КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет комп'ютерних наук та кібернетики

Кафедра теорії та технології програмування

Кваліфікаційна робота

на здобуття ступеня бакалавра

за спеціальністю 122 Комп'ютерні науки

на тему:

СЕРВІС ОРГАНІЗАЦІЇ НАВЧАЛЬНИХ ОНЛАЙН-РЕСУРСІВ

Виконав студент 4-го курсу
Максим МИКИТЮК

Науковий керівник:
доцент, кандидат фізико-математичних наук
Тарас ПАНЧЕНКО

_____
(підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент 

Роботу розглянуто й допущено до захисту на засіданні кафедри математичних основ комп'ютерних наук

«____» _____ 2023 р.,
протокол № ___

Завідувач кафедри

професор, доктор фізико-математичних наук

Микола НІКІТЧЕНКО (підпис)

Київ – 2023

РЕФЕРАТ

Обсяг роботи 37 сторінок, 10 ілюстрацій, 1 таблиця, 27 джерел посилань.

ОНЛАЙН-ОСВІТА, ДОШКА ЗАВДАНЬ, СЕРВІС, ПРИКЛАДНИЙ КОРИСТУВАЦЬКИЙ ІНТЕРФЕЙС, БАЗА ДАНИХ

Об’єктом даної роботи є процес дистанційної освіти. Предметом даної роботи є веб-сервіси для організації навчального процесу.

Метою роботи є проектування і розробка серверної частини та користувацького інтерфейсу веб-сервісу з функціями дошки завдань для допомоги студентам у організації власного навчального процесу.

Методами розробки є проектування бази даних, побудова архітектури, розробка веб-застосунку на основі моделей. Інструменти – мова C#, Visual Studio Community 2022, вбудовані фреймворки.

Результати: проаналізовано існуючі рішення, визначено проблематику, поставлені задачі, спроектовано систему, розроблено серверну частину системи для підтримки освітнього процесу.

Сфера застосування – проект може бути застосований студентами та учнями в процесі навчання в онлайн форматі.

Значимість роботи полягає насамперед в потенційному збільшенні продуктивності студентів, що позитивно впливає на якість засвоєної інформації під час навчання та зменшенні негативних факторів дистанційного формату освіти.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

ASP.NET – Active Server Pages, ASP.NET Core, фреймоврк .NET Core

CRUD – Create, Read, Update, Delete

HTML – HyperText Markup Language

CSS – Cascading Style Sheets

API - Application Programming Interface

IDE - integrated development environment

C# - мова програмування C#, використовується в розробці

СКБД – Система керування базами даних

SQL - Structured query language

EF – Entity Framework Core

БД – База даних

CLI - Command-line interface

ІД - ідентифікатор

JWT – JSON Web Token

CORS – Cross-Origin Resource Sharing

ВСТУП

Оцінка сучасного стану об’єкта розробки. Навчальний процес став все частіше відбуватись у онлайн-форматі. Цей формат залишається актуальним не лише через зовнішні обставини, які ускладнюють живе відвідування навчальних закладів, а й через доступність, через поширення портативних пристроїв та активну розробку веб-додатків.

В результаті переходу на онлайн-освіту студенти стикаються з новим набором проблем, що створює нагальну потребу в адаптованій системі, яка може подолати ці перешкоди.

Більшість онлайн-курсів вимагають від студентів отримання навчальних матеріалів з різних джерел, таких як системи управління навчанням, онлайн-бібліотеки та зовнішні веб-сайти. Через таку фрагментацію студентам може бути складно ефективно знаходити та організовувати свої ресурси.

Далі, через адаптивну природу онлайн-навчання деяким студентам може бути складно ефективно розподіляти свій час і підтримувати дисципліну. Для них це потенційна проблема – визначити пріоритети, уникати відволікань і дотримуватися навчального розпорядку за відсутності структурованого середовища.

Онлайн-навчання вимагає від студентів активного та постійного перемикання уваги кількома завданнями одночасно, такими як відвідування лекцій, участь в онлайн-спілкуванні та виконання завдань. Підвищене навантаження і необхідність виконувати декілька задач паралельно можуть негативно вплинути на результати навчання і ускладнити сприйняття інформації.

Наступною слідує брак способів для студента організовувати свій навчальний простір самостійно. Цілком можливо, що стандартні системи

управління завданнями не відповідають потребам студентів, які виконують більшу частину обсягу робіт онлайн. Студентам потрібне рішення, пристосоване до їхніх потреб, яке зможе збільшити продуктивність.

Актуальність роботи та підстави для її виконання. Вирішенням вищеназваних проблем може виявитись розробка сервісу для підтримки навчального процесу. Тому системи які дозволяють долати складнощі онлайн-освіти та збільшують ефективність користувачів досі актуальні. Зокрема, підставою для створення даної розробки була потреба в рішенні для студентів, які страждають від розпорошеності навчальних джерел та потребують нових інструментів для більшої систематизації їхньої освіти.

Мета й завдання роботи. Метою кваліфікаційної роботи є створення серверної частини веб-додатку для управління індивідуальними завданнями студентів. Для досягнення було поставлено наступні задачі:

а) Визначити проблематику

б) Дослідити існуючі рішення;

в) Спроектувати базу даних та серверну частину

г) Реалізувати серверну частину за допомогою обраних технологій згідно архітектури;

Функціональні вимоги до розробки:

а) Можливість додавати, редагувати та видаляти завдання;

б) Можливість переглядати завдання у різних представленнях;

в) Функція поширення інформацією з карток між користувачами;

г) Можливість відслідковувати прогрес по виконанню завдань.

Об'єкт, методи й засоби розроблення. Об’єктом дослідження є навчальний процес в онлайн форматі. Предметом дослідження є веб-сервіси для допомоги в організації навчального процесу. Серед методів для

виконання завдань, варто зазначити проектування, моделювання та розробку системи на основі створених моделей.

РОЗДІЛ 1. ОГЛЯД НАЯВНИХ НА РИНКУ СИСТЕМ

1.1 Сервіс Trello

Trello - це популярна веб-програма для організації та керування діяльністю у візуально привабливий та зручний спосіб. Вона надає користувачам можливість створювати дошки, списки та картки для організації завдань і має функцію перетягування для зручного переміщення елементів [1]. Trello також пропонує широкий набір функціональних можливостей, включаючи делегування завдань, встановлення дедлайнів, мітки та контрольні списки. Він також інтегрується з багатьма сторонніми додатками, такими як Google Drive, Slack та GitHub.

Переваги використання Trello включають зручний інтерфейс, гнучкість організації завдань, адаптивність інструментів для співпраці та інтеграцію з іншими сервісами. Однак, сервіс має обмежені інструменти аналітики та звітності, а також обмежені можливості кастомізації макетів дошок завдань. Деякі користувачі також можуть вважати, що Trello не надає ефективних рішень для конкретних сфер діяльності, що потребують специфічного налаштування.

Загалом, Trello є популярним інструментом зі зручним інтерфейсом та широкими функціональними можливостями, але має обмеженість у аналітиці та специфічних налаштуваннях для окремих сфер діяльності.

1.2 Система управління проектами Asana

Asana є комплексною дошкою завдань та інструментом управління проектами, який пропонує широкі можливості для управління та відстеження діяльності [3]. Вона дозволяє користувачам організовувати свою роботу,

встановлювати залежності між завданнями, створювати під задачі, визначати проміжні етапи і графіки проектів. Asana також надає сервіси для аналітики та звітності, які дозволяють відстежувати прогрес проектів і аналізувати продуктивність [4].

Однак, Asana має деякі недоліки. Перш за все, вона має круту криву навчання, і нові користувачі можуть знайти її функціональність та можливості переповненими. Використання всіх функцій може бути викликом, але з практикою користувачі можуть отримати значні переваги для керування своєю роботою та проектами.

Крім того, Asana не має прямого інтерфейсу для певних сторонніх додатків, хоча вона інтегрується з багатьма іншими програмами і сервісами. Це може вимагати використання обхідних шляхів або послуг третіх сторін для інтеграції обраних інструментів з Asana.

В цілому, Asana є потужним інструментом для управління завданнями і проектами зі значними аналітичними можливостями і гнучким інтерфейсом. Проте, новим користувачам може знадобитися час для ознайомлення з ним, а деякі інтеграційні обмеження можуть вимагати додаткових зусиль.

1.3 Веб сервіс Google Classroom

Google Classroom - це веб-рішення, яке допомагає викладачам управляти навчальним процесом у класі [5]. Вона надає централізовану платформу для адміністрування класів, організації матеріалів і спілкування з учнями. Google Classroom інтегрується з іншими продуктами Google, такими як Google Drive, Google Docs і Google Calendar, що полегшує обмін файлами та співпрацю над завданнями [6]. Вона також допомагає вчителям уникнути використання друкованих копій завдань і спрощує процес оцінювання. Платформа сприяє активній участі та спільній роботі в класі шляхом

можливості оголошень, обговорень і надання зворотного зв'язку [7]. Однак Google Classroom має функціональні обмеження на дошці завдань і вимагає використання продуктів Google. Незважаючи на це, вона є корисним інструментом для викладачів, який спрощує управління класами та полегшує співпрацю [8].

1.4 Дошка завдань ClickUp

ClickUp є веб-дошкою для завдань та додатком для управління проектами, які пропонують широкий спектр функцій для планування, моніторингу та спільної роботи. Він має можливості налаштування та представлення завдань, дозволяє делегувати завдання, встановлювати терміни, створювати під задачі та відстежувати час. ClickUp також пропонує різноманітні варіанти налаштування макетів дошок і списків, що дозволяє користувачам персоналізувати свою роботу. Він також інтегрується з іншими програмами та сервісами, забезпечуючи централізовану інформацію та полегшуючи комунікацію. Через свої широкі можливості та налаштування, ClickUp може вимагати деякого часу для освоєння, але він надає потужні інструменти для управління завданнями та проектами [9].

1.5 Система Moodle

Moodle є безкоштовною системою управління навчанням (LMS) з відкритим кодом, призначеною для бізнесу та вищих навчальних закладів [10]. Вона надає широкий спектр функціональних можливостей для організації онлайн-навчання та управління курсами. Moodle дозволяє викладачам розробляти та управляти курсами, доставляти матеріали, оцінювати студентів, стимулювати дискусії та відстежувати прогрес. Вона

також забезпечує засоби взаємодії та командної роботи, такі як форуми, чати та платформи обміну повідомленнями [11].

Одним з переваг Moodle є його гнучкість. Він дозволяє користувачам налаштовувати платформу відповідно до їхніх потреб, надаючи великий вибір плагінів, тем та інструментів налаштування. Це дозволяє установам адаптувати Moodle до власних робочих процесів і потреб.

Moodle також має активну спільноту користувачів, викладачів та розробників, яка підтримує його розвиток та забезпечує доступ до різноманітних плагінів та ресурсів. Ця спільнота надає підтримку, обмін кращими практиками та розширення функціональності.

Незважаючи на свої переваги, Moodle має деякі недоліки. Початкове налаштування та крива навчання можуть вимагати часу та технічних знань. Крім того, платформа не має такого самого рівня спеціалізованої функціональності дошки завдань, як окремі рішення для управління завданнями.

У підсумку, Moodle є потужною та адаптивною системою управління навчанням, яка забезпечує повний спектр функціональних можливостей для онлайн-навчання та курсів. Вона може не мати специфічної функціональності дошки завдань, але надає освітнім установам та організаціям потужну платформу для проведення курсів, співпраці та моніторингу прогресу студентів.

1.6 Порівняння наявних рішень

Для більшої показовості наведено таблицю з окремими ключовими моментами з аналізу існуючих рішень:

	Trello	Asana	ClickUp	Google Classroom	Moodle
Користувач може налаштовувати свій робочий простір	Так	Так	Так	Ні	Так
Сервіс створений для оптимізації онлайн-освіти	Ні	Ні	Ні	Так	Так
Сервіс реалізовує функціонал дошки завдань	Так	Так	Так	Ні	Ні
Наявні функції окрім дошки завдань	Ні	Так	Ні	Так	Так
Користувачі можуть взаємодіяти між собою	Так	Так	Так	Так	Так
Наявність аналітики для завдань	Так (обмежена)	Так	Так	Так (обмежена)	Так

Таблиця 1 – аналіз існуючих рішень

Дана табличка ілюструє відмінності та спільні риси по переліченим вище існуючим рішенням. З неї можна зробити висновок, що не всі вони надають користувачу можливість власноруч налаштовувати власний робочий простір та не всі мають першочерговим призначенням безпосередньо функціонал дошки оголошень.

Щодо більш важливих відмінностей, варто зауважити, що лише Google Classroom та Moodle були створені насамперед для допомоги в освітніх процесах, і вони не мають на меті полегшити студентам процес організації особистих завдань.

Також всі існуючі сервіси надають можливість користувачам взаємодіяти між собою. Тобто, якщо розроблений в роботі сервіс не буде мати механізмів взаємодії між студентами, він буде негативно виділятись на фоні інших.

І насамкінець, варто виокремити можливості аналітики – всі рішення надають користувачу інструменти для оцінки виконаної та невиконаної роботи, що є важливим, якщо розглядати саме організацію завдань. Тому цей пункт є також важливим для сервісу.

РОЗДІЛ 2. ВИКОРИСТАНІ ТЕХНОЛОГІЇ

2.1 Середовище розробки. IDE Visual Studio Community

Visual Studio Community - інтегроване середовище розробки (IDE), розроблене Microsoft, яке спрощує процес розробки, налагодження та розгортання додатків, з фокусом на мові програмування C#. Це безкоштовне інструмент для індивідуальних програмістів, студентів та невеликих команд. IDE підтримує широкий спектр мов програмування та фреймворків, має дружній інтерфейс та багато шаблонів проектів для веб-розробки.

IDE легко інтегрується з різними технологіями та фреймворками, такими як ASP.NET і Entity Framework, надаючи надійну підтримку для розробки веб-додатків. Функція IntelliSense спрощує написання коду та економить час. Вбудоване середовище налагодження забезпечує ефективну роботу з точками зупинки та діагностикою в реальному часі.

Visual Studio Community має велику екосистему розширень і модулів, які дозволяють адаптувати середовище до потреб проекту. Інструменти сприяють командній роботі, підтримуючи взаємодію з системами контролю версій, такими як Git. Спільнота Visual Studio надає багато документації, навчальних ресурсів та форумів для підтримки і обміну знаннями [12].

2.2 Фреймворк ASP.NET Core

ASP.NET Core - надійний і широко використовуваний веб-фреймворк, що набув значної популярності серед розробників програмного забезпечення [13]. Він працює на різних операційних системах, включаючи Windows, Linux і macOS, надаючи розробникам більшу гнучкість і свободу вибору платформи. Фреймворк забезпечує швидкість онлайн-додатків завдяки

використанню передових технік веб-розробки та оптимізації, таких як вбудоване кешування і асинхронне програмування. Архітектурний шаблон ASP.NET Core підтримує модульність, що сприяє ефективному розробленню та обслуговуванню додатків. Його масштабованість і сумісність з хмарними середовищами та архітектурою мікросервісів роблять його відмінним вибором для розробки великомасштабних додатків. ASP.NET Core також надає комплексний набір засобів для безпеки, включаючи захист від поширених вразливостей і підтримку стандартних протоколів автентифікації [14].

2.3 Фреймворк Entity Framework Core

EF Core є фреймворком об'єктно-реляційного відображення (ORM), створеним Microsoft. Він дозволяє розробникам взаємодіяти з реляційними базами даних в об'єктно-орієнтованому стилі, що спрощує роботу зі складними SQL-запитами. EF Core забезпечує високий рівень абстракції, що дозволяє розробникам сконцентруватися на бізнес-логіці, а не на операціях з базами даних [15].

Фреймворк підтримує безліч постачальників баз даних, таких як Microsoft SQL Server, SQLite, PostgreSQL і MySQL, що дає змогу легко переключатися між ними без необхідності змінювати код доступу до даних. EF Core інтегрований з LINQ, мовою запитів для додатків .NET, що дозволяє виконувати складні запити до баз даних в зрозумілому та читабельному способі.

Крім того, EF Core має систему міграцій, яка спрощує внесення змін до структури бази даних з часом, зберігаючи раніше збережені дані. Технологія також надає функції для модульного тестування коду доступу до даних,

включаючи провайдери баз даних в пам'яті для створення тестів без реальної бази даних [16].

Загалом, EF Core є потужним і гнучким фреймворком, який спрощує взаємодію з базами даних і полегшує розробку додатків з використанням ASP.NET Core.

2.4 Система керування базами даних Microsoft SQL Server

SQL Server є потужною системою керування базами даних, спеціально розробленою для ефективного управління великими обсягами даних і транзакцій [17]. Це досягається завдяки складним стратегіям оптимізації запитів, механізмам індексування та паралельної обробці, що забезпечують виняткову продуктивність. SQL Server підтримує горизонтальну та вертикальну масштабованість, що дозволяє розширювати систему баз даних зі зростанням обсягу даних. Він також надає широкий спектр опцій безпеки, забезпечуючи цілісність і конфіденційність даних. SQL Server легко інтегрується з іншими продуктами Microsoft і має потужні інструменти для бізнес-аналітики та аналізу даних. Мова програмування SQL Server - Transact-SQL (T-SQL) - дозволяє створювати складні сценарії баз даних і збережені процедури. Система надає функції високої доступності та реплікації даних для забезпечення постійного доступу до даних і зменшення ризиків збоїв. Крім того, SQL Server має компоненти SSAS і SSRS, які надають широкі можливості для аналізу даних і створення звітів у різних форматах файлів [18].

2.5 Архітектура RESTful API

RESTful API є поширеним архітектурним стилем розробки, який надає стандартний та масштабований метод для міжсистемного зв'язку та обміну даними [19]. Вони побудовані навколо ідеї ресурсів, що представляють основні сутності або об'єкти API. Кожен ресурс ідентифікується унікальною URL-адресою. RESTful API дотримуються парадигми комунікації без стану, де кожен запит містить усю необхідну інформацію, і сервер не зберігає стан клієнта між запитами. Захист доступу до ресурсів часто забезпечується механізмами автентифікації та авторизації, такими як ключі API, токени та OAuth. Документація RESTful API надає детальну інформацію про доступні кінцеві точки, методи HTTP, формати даних та структури запитів-відповідей. За допомогою принципів проектування RESTful API розробники можуть створювати масштабовані та сумісні веб-інтерфейси, які легко використовуватимуться різними клієнтами [20].

2.6 Авторизація. JSON Web Token та Шифрування паролів

Для реалізації авторизації користувачів можна використовувати JSON Web Token (JWT). Це стандартний механізм для передачі токенів в безстанових середовищах, таких як HTTP. JWT використовується для ідентифікації та авторизації користувачів, забезпечуючи безпечну комунікацію між клієнтом та сервером [21].

JWT складається з трьох частин: заголовка, набору заявок та підпису. У заголовку зазначається тип токена та алгоритм шифрування. Набір заявок містить інформацію про користувача, таку як ідентифікатор та ролі. Підпис гарантує цілісність токена та його автентичність.

При вході в систему користувач отримує JWT, який зберігається на клієнтському боці (зазвичай в локальному сховищі). Після цього, при кожному запиті до сервера, токен передається у заголовку або запиті. Сервер перевіряє підпис та достовірність токена, використовуючи секретний ключ, і авторизує користувача на основі отриманої інформації.

Також для підвищеної безпеки, паролі не зберігаються в базі даних в оригінальному вигляді. Для уникнення потенційних небезпек, вони зашифровуються в серверній частині за допомогою бібліотеки BCrypt.Net, яка містить основні алгоритми криптографії та дозволяє звіряти зашифровані паролі [22].

2.7 Хмарна платформа Microsoft Azure

Azure надає широкі можливості для публікації Web-API систем, розроблених в Visual Studio на ASP.NET, та забезпечує потужну інфраструктуру для баз даних, необхідних для цих систем [23]. Одним з головних переваг використання Azure є його висока масштабованість та гнучкість. За допомогою Azure App Service, можна легко розгортати та масштабувати Web-API, забезпечуючи високу доступність та продуктивність [24]. Технології, такі як автоматичне масштабування та балансування навантаження, дозволяють ефективно управляти ресурсами та забезпечувати стабільну роботу систем навіть при високому навантаженні.

Azure також надає розширені можливості для зберігання та керування базами даних. Azure SQL Database, наприклад, забезпечує масштабовану, стійку та безпечну платформу для зберігання даних Web-API. Його функції включають автоматичне резервне копіювання, відновлення даних, високу доступність та захист від вторгнень [25]. Крім того, Azure Cosmos DB надає глобально розподілену базу даних, що дозволяє зберігати та реплікувати дані

на всіх континентах, що забезпечує швидкий доступ до даних з будь-якої точки світу [26].

Azure також підтримує широкий набір інструментів для розробки, тестування та налагодження Web-API. Зокрема, можна використовувати Visual Studio для створення, збирання та публікації елементів безпосередньо в Azure. Інтеграція між Visual Studio та Azure забезпечує швидкий та зручний процес розгортання та керування Web-API. Крім того, Azure DevOps забезпечує можливості для автоматизації процесу розгортання та надання послуг Continuous Integration/Continuous Deployment (CI/CD) [27].

Загалом, Azure надає розробникам Web-API на ASP.NET широкі можливості для ефективного розгортання, масштабування та управління базами даних. Висока масштабованість, потужність та надійність Azure дозволяють створювати стабільні та швидкодіючі застосунки з безперебійним доступом до даних. Інструменти, такі як Visual Studio і Azure DevOps, спрощують процес розробки та розгортання, забезпечуючи швидку та зручну роботу з Web-API.

РОЗДІЛ 3. ПРОЦЕС РОЗРОБКИ

3.1 Короткий опис процесу

Під час розробки було використано IDE Visual Studio Community для бекенду, фреймворк ASP.NET Core для реалізації веб-функціоналу, ORM фреймворк EF Core та СКБД MS SQL Server для створення та взаємодії з базою даних і Visual Studio Code та веб-фреймворк Angular для розробки користувацького інтерфейсу.

При розробці даного сервісу було використано підхід code-first, тобто, початково було розроблено моделі для серверної частини та визначено зв’язки між ними, а опісля цього за допомогою EF Migrations було згенеровано базу даних.

Проектування було здійснено за допомогою UML, відштовхуючись від необхідного функціоналу. Зв’язки також створено таким чином, щоб уникнути рекурсій та потенційних нескінченних циклів. За допомогою міграцій визначити проблемні місця було нескладно, оскільки фреймворк повідомляє про подібні помилки на етапі створення БД.

Архітектурою було обрано RESTful API, що зумовлює більшу гнучкість та надійність API. Вона реалізована за допомогою контролерів, які слугують в якості функцій для обміну інформацією з клієнтом, приймаючи всі необхідні дані для проведення операції та повертаючи результат.

3.2 Проектування. UML-діаграми

Першим етапом розробки було проектування. Для визначення необхідних функцій програми створено UML-діаграму Use Case, яка зображує приклад використання користувачем. Користувач (позначений

актором) може здійснювати дії описані в овалах, які можуть переходити в інші дії, або доповнювати одні одних (додаток А).

Наступною було створено діаграму класів. Оскільки використовується підхід code-first, цей етап є критично важливим, оскільки на основі нього згодом буде створено моделі та згенеровано базу даних.

Дана діаграма (додаток Б) зображує різні класи та взаємозв'язки між ними. Зокрема присутні класи які мають виключно функціональне призначення, в тому числі і в якості DTO (Data Transfer Object, як AuthUser.cs) та не будуть зберігатись в базі даних.

3.3 Розробка бази даних

У результаті реалізованих моделей та міграцій Entity Framework Core було створено базу даних (додаток В). Для власне керування цією базою даних було використано MS SQL, одну з найпоширеніших СКБД. З метою початкового заповнення рядків таблиць для подальших тестів був використаний графічний інтерфейс, що належить вищеназваній системі. Цей інтерфейс надає зручні можливості для взаємодії з базою даних, дозволяючи виконувати різноманітні операції, такі як створення таблиць, введення початкових даних та виконання тестів для перевірки функціональності бази даних. Використання графічного інтерфейсу спрощує процес роботи з базою даних, роблячи його більш інтуїтивно зрозумілим та доступним для користувачів.

3.4 Оптимізація безпеки у використанні системи

Для забезпечення надійності використання системи, було налаштовану політику CORS, яка дозволяє функціонування веб-додатків з різним походженням.

Також паролі користувачів зберігаються в базі в зашифрованому вигляді, що дозволяє зберігати конфіденційність даних навіть в разі потенційних проблем.

Рисунок 1 – Приклад хешу пароля “password”

Паролі зашифровуються за допомогою алгоритму SHA512 та розшифровуються безпосередньо в API.

3.5 Тестування за допомогою Swagger

Тестування було здійснено за допомогою Swagger із розширеними налаштуваннями. Наприклад, ось перевірка функцій авторизації з його допомогою:

Надаємо пошту та пароль в метод register

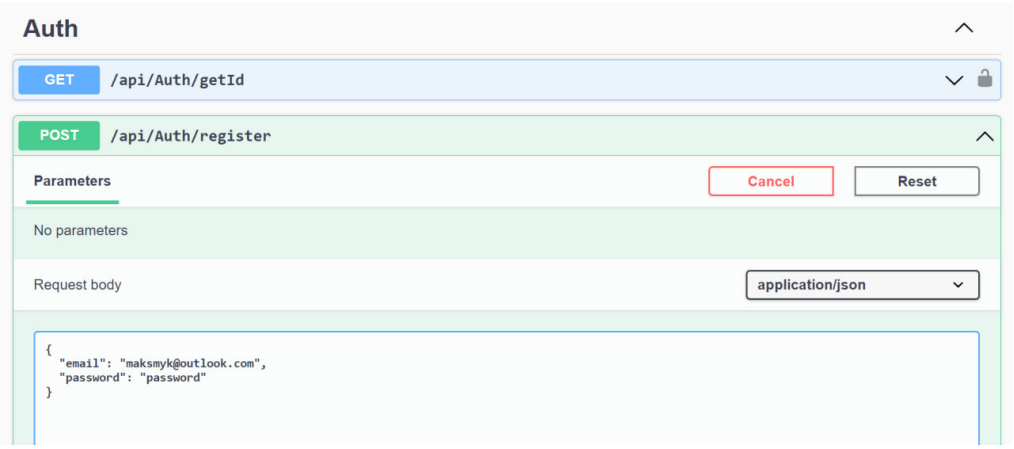


Рисунок 2 – Введення даних в Swagger

Реєстрація пройшла успішно. Результат:

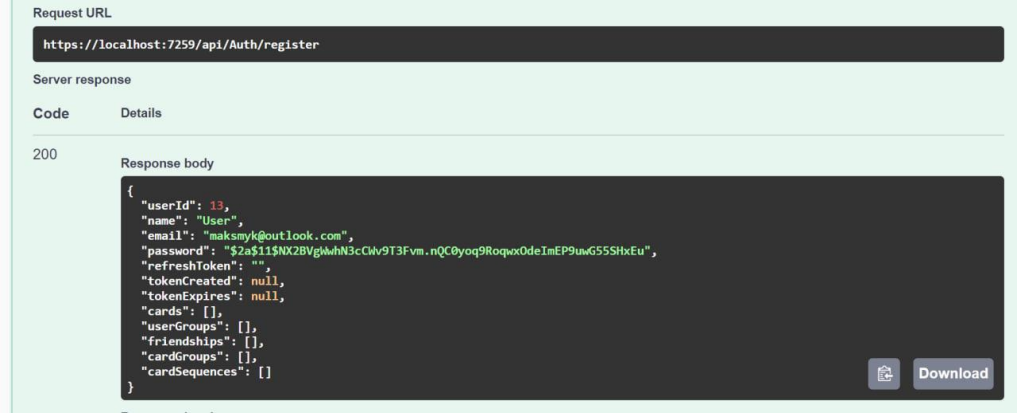


Рисунок 3 – Результат реєстрації

1. Опісля вводимо той же логін і пароль в метод login
2. У відповідь отримуємо JWT токен що містить інформацію про себе та користувача
3. Вводимо токен у віконце, забезпечене Swagger

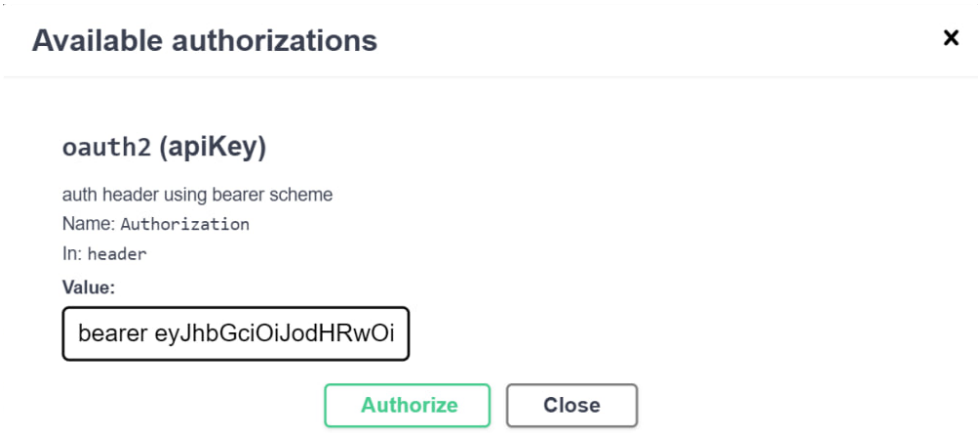


Рисунок 4 – Авторизація за допомогою токена

4. Авторизація успішна. Ми тепер можемо отримати ІД користувача, який користується системою в даний момент (цей метод вимагає авторизації для доступу).

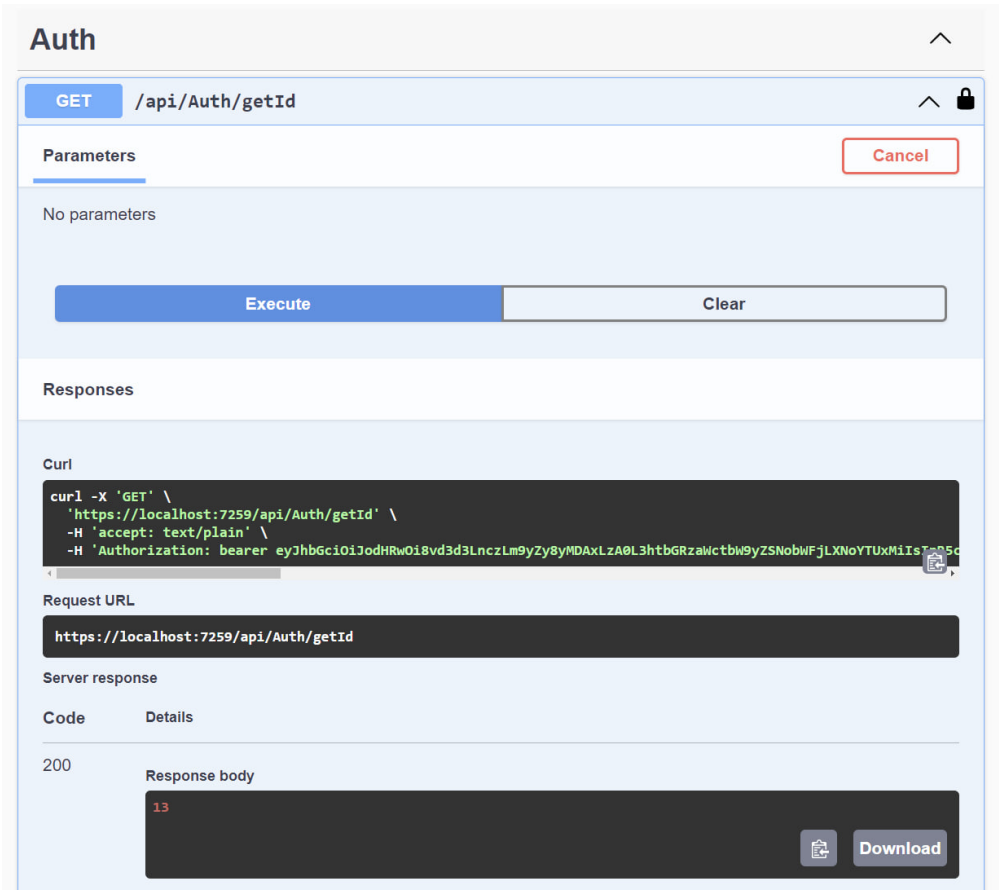


Рисунок 5 – використання методу, для якого необхідний доступ

5. Також ми можемо перевірити валідність нашого токена, виділивши з нього інформацію за допомогою сервісу jwt.io:

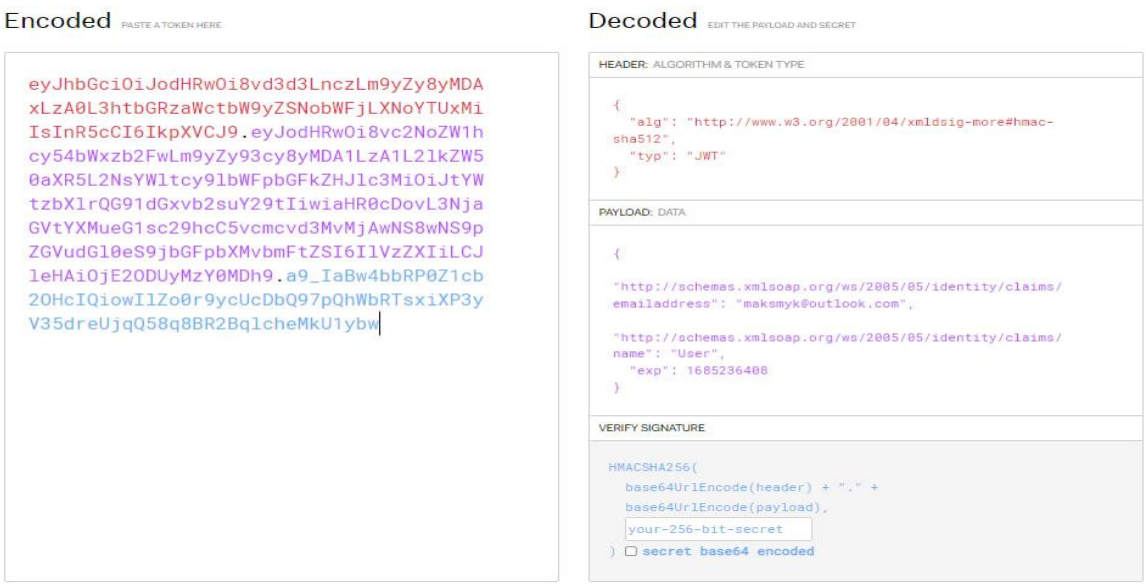


Рисунок 6 – Розшифрований JWT-токен

6. Переконуємось, що інформація відповідає дійсності.

Також по аналогії створюємо для даного користувача завдання. Залишаємо лише необхідні поля, які не можуть мати порожнього значення.

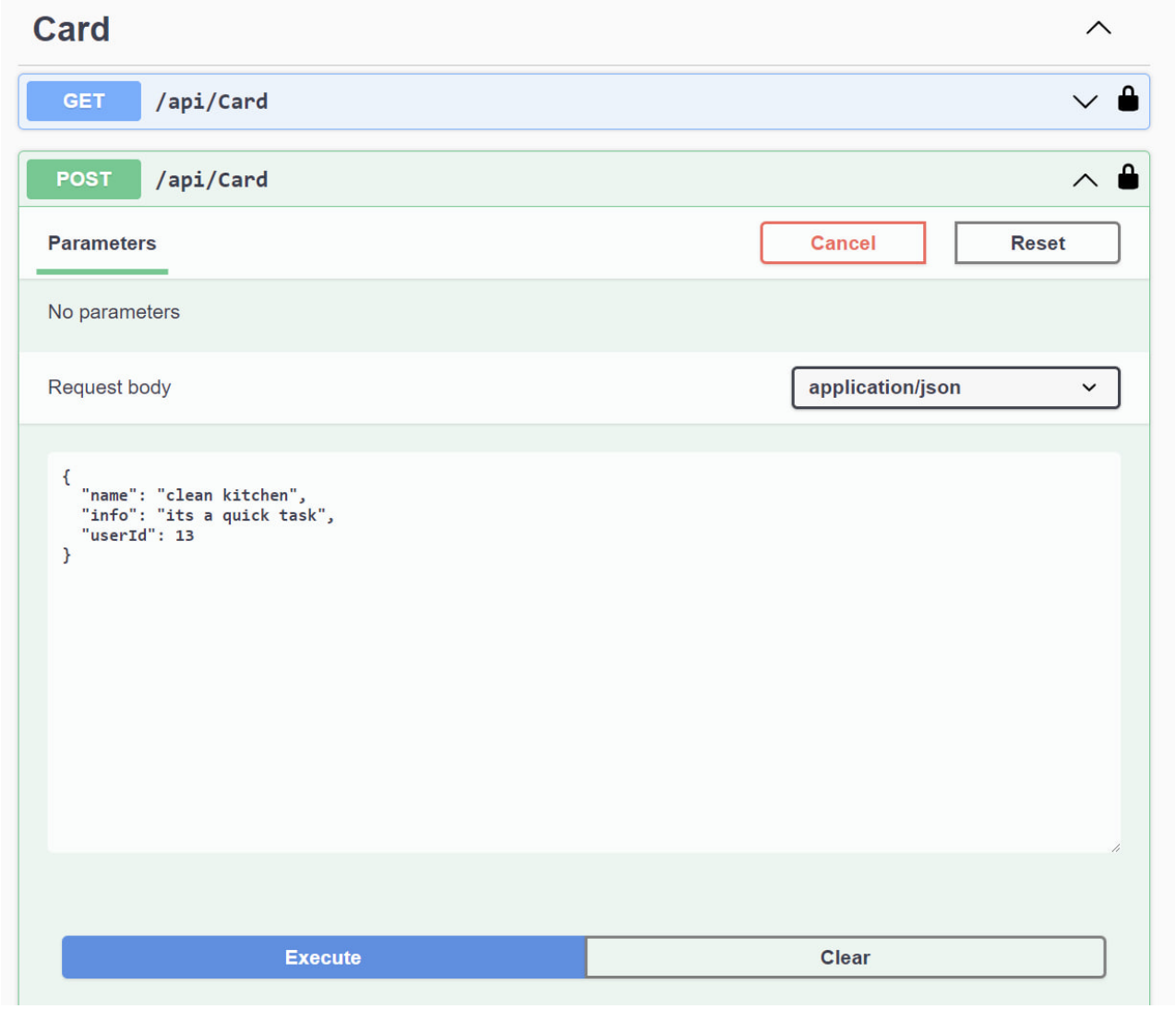


Рисунок 7 – Створення завдання

Після того, як ми ініціюємо процес надсилання запиту, ми отримуємо відповідь, яка містить об'єкт класу Card. Об'єкт класу Card містить в собі важливу інформацію, яка включає, наприклад, ідентифікатор картки, дату початку виконання, статус та інше. Отримання відповіді з об'єктом класу Card дозволяє нам подальше використання цієї інформації для різних

операцій, аналізу або візуалізації даних, залежно від наших потреб та бізнес-логіки системи.

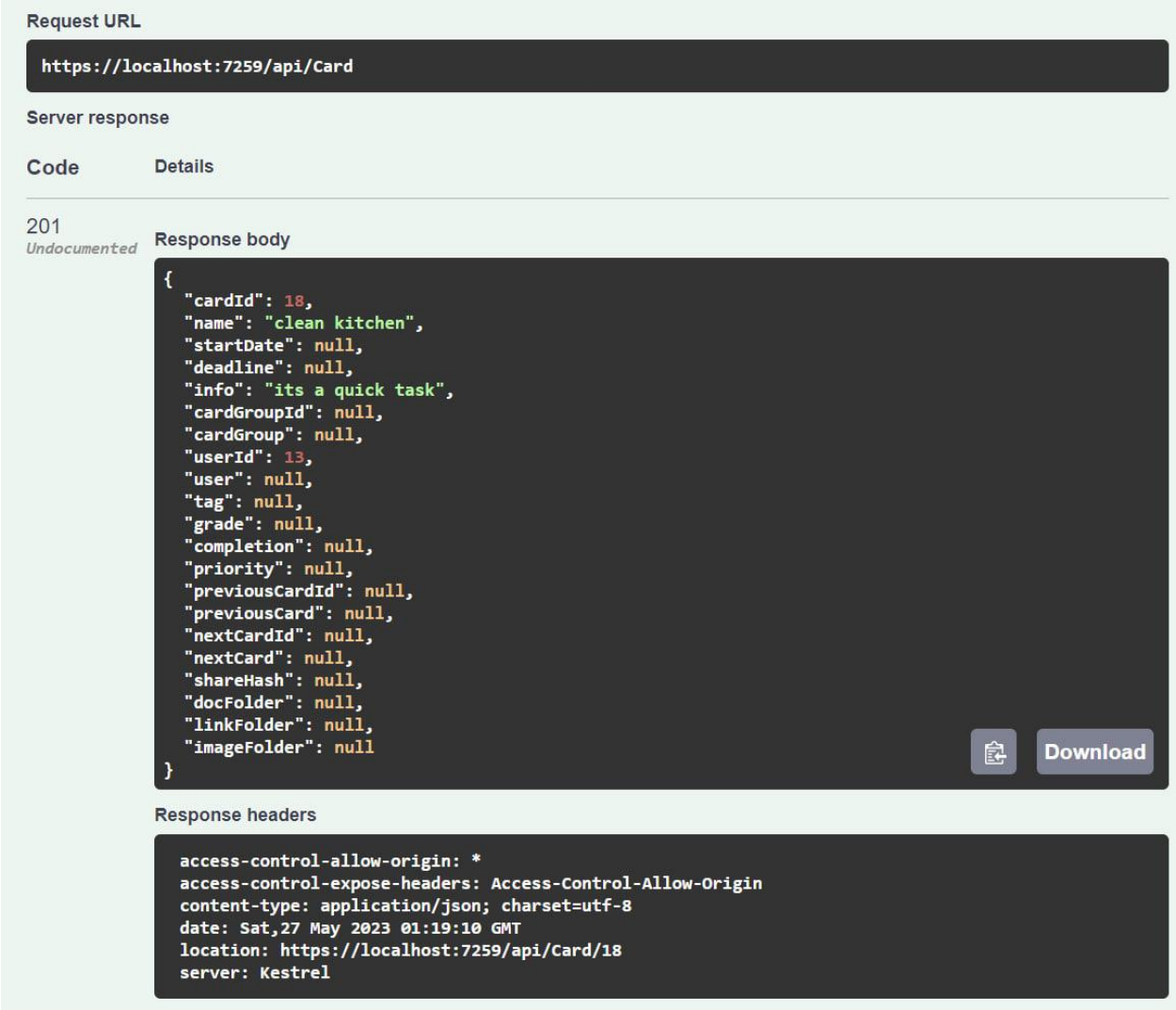


Рисунок 8 – Результат тестування контролера

Завдання було успішно створене. В процесі створення запису, всі поля, які можуть бути заповнені користувачем, залишилися порожніми, і їх значення було призначено як null. Це означає, що вони не містять жодних конкретних даних або значень, але можуть бути змінені в майбутньому, коли користувач буде вводити або надавати потрібну інформацію. Наприклад, у подальшому, коли користувач вирішить заповнити ці порожні поля, він зможе встановити конкретні значення для них, що буде відображати актуальну інформацію або дані, необхідні для виконання завдання..

3.6 Публікація на хмарній платформі

Опісля тестування системи для публікації використовуються сервіси Azure та вбудовані можливості IDE та СКБД відповідно.

Для публікації бази даних використовується Azure SQL Database. Доступ до ресурсу можна отримати через портал Azure. Для розгортання ресурсу необхідно спочатку створити послідовно підписку, SQL сервер і лише потім SQL Database. Стосовно налаштувань були обрані сервери із рекомендованих Azure, бюджетна підписка та авторизація через логін та пароль адміністратора бази (їх обох було обрано в процесі налаштувань).

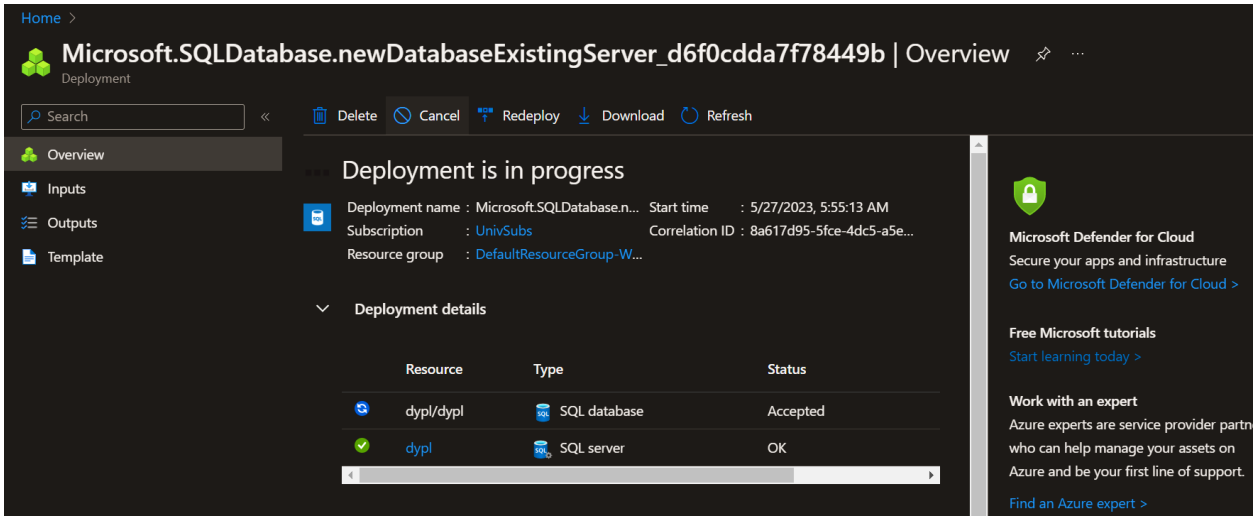


Рисунок 9 – Екран з portal.azure.com

Отримане наступне вікно, після завершення операцій використовуються вбудовані функції MS SQL для публікації в даний контейнер.

Аналогічно з серверною частиною. Перед публікацією змінюємо дані для підключення до бази, додаємо змінні середовища, і публікуємо. Для публікації використовуємо Azure App Service (Windows).

3.7 Розробка користувацького інтерфейсу

При розробці UI було використано фреймворк Angular, який дозволяє створювати більш гнучкі додатки. В даному фреймворку основними частинами проекту є сервіси, що взаємодіють через протокол HTTP з методами API, компоненти, які часто відповідають за сторінки або їх елементи та передають чи отримують інформацію через взаємодію з сервісами, та моделі, які використовуються для представлення сутностей на стороні клієнта.

Наприклад, були розроблені окремі сторінки для відображення списку всіх завдань які належать користувачу та можливості роботи з ними.

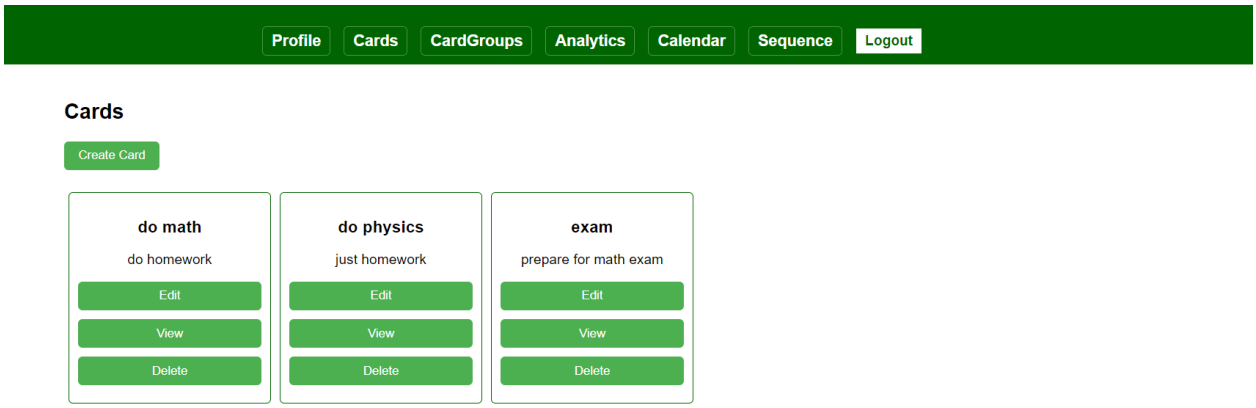


Рисунок 10 – Сторінка із завданнями користувача

Взаємодія з карткою відбувається при натисканні клавіші View, яка відкриває сторінку з інформацією про картку (Додаток Г). На цій сторінці можна додавати файли та передивлятись інформацію про завдання, редагувати її або видалити.

Всі завантажені користувачем файли зберігаються у сховищі Azure, куди їх завантажує API. Згодом файли можна повторно переглядати або видалити.

ВИСНОВКИ

Під час виконання роботи було:

а) Сформульовано проблематику та сферу дослідження;

б) Проаналізовано існуючі рішення, визначені їхні недоліки та переваги та очікувані відмінності з розробленим сервісом;

в) Спроектовано та згенеровано базу даних, для зберігання даних необхідних для функціонування;

г) Розроблена серверна частина та користувацький інтерфейс додатку згідно REST архітектури та проведене тестування, для визначення коректності роботи сервісу

Зокрема робота відповідає поставленим функціональним вимогам. На даний момент наявні можливості додавати, редагувати та видаляти завдання, переглядати їх у різних представленнях; доступна функція поширення інформацією з карток між користувачами та можливість відслідковувати прогрес по виконанню завдань.

Впродовж процесу було досліджено визначений обсяг документації, закріплені навички роботи з фреймворками ASP.NET Core та Entity Framework Core, а також з системою керування базами даних MS SQL.

В підсумку було створено функціонуючий додаток на основі створених моделей, який реалізовує можливості дошки оголошень та може допомогти студентам в організації онлайн-ресурсів.

ДЖЕРЕЛА

1. Atlassian Trello [Електронний ресурс] – Режим доступу до ресурсу: https://trello.com/uk

2. The Trello Tech Stack [Електронний ресурс] – Режим доступу до ресурсу:

https://blog.trello.com/the-trello-tech-stack

3. Asana [Електронний ресурс] – Режим доступу до ресурсу: https://asana.com/guide/help/views/boards

4. Asana Developers [Електронний ресурс] – Режим доступу до ресурсу: https://asana.com/developers

5. Dan Koeppel How to Use Google Classroom Like a Pro / Dan Koeppel [Електронний ресурс] – Режим доступу до ресурсу: https://www.nytimes.com/wirecutter/blog/use-google-classroom-like-a-pro/

6. Introducing the Google for Education Teaching Center [Електронний ресурс] – Режим доступу до ресурсу: https://www.blog.google/outreach-initiatives/education/introducing-google-for-education-teaching-center/

7. 20 Tips for Google Classroom Success [Електронний ресурс] – Режим доступу до ресурсу: https://shakeuplearning.com/blog/20-tips-for-google-classroom-success/

8. Google Classroom: pros and cons of the ultimate learning management system] [Електронний ресурс – Режим доступу до ресурсу: https://www.neolms.com/info/google-classroom-pros-and-cons-of-the-ultimate-learning-management-system/

9. ClickUp [Електронний ресурс] – Режим доступу до ресурсу: https://clickup.com/

10. Roadmap | Moodle developer resourses [Електронний ресурс] – Режим доступу до ресурсу: https://moodledev.io/general/community/roadmap

11. Moodle Docs [Електронний ресурс] – Режим доступу до ресурсу: moodle.org.

12. Visual Studio documentation [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/visualstudio/windows/?view=vs-2022

13. ASP.NET documentation [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/aspnet/core/?view=aspnetcore-7.0

14. Architect Modern Web Applications with ASP.NET Core and Azure [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/

15. What is Entity Framework? [Електронний ресурс] – Режим доступу до ресурсу: https://www.entityframeworktutorial.net/what-is-entityframework.aspx

16. Entity Framework Core Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/ef/core/

17. Microsoft SQL documentation [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/sql/?view=sql-server-ver16

18. SQL Server Features [Електронний ресурс] – Режим доступу до ресурсу: https://www.sqlservercentral.com/features

19. Alexander S. Gills What is REST API (RESTful API)? / Alexander S. Gills [Електронний ресурс] – Режим доступу до ресурсу: https://www.techtarget.com/searchapparchitecture/definition/RESTful-API

20. Richardson, L. RESTful Web Services / Richardson, L., Ruby, S. , 2007. – 422 с.

21. JSON Web Tokens [Електронний ресурс] – Режим доступу до ресурсу: https://jwt.io/.

22. BCrypt.Net[Електронний ресурс] – Режим доступу до ресурсу: https://github.com/BcryptNet/bcrypt.net.

23. Microsoft Azure[Електронний ресурс] – Режим доступу до ресурсу: https://azure.microsoft.com/

24. App Service documentation [Електронний ресурс] – Режим доступу до ресурсу:: https://docs.microsoft.com/azure/app-service/

25. Azure SQL Database documentation [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/azure/azure-sql/database/?view=azuresql

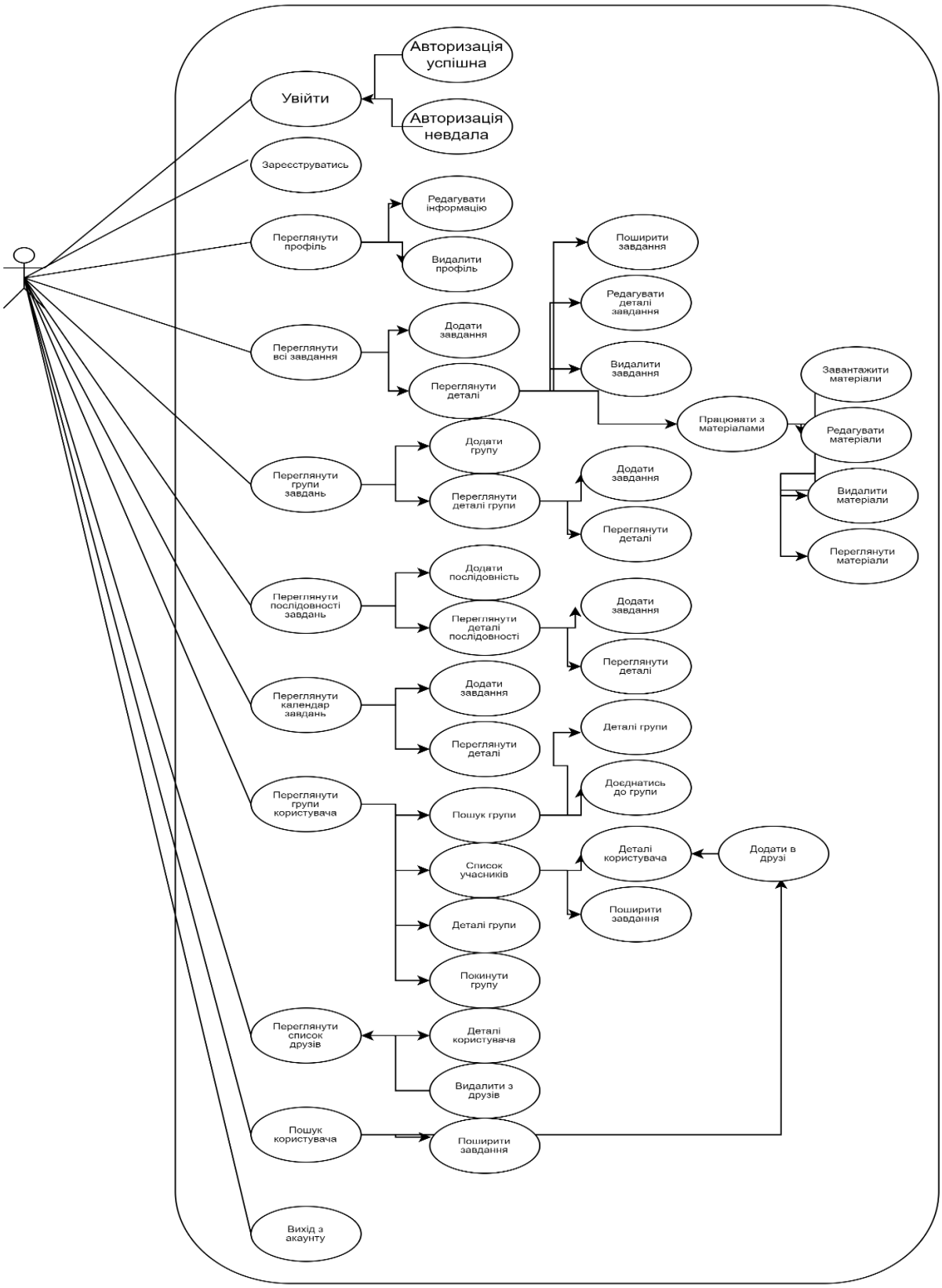
26. Azure Cosmos DB documentation [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/azure/cosmos-db/

27. Azure DevOps documentation [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/azure/devops/

ДОДАТКИ

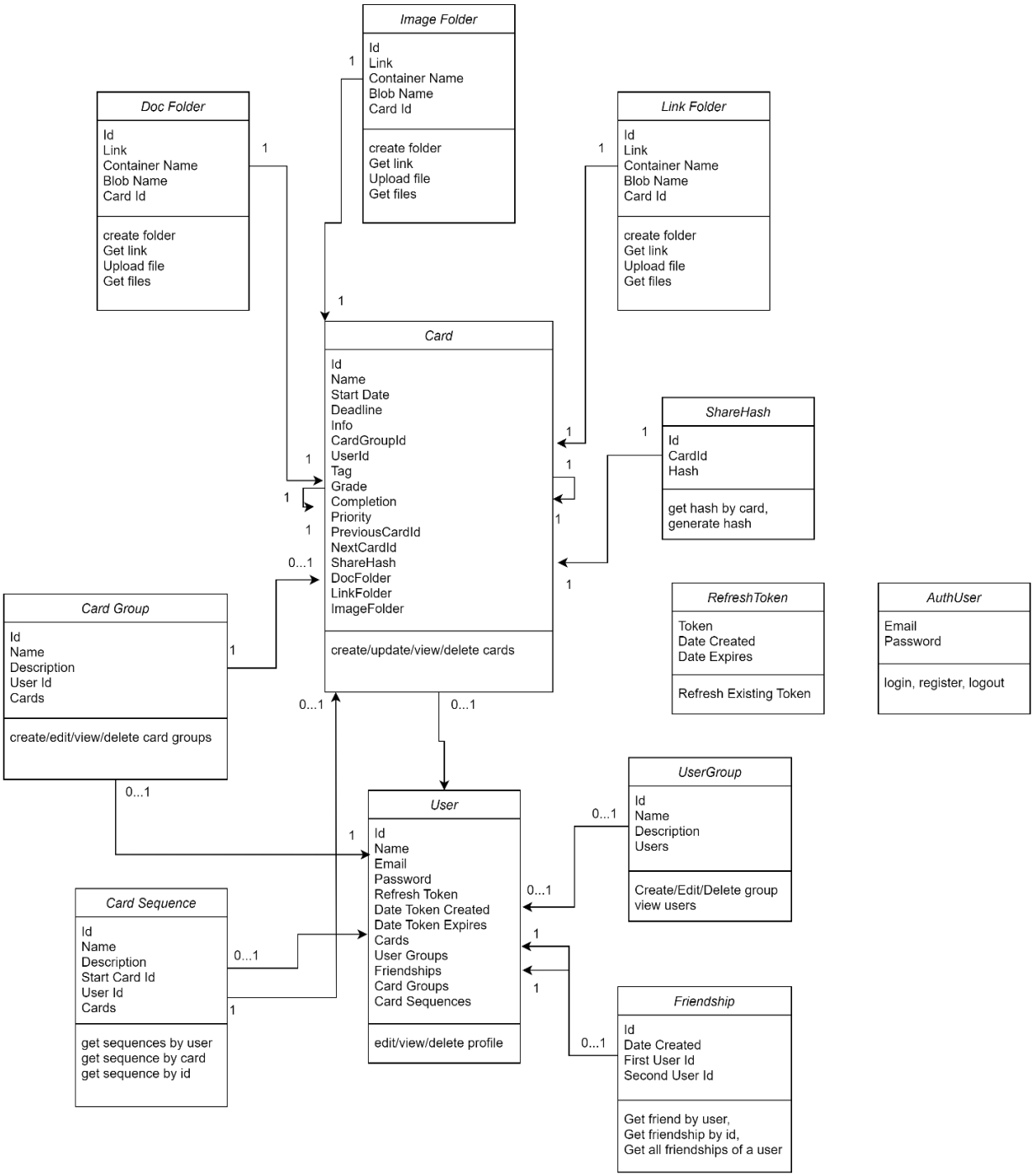
ДОДАТОК А

Діаграма UML use case



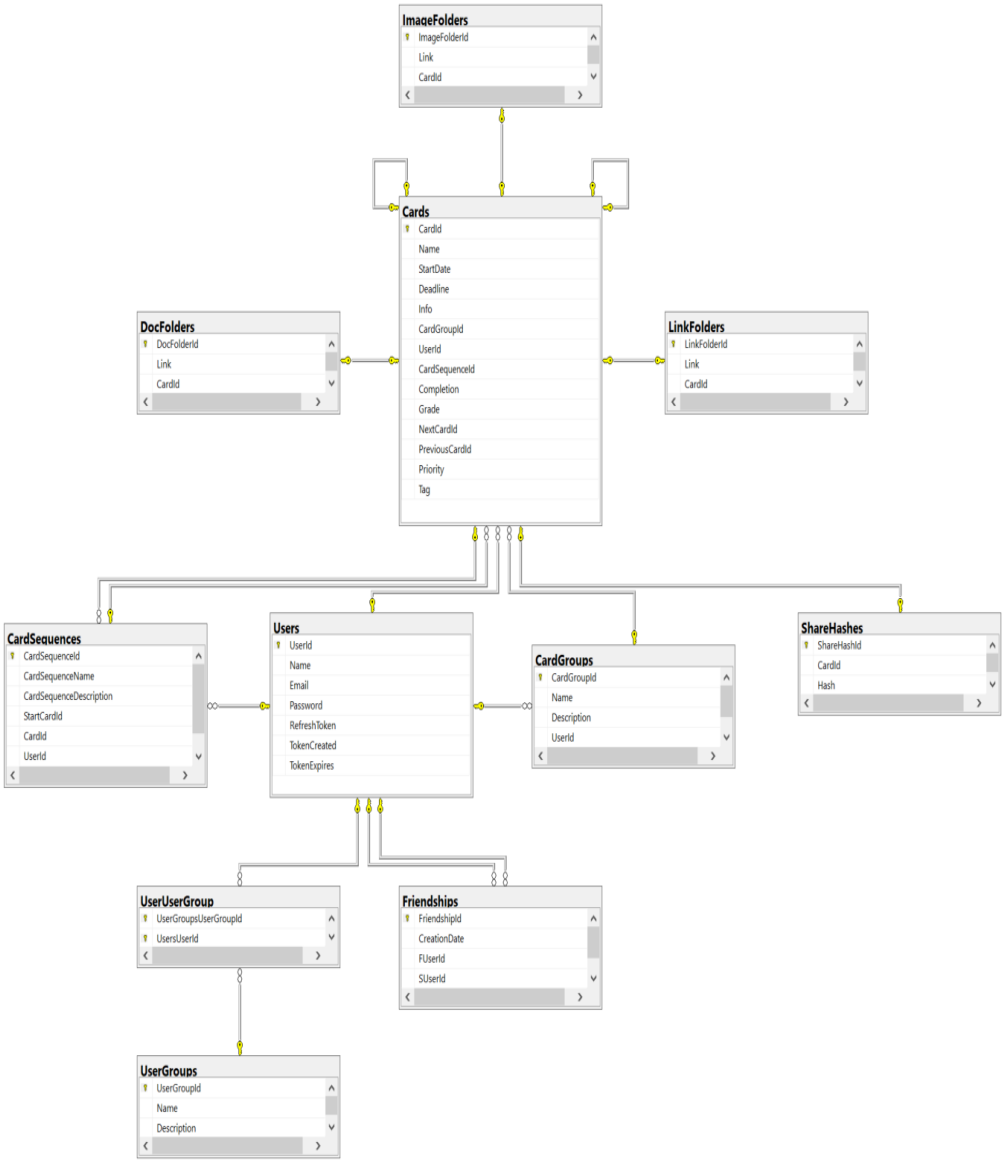
ДОДАТОК Б

Діаграма класів



ДОДАТОК В

Діаграма бази даних



ДОДАТОК Г

Зображення сторінки для роботи із завданням