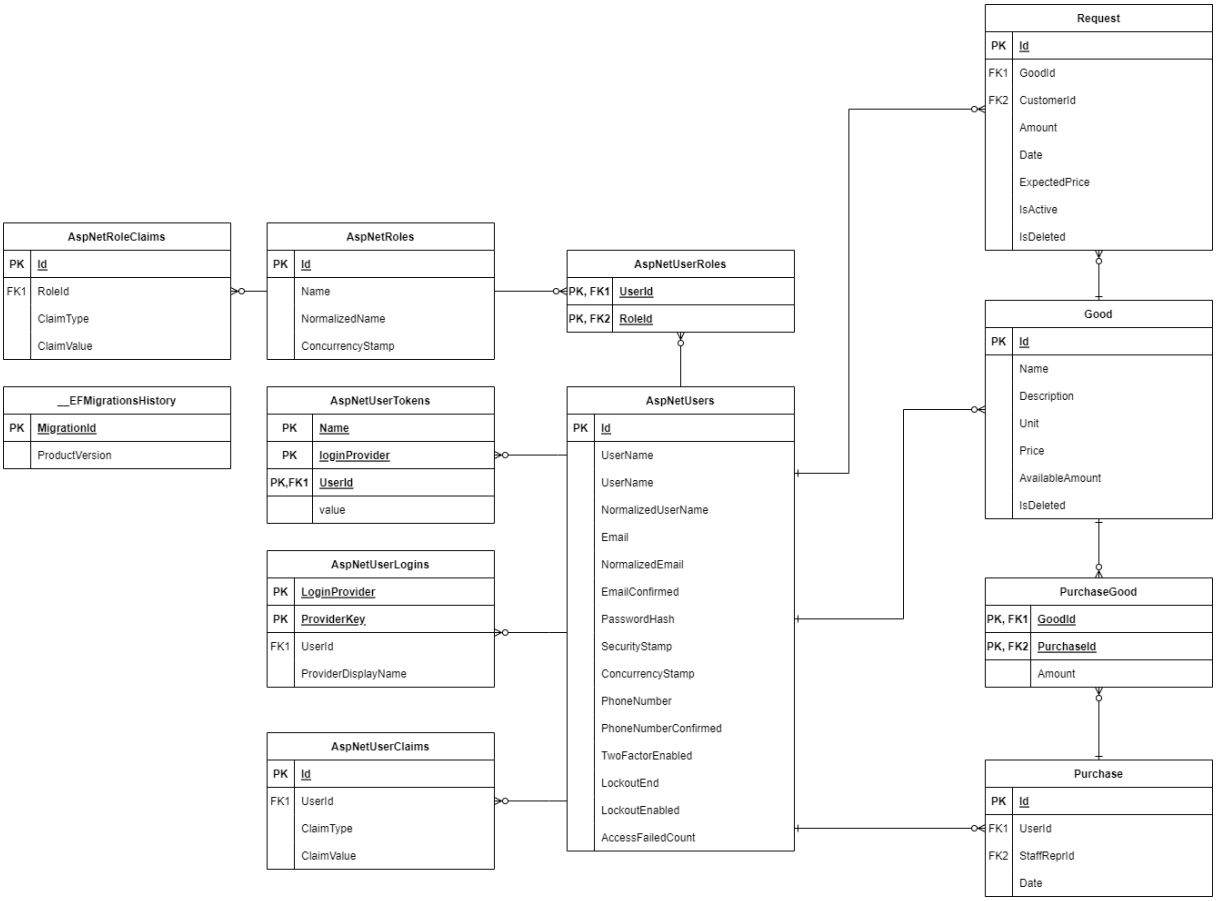
Тоді розширюємо відповідні компоненти до їхніх складових класів:


Controller
ModifyGoodDTO
Uses
GoodDTO
Uses
GoodsController
<<Interface>>
IGoodsService
Uses
LoginUserDTO
Uses
<<Interface>>
IAuthService
AuthController
Uses
Uses
RegisterUserDTO
Uses
ModifyRequestDTO
Uses
RequestDTO
RequestsController
<<Interface>>
IRequestService
Uses
<<Interface>>
IAuthService
CreateRequestDTO
Uses
UpdateUserSummaryDTO
UserSummaryController
Uses
UserSummaryDTO
Uses
BaseResponse<>
Uses
<<Interface>>
IUserSummaryService
<<Interface>>
IAuthService
Uses
PurchaseGoodDTO
Uses
PurchaseController
PurchaseDTO
Uses
Uses
CreatePurchaseDTO
Uses
CreatePurchaseGoodDTO
<<Interface>>
IPurchaseService
GoodsStorage.API
[Component]



2. Створити ER-діаграму для DAL (Data Access Layer), яка відображатиме структуру бази даних веб-API.

Результуюча схема БД має наступний вигляд:



Частина сутностей – AspNetUsers, AspNetUserRoles, AspNetRoleClaims, AspNetUSerTokens, AspNetUserLogins, AspNetUserClaims – будуть створені автоматично бібліотекою Identity Core й будуть пов’язані з авторизацією та автентифікацією користувачів. Решта ж сутностей відповідають моделям доменної області:

1. Request – відповідає запиту на замовлення конкретного виду товару
2. Good – відповідає товару
3. Purchase – відповідає запису купівлі товару користувачем
4. PurchaseGood – сполучна таблиця між покупкою та товаром й містить кількість купленого товару

Опис кінцевих точок додатку:

1. GET api/goods?pageNumber=[int]&pageSize=[int]
    - Отримує всі товари
    - pageNumber та pageSize задаватимуть пагінацію
2. GET api/goods/{id:guid}
    - Повертає значення товару за ідентифікатором
3. DELETE api/goods/{id:guid}
    - Проводить soft delete товару за ідентифікатором

4. POST api/goods
    - Створює товар
5. PUT api/goods/{id:guid}
    - Оновлює товар за ідентифікатором
6. GET api/purchases?pageNumber=[int]&pageSize=[int]&userId=[guid]
    - Отримує всі покупки
    - pageNumber та pageSize задаватимуть пагінацію
    - userId задається для вибору покупок конкретного покупця
7. GET api/purchases/{id:guid}
    - Отримує покупку за ідентифікатором
8. POST api/purchases
    - Створює нову покупку
9. POST api/auth/register
    - Реєструє користувача
10. POST api/auth/login
    - Логінить користувача
11. GET api/requests?pageNumber=[int]&pageSize=[int]&userId=[guid]&activeOnly=[bool]
    - Отримує всі запити
    - pageNumber та pageSize задаватимуть пагінацію
    - userId задається для вибору запитів конкретного покупця
    - activeOnly використовується, аби відсортувати лише незаповнені запити
12. GET api/requests/{id:guid}
    - Повертає значення запиту за ідентифікатором
13. DELETE api/requests/{id:guid}
    - Проводить soft delete запиту за ідентифікатором
14. POST api/requests
    - Створює запит
15. PUT api/requests/{id:guid}
    - Оновлює запит за ідентифікатором
16. GET api/usersummary/{id:guid}
    - Отримує інформацію про користувача за ідентифікатором
17. PUT api/usersummary/{id:guid}
    - Оновлює інформацію про користувача за ідентифікатором

Висновок

Отож, у ході виконання лабораторної роботи було спроектовано REST API з 3-рівневою архітектурою для обслуговування слкаду з товарами та створено на основі даної проекції діаграму C4. У рамках виконання даної діаграми було зафіксовано 2 контейнери – клієнт та API – 3 компоненти, що відповідають шарам додатку, 5 контролерів та сервісів, DbContext, три репозиторії, а також низку бізнес-моделей та моделей зберігання. Побудовано ER діаграму, що включає в себе як сутності, згенеровані автоматично Identity Provider’ом, так і сутності предметної області. Набуто практичних навичок проектування REST API та побудови C4 діаграм.