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

Звіт

з лабораторної роботи № 3 з дисципліни
«Сучасні технології розробки WEB-застосувань на платформі Microsoft.NET»

“Проектування REST веб-API”

Виконав(ла): Маланічев Д., ІС-13

Перевірив(ла): Бардін В.

Київ 2023

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:

Практична частина:

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.

Документація:

1. Підготувати документацію(звіт до ЛР), яка включатиме опис веб-API, а також структуру бази даних з урахуванням ER-діаграми.

Варіант завдання:

9	Реєстратура лікарні. Запис до лікарів на прийом	1. Реєстратура надає дані стосовно наявності лікарів та розкладу прийому хворих. 2. Хворим можливо записатись на прийом до лікаря, якщо є вільний час у розкладі лікаря. 3. В реєстратурі ведуться картки відвідування хворими лікарні, в які записується час відвідання лікаря, діагноз та лікар, що його поставив. **Функціональні вимоги:** 1. Ведення картотеки лікарні; 2. Керування прийомами хворих у лікарів.

Хід роботи

В ході виконання роботи потрібно створити діаграми за моделлю C4. А саме System context diagram, Container diagram, Component diagram та Code diagram.

Розпочнемо з **System context diagram:**

System context diagram показує систему здалеку, як систему як коробку в центрі, оточену користувачами та іншими системами, з якими вона взаємодіє. В даній системі доцільно буде виділити двох користувачів: пацієнта та лікаря. Пацієнт може отримати дані про наявних лікарів та години їх прийому. Відповідно записатися на прийом до обраного. Лікар, перед прийомом, може отримати дані пацієнта, який прийде до нього на прийом. Сама система зберігає ці дані та картки відвідування хворими лікарні, в які записується час відвідання лікаря, діагноз та лікар, що його поставив. Цю взаємодію можна побачити на діаграмі контексту:

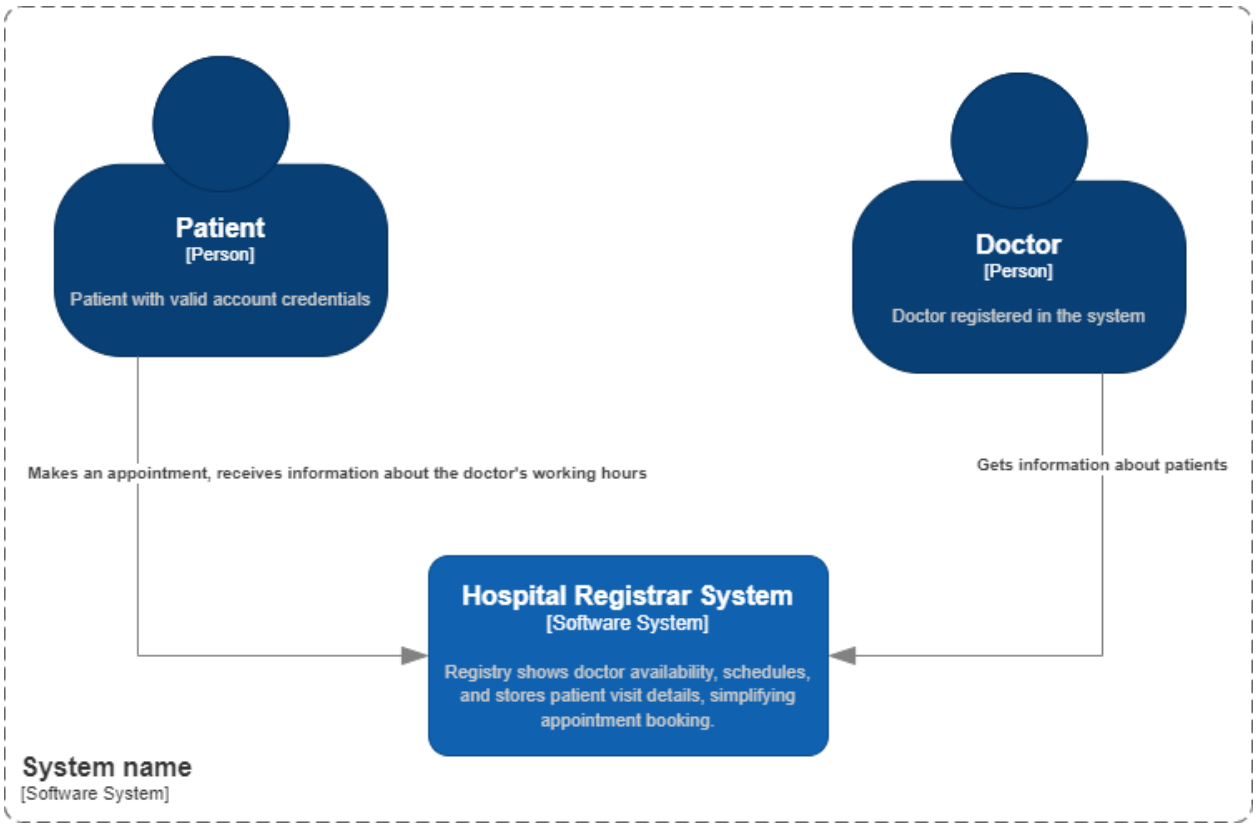

Рис. 1 – System Context Diagram

Перейдемо до **Container Diagram:**

Діаграма контейнерів показує високорівневу структуру архітектури програмного забезпечення та розподіл обов'язків між її компонентами. Вона також показує основні технології і те, як контейнери взаємодіють один з одним.

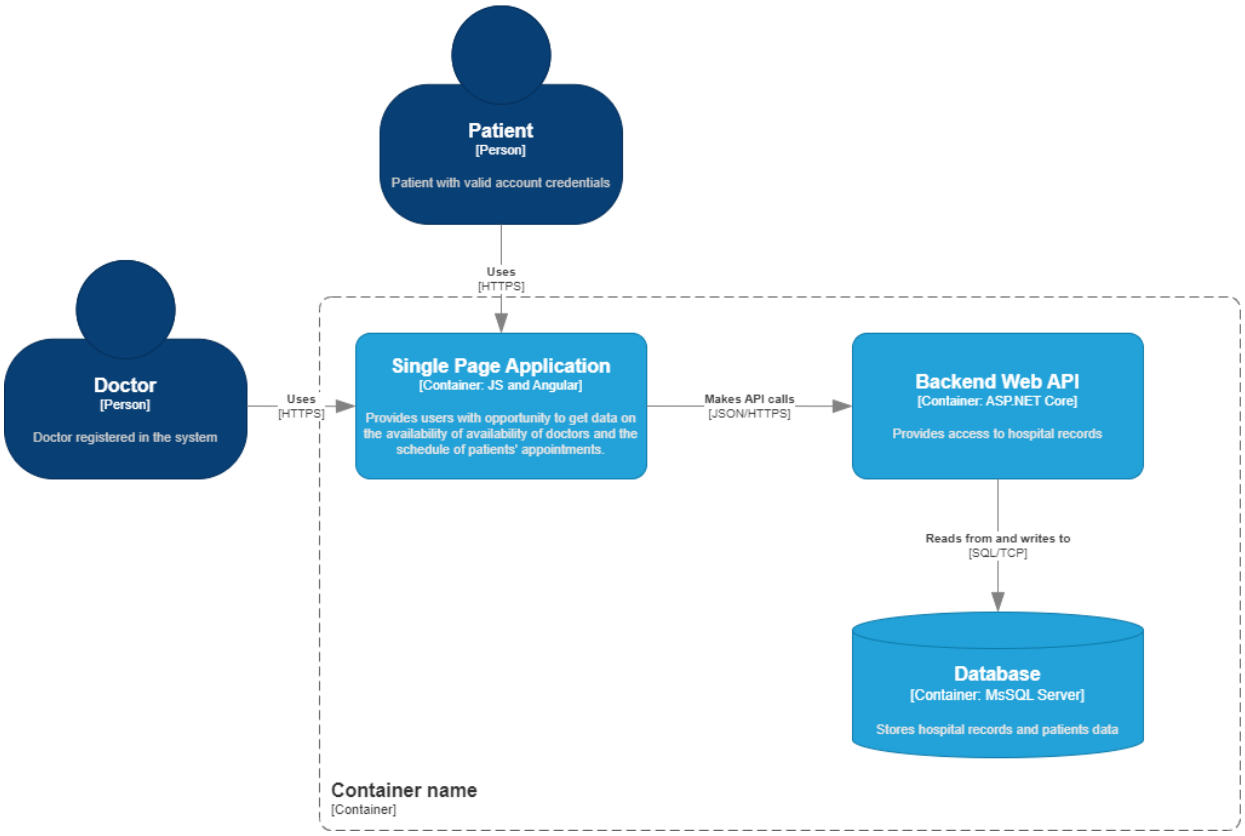

Рис. 2 – Container Diagram

Component Diagram:

Діаграма компонентів показує, як контейнер складається з низки «менших» компонентів, їхні обов'язки та деталі технічної реалізації.

1. В даному випадку розглянуто діаграму Web API, що спроєктовано за N-tier architecture. Компонент складається з трьох рівнів:
2. API Layer, що отримує запити з Frontend оброблю їх та повертає відповідь;
3. Business Logic Layer – містить в собі, відповідно, бізнес логіку;
4. Access Layer – за допомогою EF Core отримую доступ до БД.

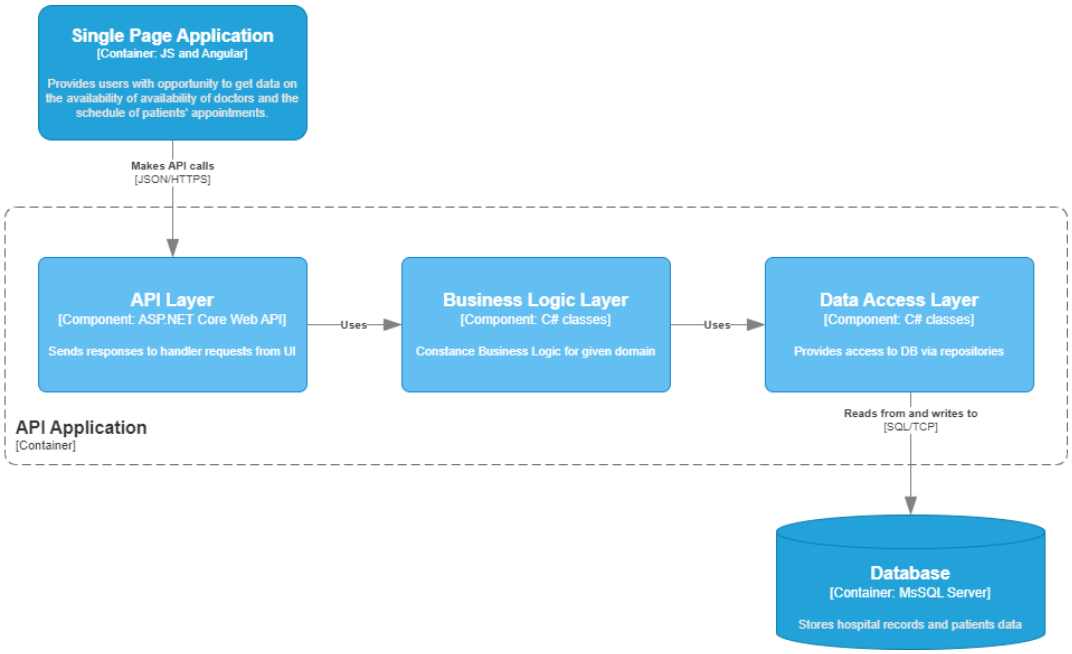

Рис. 3 – Component Diagram

Останнім компонентом C4 є Code Diagram. Вона збільшує масштаб кожного компонента, щоб показати, як він реалізований у вигляді коду. В нашому випадку, маємо N-Layer архітектуру. В Persistence є підрівень Domain, що описує сутності доменної області. Сам Persistence відповідає за взаємодію з базою даних. CRUD операціє тут винесені в GenericRepository, для запобігання дублювання коду. BL Layer використовує Persistence Layer та виконує, відповідно, бізнес логіку. В решті-решт, Presentation Layer виконує взаємодію через контролери, використовуючи сервіси з BL.

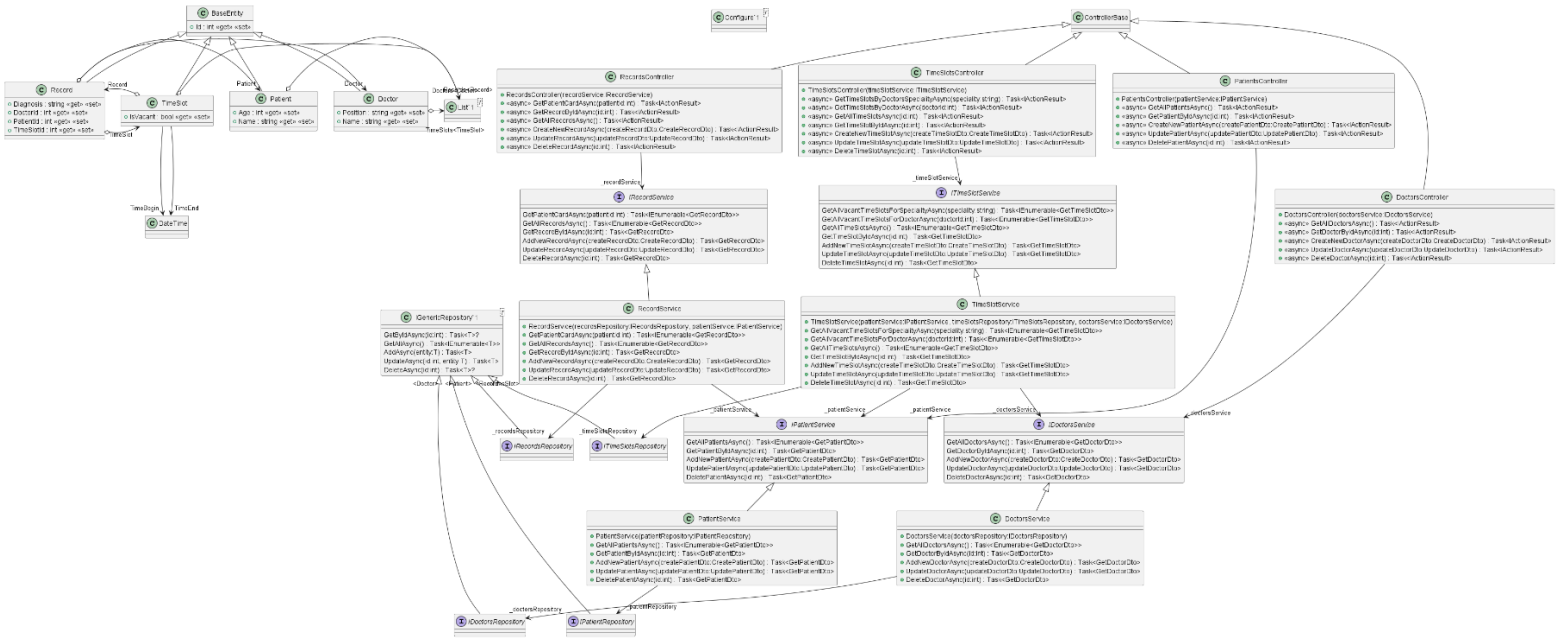

Рис. 4 – Code Diagram

Для бази даних було виокремлено наступні сутності: Лікар, Пацієнт, Часовий слот, Запис. Їх поля та зв’язки можна побачити на діаграмі:

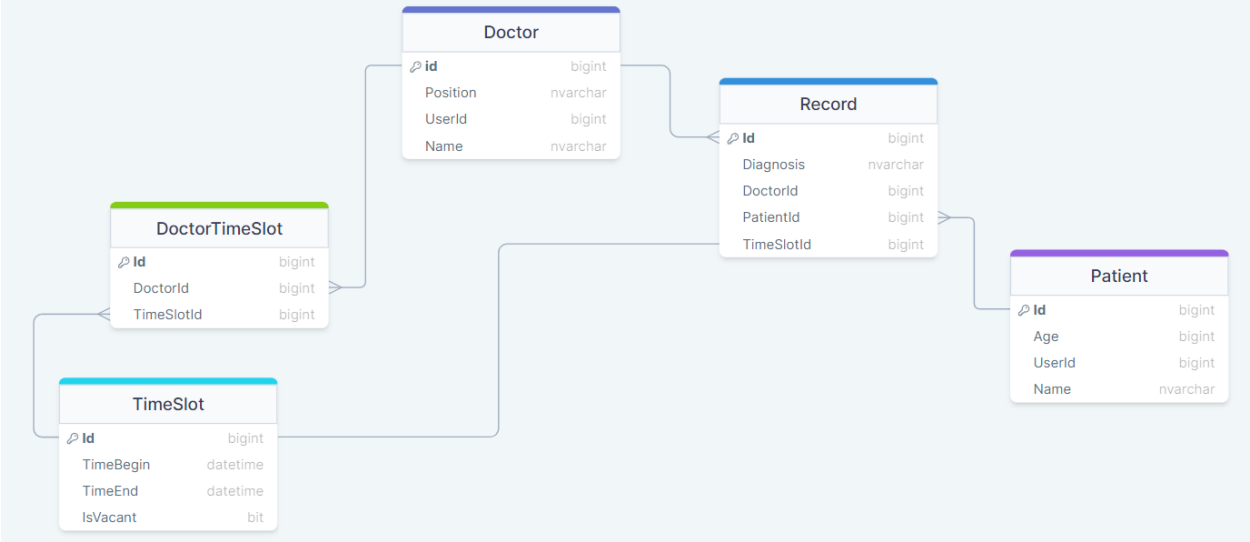

Рис. 5 – ER-діаграма

Контракти:

Рис. 6 – ендпоінти для Doctors, Patients та Records

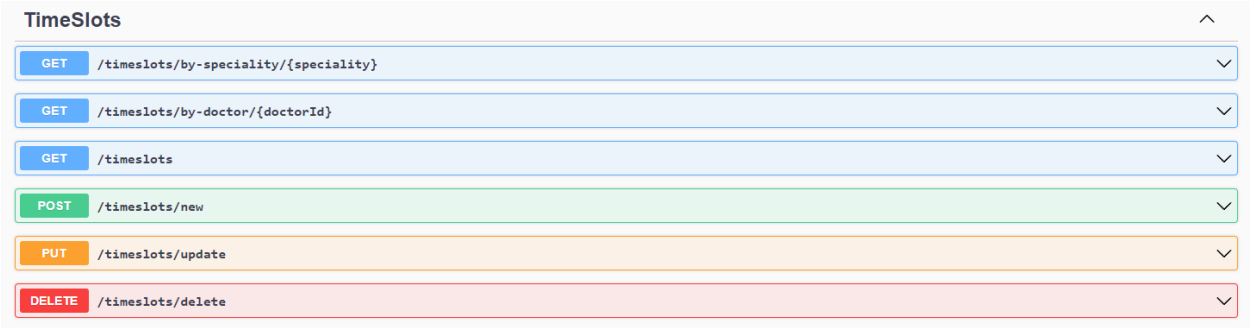

Рис. 7 – ендпоінти для TimeSlots

Окрім звичайних CRUD операцій тут варто відмітити наступні ендпоінти: /timeslots/by-speciality/{speciality}, /timeslots/by-doctor/{doctorId} та /card/{patientId}.

/timeslots/by-speciality/{speciality}

- повертає всі доступні таймслоти всіх лікарів, що спеціалізуються обраним профілем.

/timeslots/by-doctor/{doctorId}

- повертає всі доступні таймслоти обраного лікаря (вказаного його ID).

/card/{patientId}

- повертає всі записи вказаного пацієнта, що являє собою його картку.

Висновок:

Отже, в результаті виконання даної лабораторної роботи було створено набір діаграм за методологією C4, за заданою предметною областю, в даному випадку – Реєстратура лікарні. Запис до лікарів на прийом. Таким чином, було задокументовано 4 діаграми - System Context Diagram, Container Diagram, Component Diagram та Code Diagram. Окрім того додано ER-модель спроєктованої бази даних. За створеними діаграмами був написаний «шаблон» майбутнього API. Відповідні контракти додано в звіт.

Посилання на GitHub: https://github.com/DenysMalanichev/HospitalRegistrar