Національний технічний університет України «КПІ»
Факультет інформатики та обчислювальної техніки
Кафедра Інформаційних систем та технологій

Лабораторна робота №3

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

на тему: « Проектування REST веб-API»

Виконав:
студент групи ІК-13
Хрисанфов Дмитро

Перевірив:
Бардін В.

2023 рік

Завдання:

Теоретична частина:

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.

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.

Варіант:

6	Соціальна мережа. Комунікації між членами мережі	1. Соціальну мережу складають групи її членів, пов'язаних між собою відносинами дружності. 2. Кожний її член може керувати цими відносинами, а саме додавати до своєї мережи друзів: запрошувати до своїх друзів та просити запрошення для себе. 3. Друзі можуть обмінюватись повідомленнями та передивлятись свої розмови у мережі. **Функціональні вимоги:** 1. Створення соціальної мережі; 2. Забезпечення спілкування в ній.

Хід виконання роботи:

С4 модель - це спосіб уявити архітектуру програмного забезпечення у вигляді ієрархії абстракцій. Ця модель складається з чотирьох видів діаграм:

- Діаграма контексту (Context diagram) - це загальний огляд архітектури, який показує взаємодію між програмною системою та її зовнішнім середовищем.
- Діаграма контейнерів (Container diagram) - це більш детальний огляд архітектури, який показує взаємодію між контейнерами, які є фізичними або логічними одиницями, що розміщують компоненти.
- Діаграма компонентів (Component diagram) - це ще більш детальний огляд архітектури, який показує взаємодію між компонентами, які є логічними одиницями, що реалізують певні функції.
- Діаграма коду (Code diagram) - це найдетальніший огляд архітектури, який показує взаємодію між елементами коду, які є найменшою одиницею архітектури.

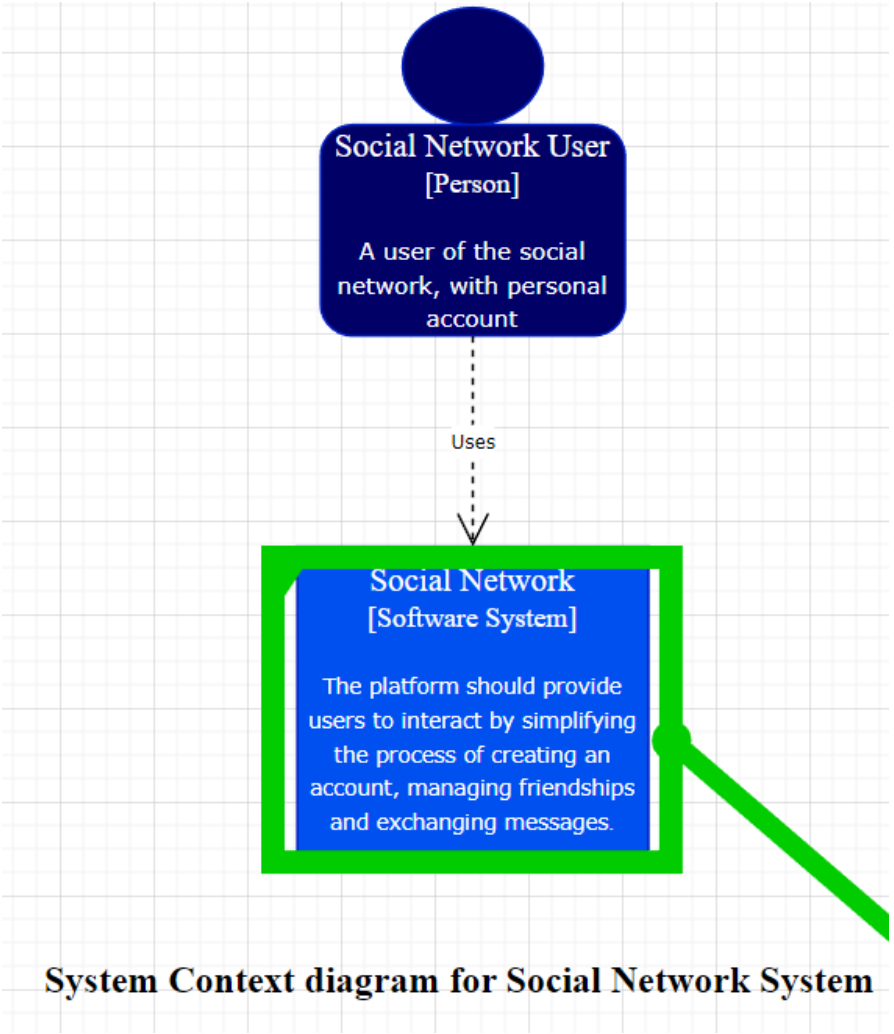


Рис.1 – Context diagram

У системі "Соціальна мережа" існує тип користувача "Користувач", який має можливість зареєструватись у системі та використовувати її для спілкування та обміну повідомленнями з іншими користувачами. Передбачено можливість створення діалогів та обмін повідомленнями між користувачами у межах соціальної мережі.

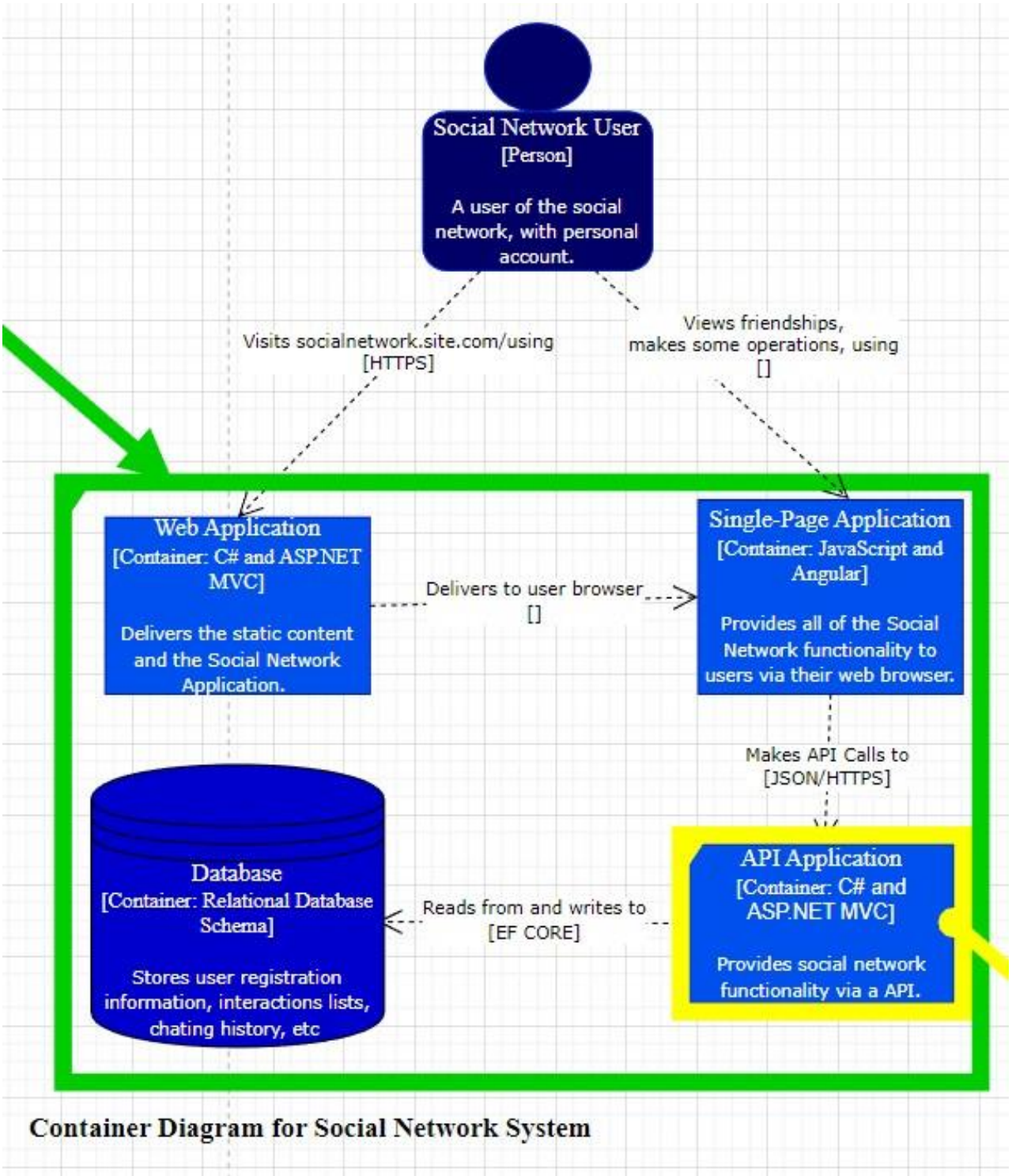


Рис.2 – Container diagram

Щоб отримати більш детальний огляд архітектури, я розбиваю кожен контейнер на його складові частини. Я визначаю, які компоненти містяться в кожному контейнері, що вони роблять і як вони взаємодіють один з одним. Діаграма компонентів показує, як контейнери складаються з компонентів. Вона також показує, які технічні деталі є важливими для кожного компонента.

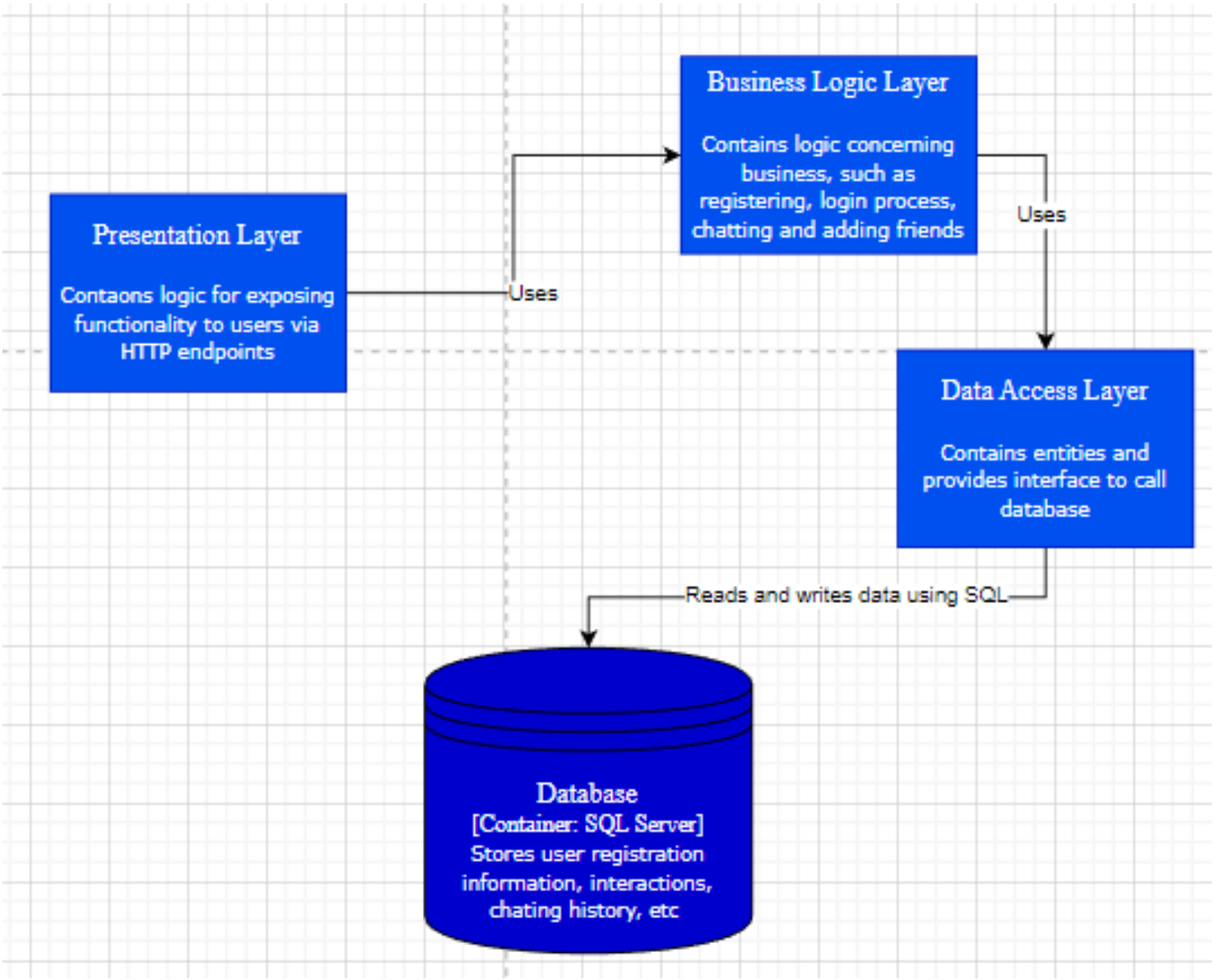


Рис.3 – Component diagram

Для розробки бекенду для застосунку буде використаний архітектурний шаблон MVC. Цей шаблон дозволяє розділити програму на три основні частини, які відповідають за різні завдання:

- **Модель(Model)** відповідає за збереження даних і обробку бізнес-логіки. У цьому випадку модель буде відповідати за зберігання інформації про бюджети та рахунки, а також за обчислення балансу.
- **Вид(View)** відповідає за відображення даних користувачеві. У цьому випадку вид буде відповідати за відображення інтерфейсу користувача для роботи з бюджетом та звітами.
- **Контролер(Controller)** відповідає за обробку вхідних запитів користувача та передачу їх моделі. У цьому випадку контролер буде відповідати за обробку запитів користувача для створення рахунків, додавання транзакцій, переказу коштів тощо.

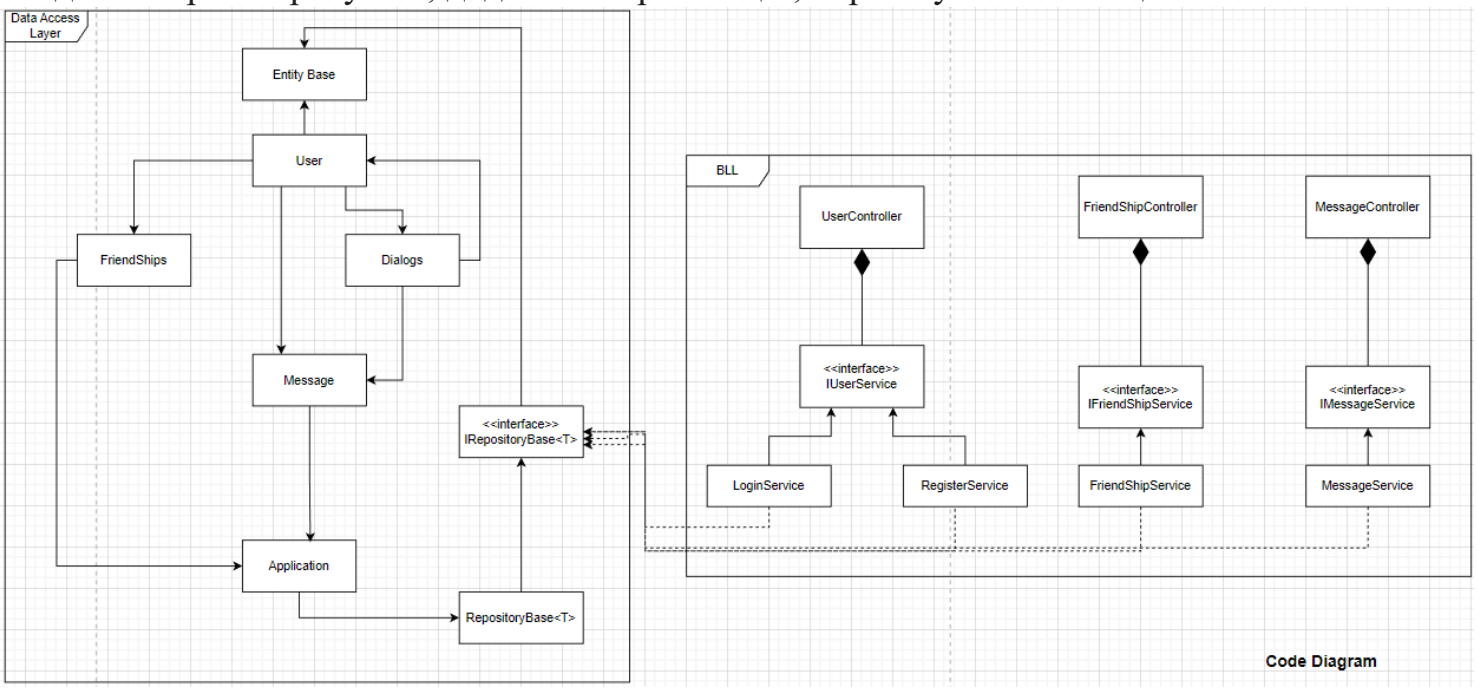
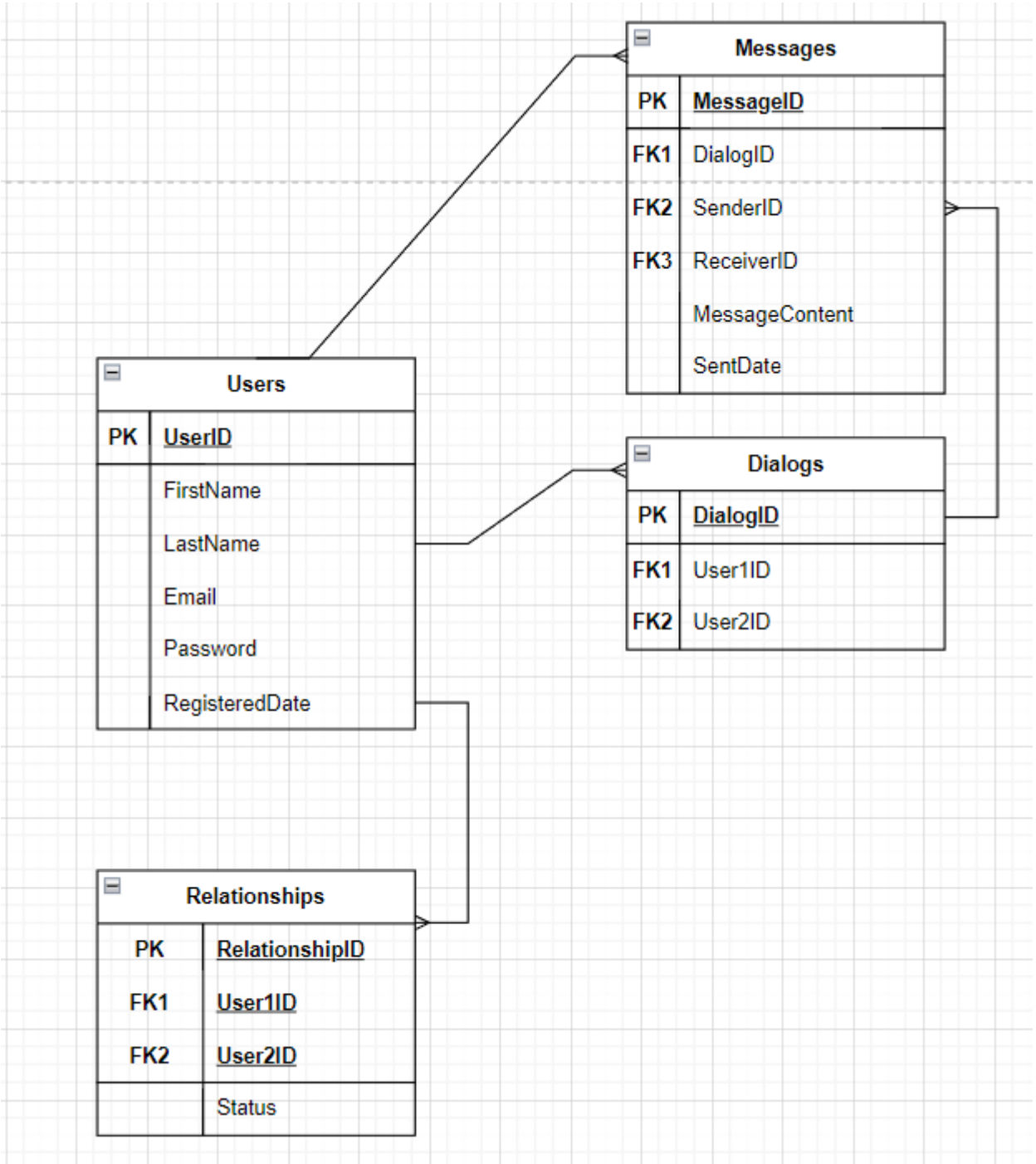


Рис.4 – Code diagram

1. Зв'язки між сутностями:

- Users → Relationships: Користувач може мати багато друзів.
- Users → Dialogs: Користувач може брати участь в багатьох діалогах.
- Dialogs → Messages: Діалог може мати багато повідомлень
- Users → Messages: Користувач може надсилати багато повідомлень

ER діаграма


Messages
PK MessageID
FK1 DialogID
FK2 SenderID
FK3 ReceiverID
MessageContent
SentDate
Users
PK UserID
FirstName
LastName
Email
Password
RegisteredDate
Dialogs
PK DialogID
FK1 User1ID
FK2 User2ID
Relationships
PK RelationshipID
FK1 User1ID
FK2 User2ID
Status

Таблиця користувачів

Поле	Тип	Опис
UserID	INT	Унікальний ідентифікатор користувача
FirstName	NVARCHAR(50)	Ім'я користувача
LastName	NVARCHAR(50)	Прізвище користувача
Email	NVARCHAR(100)	Унікальна адреса електронної пошти користувача
Password	NVARCHAR(100)	Пароль користувача
RegisteredDate	DATETIME	Дата реєстрації користувача

Таблиця відносин

Поле	Тип	Опис
RelationshipID	INT	Унікальний ідентифікатор дружби
UserID	INT	Ідентифікатор користувача, який додав друга
FriendID	INT	Ідентифікатор друга
Status	BIT	Статус дружби (0 - запрошення, 1 - прийнято)

Таблиця діалогів

Поле	Тип	Опис
DialogID	INT	Унікальний ідентифікатор діалогу
User1ID	INT	Ідентифікатор першого користувача в діалозі
User2ID	INT	Ідентифікатор другого користувача в діалозі

Таблиця повідомлень

Поле	Тип	Опис
MessageID	INT	Унікальний ідентифікатор повідомлення
DialogID	INT	Ідентифікатор діалогу, в якому було надіслано повідомлення
SenderID	INT	Ідентифікатор користувача, який надіслав повідомлення
ReceiverID	INT	Ідентифікатор користувача, який отримав повідомлення
MessageContent	NVARCHAR(500)	Текст повідомлення
SentDate	DATETIME	Дата та час надіслання повідомлення

Endpoints:

1. Створення користувача (Реєстрація):

HTTP метод: POST URL: /api/users/register

2. Вхід користувача (Логін):

HTTP метод: POST URL: /api/users/login

3. Зміна паролю користувача:

HTTP метод: PUT URL: /api/users/changePassword/{userID}

4. Додавання запиту на дружбу:

HTTP метод: POST URL: /api/relationships/sendRequest

5. Прийняття/Відхилення запиту на дружбу:

HTTP метод: POST URL: /api/relationships/respondToRequest

6. Отримання списку друзів:

HTTP метод: GET URL: /api/relationships/getFriends/{userID}

7. Видалення друзів:

HTTP метод: DELETE URL: /api/relationships/removeFriend/{userID}/{friendID}

8. Створення діалогу:

HTTP метод: POST URL: /api/dialogs/createDialog

9. Отримання діалогів користувача:

HTTP метод: GET URL: /api/dialogs/getUserDialogs/{userID}

10. Надсилання повідомлення:

HTTP метод: POST URL: /api/messages/sendMessage

11. Отримання повідомлень у діалозі:

HTTP метод: GET URL: /api/messages/getMessages/{dialogID}

12. Отримання інформації про користувача:

HTTP метод: GET URL: /api/users/getUserInfo/{userID}