Мета:

1. Ознайомитися з основами створення REST веб-API та методологією C4 для відображення архітектури системи.
2. 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 3:

3	Театральна каса. Розповсюдження квитків	1. Афіша вистав надає функції пошуку потрібної вистави за автором, назвою, жанром та датою. 2. Кожній виставі відповідає кількість квитків різної вартості. 3. Квитки можна безпосередньо продати чи попередньо забронювати та згодом перевести їх у стан проданих. **Функціональні вимоги:** 1. Ведення афіши вистав; 2. Продаж квитків

ER-діаграма

Таблиця: **Genre**

Поля:

- IdGenre: GUID – унікальний ідентифікатор жанру
- NameGenre: varchar(20) – назва жанру

Таблиця: **Author**

Поля:

- IdAuthor: GUID – унікальний ідентифікатор автора
- FirstName: varchar(20) – ім’я автора
- MiddleName: varchar(20) – по батькові автора
- LastName: varchar(20) – прізвище автора

Таблиця: **Performance**

Поля:

- IdPerformance: GUID – унікальний ідентифікатор вистави
- Name: varchar(20) – назва вистави
- IdGenre: GUID – унікальний ідентифікатор жанру
- IdAuthor: GUID – унікальний ідентифікатор автора
- DateTimeEvent: datetime – дата та час вистави

Таблиця: **TypeOfTickets**

Поля:

- IdType: GUID – унікальний ідентифікатор типу квитків
- NameGenre: varchar(20) – назва типу квитків

Таблиця: **AllTickets**

Поля:

- IdAllTickets: GUID – унікальний ідентифікатор квитків
- IdType: GUID – унікальний ідентифікатор типу квитків
- IdPerformance: GUID – унікальний ідентифікатор вистави
- CountOfTickets: int – кількість квитків
- Price: int – ціна квитка

Таблиця: **StatusTicket**

Поля:

- IdStatus: GUID – унікальний ідентифікатор статусу квитка
- NameStatus: varchar(20) – назва статусу квитка

Таблиця: **Ticket**

Поля:

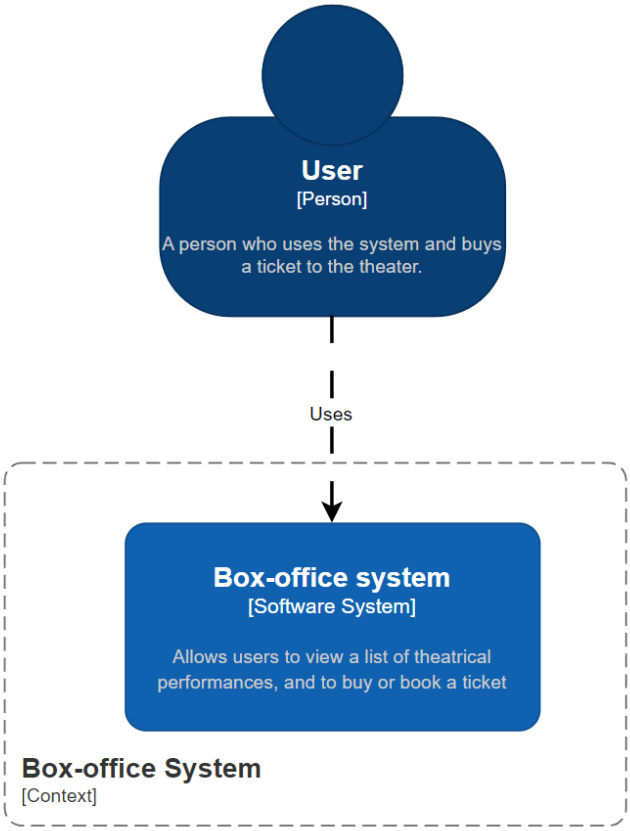
- IdTicket: GUID – унікальний ідентифікатор квитка
- IdAllTickets: GUID – унікальний ідентифікатор квитків
- FirstName: varchar(20) – ім’я покупця
- MiddleName: varchar(20) – по батькові покупця
- LastName: varchar(20) – прізвище покупця
- PhoneNumber: varchar(20) – номер телефону
- IdStatus: GUID - унікальний ідентифікатор статусу квитка
- SeatNumber: int – номер місця

Діаграма C4

C4 (Context, Container, Component, Code) - це методологія для моделювання архітектури програмного забезпечення на різних рівнях абстракції. Ця методологія дозволяє структурувати розгляд архітектури на різних рівнях деталей, починаючи від високорівневого контексту та закінчуючи кодом компонентів.

Рівень 1 : System Context Diagram

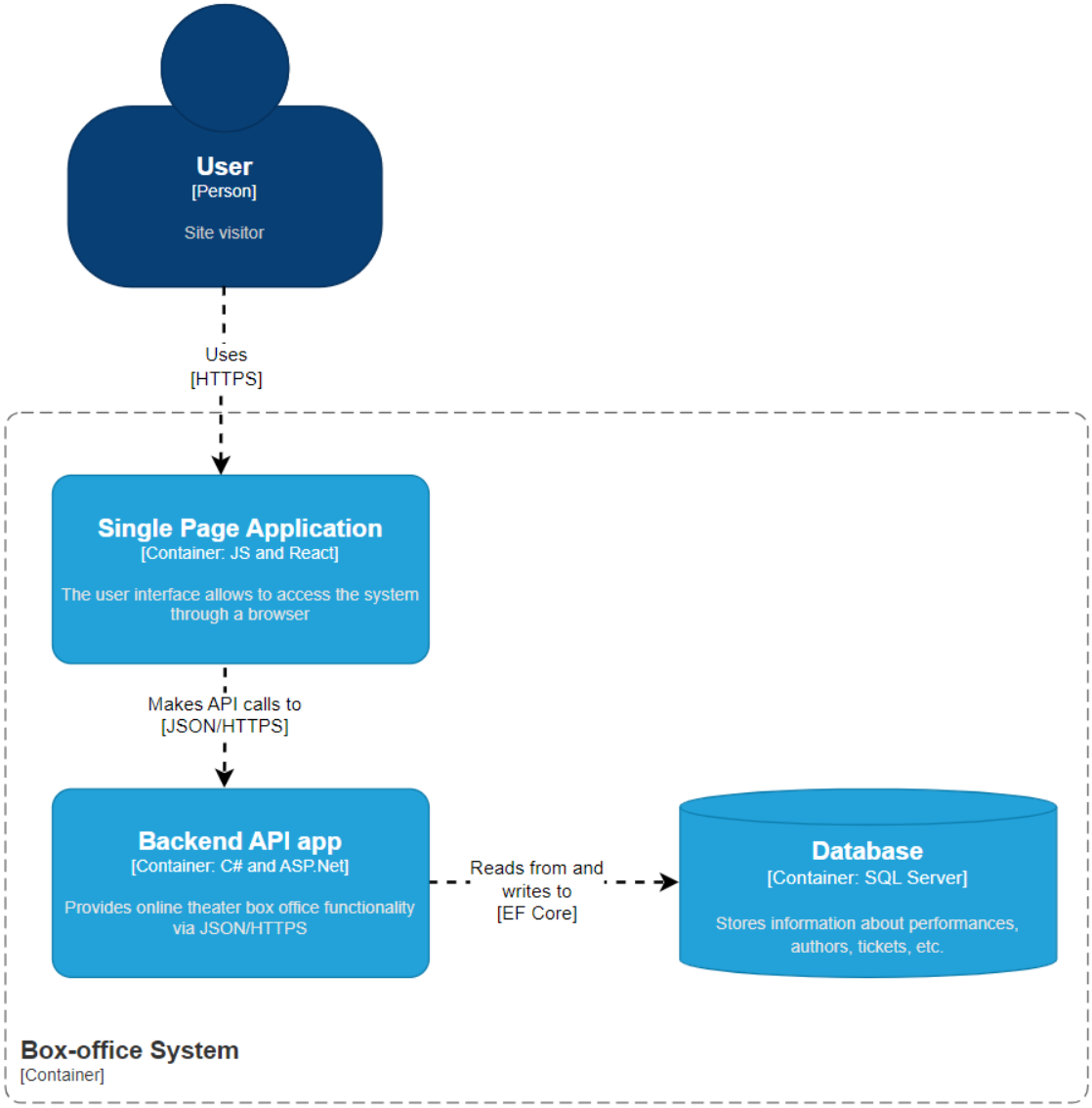
System Context Diagram (Контекстна діаграма системи) – Це перший рівень діаграм C4 і представляє високорівневий огляд системи та її зовнішнього середовища. Вона вказує на основні актори (людей або системи), які взаємодіють з системою, і визначає основні взаємодії між ними.



Система театральної каси має один тип користувача, який є неавторизованим. Користувач може переглядати театральну афішу, де є функції перегляду всіх вистав, пошуку вистав по назві, автору та даті. Система надає можливість, що дозволяє обрати категорію та номер місця, а також купувати та бронювати квитки, які згодом можна перевести у стан проданих.

Рівень 2 : Container Diagram

Container Diagram(Діаграма контейнерів): На цьому рівні архітектурного моделювання, система розглядається як набір контейнерів (наприклад, веб-додаток, база даних, мобільний додаток тощо). Це дозволяє деталізувати внутрішню структуру системи та показати взаємодії між контейнерами.

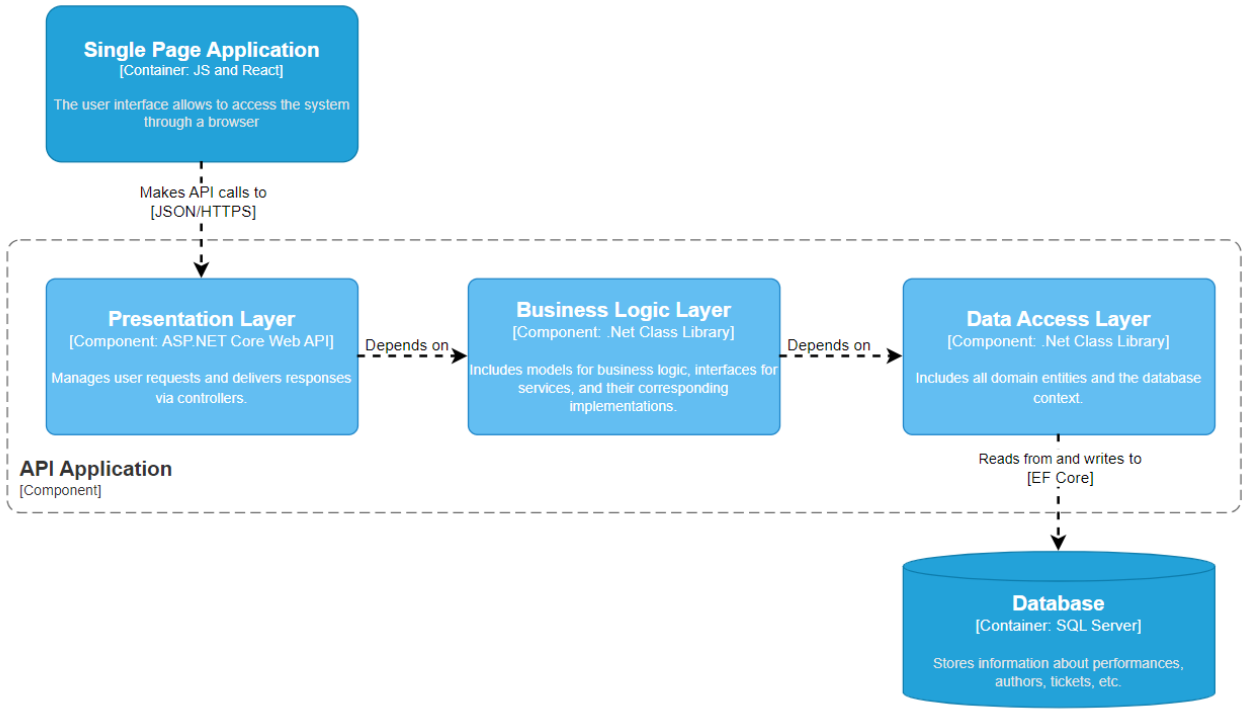


Система театральної каси поділяється на три компоненти:

1. Веб-застосунок, через який користувач взаємодіє з системою.
2. Серверний застосунок, де виконуються запити користувача. Функціонал реалізується через REST API.
3. База даних, яка зберігає всю необхідну інформацію, яка задіяна у системі.

Рівень 3 : Component Diagram

Component Diagram (Діаграма компонентів) - на цьому рівні система розглядається як набір компонентів, які визначають внутрішню реалізацію контейнерів. Компоненти в цьому контексті можуть бути реальними програмними елементами, такими як класи, бібліотеки, модулі або підсистеми.

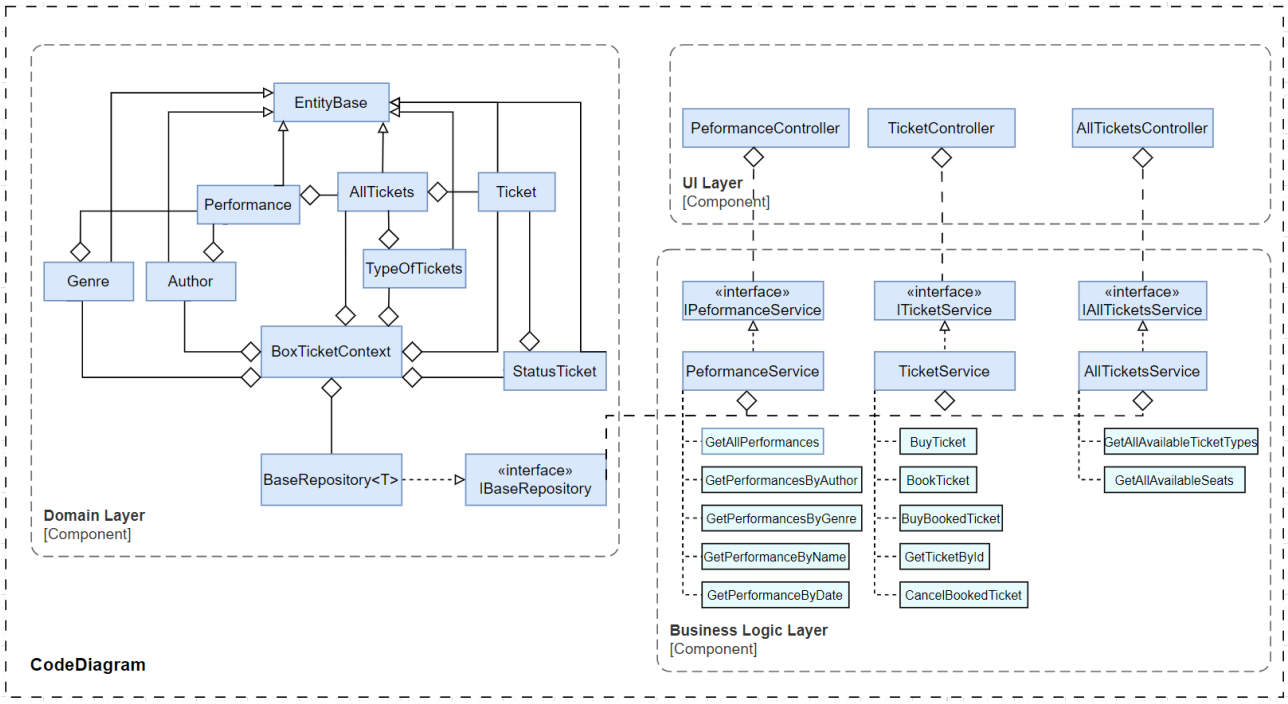


На етапі розробки застосунку буде використано N-layer архітектуру, а вона вже розділяє застосунок на три шара:

1. User Interface Layer (Шар представлення) – на цьому рівні відбувається взаємодія користувача, виконання запитів, та представлення необхідної інформації за допомогою графічного інтерфейсу. За приймання запитів відповідають контролери.
2. Business Logic Layer (Шар бізнес-логіки) – на цьому рівні відбувається обробка даних та реалізація логіки операцій і функцій.
3. Data Access Layer (Шар доступу до даних) – на цьому рівні відбувається взаємодія з базою.

Рівень 4 : Code Diagram

Code Diagram(Діаграма коду) - на останньому рівні представлення архітектури використовується діаграма коду. Вона може включати в себе високорівневі структури коду, залежності між класами або компонентами, та інші деталі реалізації.



DLL:

- BoxTicketContext – контекст бази даних;
- Performance, Genre, Author, AllTickets, TypeOfTickets, Ticket, StatusTicket – сутності бази даних;
- IBaseRepository<T> – реалізація Repository з generic інтерфейсом;
- BaseRepository<T> – реалізація generic класу для репозиторіїв;

BLL:

- IPerformanceService, ITicketService, IAllTicketsService – інтерфейси сервісів;
- PerformanceService, TicketService, AllTicketsService – імплементація сервісів;

Також на діаграмі зображено основні функції кожного сервіса.

UIL:

- PerformanceController, TicketController, AllTicketController – контролери;

Кінцеві точки REST API(Endpoints)

1. Отримання всіх вистав:
    - Метод: Get
    - URL:/api/performance
2. Отримання всіх вистав за автором:
    - Метод: Get
    - URL:/api/performance/byauthor/{Idauthor}
3. Отримання всіх вистави за жанром:
    - Метод: Get
    - URL:/api/performance/bygenre/{IdGenre}
4. Отримання всіх вистави за назвою:
    - Метод: Get
    - URL:/api/performance/byname/{IdPerformance}
5. Отримання всіх вистав за датою:
    - Метод: Get
    - URL:/api/performance/bydate/{Date}
6. Купівля квитка:
    - Метод: Post
    - URL:/api/ticket/buy/{IdAllTicket}/{SeatNumber}/{FirstName}/{MiddleName}/{LastName}/{NumberPhone}
7. Бронювання квитка:
    - Метод: Post
    - URL:/api/ticket/book/{IdAllTicket}/{SeatNumber}/{FirstName}/{MiddleName}/{LastName}/{NumberPhone}
8. Переведення броньованого квитка в статус проданого
    - Метод: Put
    - URL:/api/ticket/bookedinbought/{IdTicket}
9. Отримання квитка по ID
    - Метод: Get
    - URL:/api/ticket/byid /{IdTicket}

10. Скасування броньованого квитка
    - Метод: Delete
    - URL:/api/ticket/cancelbooked /{IdTicket}
11. Отримання всіх доступних видів квитків на виставу
    - Метод: Get
    - URL:/api/allticket/alltickets /{IdPerformance}
12. Отримання всіх доступних місць на виставу за типом квитка.
    - Метод: Get
    - URL:/api/allticket/allseats /{IdAllTickets}