МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД УНІВЕРСИТЕТ «КРОК»

Фаховий коледж Університету «КРОК»

ДИПЛОМНА РОБОТА

За темою:

«Розробка веб-застосунку розумного розкладу занять»

студент 4 курсу групи ІПЗ-20к/2	Керівник дипломної роботи
	Асистент кафедри комп'ютерних наук
	(посада керівника)
Архипенков Владислав Валерійович	Головань Володимир Володимирович
(прізвище, ім'я, та по-батькові студента)	(прізвище, ім'я, та по-батькові керівника)
	До захисту
	(резолюція «До захисту»)
(підпис студента)	11.06 2024 (дата)
	(підпис викладача)

Київ, 2024 рік

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ ТА ТЕРМІНІВ

IDE (Integrated Development Environment) - комплексне програмне рішення для розробки програмного забезпечення.

SQL (Structured Query Language) - декларативна мова програмування для взаємодії користувача з базами даних.

NOSQL (Not Only SQL) - база даних, яка забезпечує механізм зберігання та видобування даних відмінний від підходу таблиць-відношень в реляційних базах даних.

AI (Artificial Intelligence) - Штучний інтелект.

API (application programming interface) - це набір чітко визначених методів для взаємодії різних компонентів.

REST (Representational State Transfer) - підхід до архітектури мережевих протоколів, які надають доступ до інформаційних ресурсів.

JSON (JavaScript Object Notation) - це текстовий формат обміну даними між комп'ютерами.

HTML (HyperText Markup Language) - стандартизована мова розмітки документів для перегляду вебсторінок у браузері.

DOM (Document Object Model) - специфікація прикладного програмного інтерфейсу для роботи зі структурованими документами.

JIT компіляція (Just-in-time compilation) - це технологія збільшення продуктивності програмних систем, що виконують програмний код, шляхом трансляції байт-коду в машинний код безпосередньо під час роботи програми.

PaaS (Platform as a service) - модель надання хмарних обчислень, при якій споживач отримує доступ до використання інформаційно-технологічних платформ.

CSS (Cascading Style Sheets) - це спеціальна мова стилю сторінок, що використовується для опису їхнього зовнішнього вигляду.

ACID (Atomicity, Consistency, Isolation, Durability) – це набір властивостей, що гарантують надійну роботу транзакцій бази даних: атомарність, узгодженість, ізольованість, довговічність.

JWT (JSON Web Token) - це стандарт токена доступу на основі JSON, використовується для передачі даних для аутентифікації в клієнт-серверних програмах.

OBO (On-Behalf-Of) - описує сценарій веб-API, використовуючи посвідчення, відмінне від власного виклику іншого веб-API.

OIDC (OpenID Connect) - це протокол автентифікації, побудований поверх OAuth 2.0, який дозволяє користувачам автентифікуватись.

ORM (Object-relational mapping) - технологія програмування, яка зв'язує бази даних з концепціями об'єктно-орієнтованих мов програмування.

Azure AD (Azure Active Directory) - хмарне рішення для ідентифікації та керування доступом, яке захищає ваші дані.

HTTP (HyperText Transfer Protocol) - протокол передачі даних, що використовується в комп'ютерних мережах.

ВСТУП

Актуальність проекту – В умовах швидкого розвитку інформаційних технологій та збільшення потреби в оперативному доступі до навчальних ресурсів, створення зручного та функціонального веб-додатку для управління розкладом занять стає не просто корисною ініціативою, а й необхідністю. Це особливо важливо для університетів, які прагнуть надати своїм студентам та викладачам сучасні та ефективні інструменти для планування та організації навчального процесу. Розглянемо докладніше фактори, що визначають актуальність розробки такого додатку. Традиційні методи інформування, такі як паперові розклади та оголошення на інформаційних стендах, часто виявляються недостатньо зручними та оперативними в умовах швидкозмінного графіка.

З розвитком цифрових технологій та поширенням інтернету, вимоги до доступності та актуальності інформації значно зросли. Сучасні студенти очікують, що інформація про розклад буде завжди доступна у реальному часі та зручна для сприйняття.

У цьому контексті веб-програми надають унікальні можливості для покращення навчального процесу.

З урахуванням збільшення кількості онлайн занять та гібридного навчання, інтеграція із системами планування онлайн зустрічей дозволяє студентам легко відстежувати всі віртуальні заняття, заплановані на певний час. Це особливо актуально в умовах повномасштабної війни.

Актуальність розробки веб-застосунку також обумовлена такими факторами:

1. Зручність та доступність інформації:

   У сучасному освітньому процесі розклад занять часто зазнає змін через різні чинники: адміністративні коригування, заміни викладачів, зміни у графіку заходів. Швидка та точна актуалізація розкладу через веб-додаток забезпечує

своєчасне інформування студентів та викладачів, знижуючи ризик пропуску занять та непорозумінь. Застосунок дозволяє студентам отримувати актуальну інформацію про розклад занять у будь-який час та з будь-якого місця. Це особливо важливо в умовах постійних змін розкладу та необхідності оперативного доступу до даних.

2. Екосистема університету:

   Університет "КРОК" вже активно використовує технології Microsoft у своїй екосистемі, що робить інтеграцію з існуючими сервісами та обліковими записами студентів природним та ефективним кроком. Це дозволяє використовувати вже наявні ресурси та покращити їх функціональність. Це спрощує процес авторизації та дозволяє використовувати додаткові функції, такі як персоналізовані нотатки та синхронізація з календарями.

3. Персоналізація та додаткові функції:

   Завдяки інтеграції з обліковими записами Microsoft, студенти можуть отримувати персональні дані, такі як індивідуальні нотатки під парами, інформація про заплановані онлайн зустрічі, а також візуалізація розташування кабінетів. Це значно покращує сприйняття інформації та робить процес планування навчального часу ефективнішим.

4. Інтерактивність та взаємодія:

   Можливість залишати нотатки під парами та ділитися ними з іншими студентами з групи сприяє активній взаємодії та обміну інформацією. Це допомагає студентам краще готуватися до занять та обмінюватися важливими зауваженнями та спостереженнями.

5. Інтерактивність та взаємодія:

   Можливість залишати нотатки під парами та ділитися ними з іншими студентами з групи сприяє активній взаємодії та обміну інформацією. Це допомагає студентам краще готуватися до занять та обмінюватися важливими зауваженнями та спостереженнями.

6. Адаптація до зовнішніх умов:

В умовах поточних реалій, таких як графіки відключення електрики в Україні важливо мати актуальну інформацію про наявність світла в певні проміжки часу. Синхронізація програми API з графіками відключення світла дозволяє студентам заздалегідь планувати свій навчальний час і бути готовими до можливих перебоїв.

7. Підтримка дистанційного та гібридного навчання:

   Пандемія COVID-19 та повномасшабна війна прискорила перехід до дистанційного та гібридного навчання. У таких умовах веб-додаток, що відображає онлайн зустрічі, заплановані заняття та посилання на віртуальні класи, стає незамінним інструментом для студентів та викладачів. Воно забезпечує доступність навчального процесу незалежно від фізичного розташування учасників.

8. Безпека даних:

   Інтеграція з обліковими записами Microsoft забезпечує високий рівень безпеки та захисту даних, що важливо для запобігання несанкціонованому доступу та витоку інформації. Студенти з інших груп не можуть отримати доступ до нотаток з інших. Ніхто, крім самого користувача, не може бачити його онлайн-зустрічі. І лише авторизовані користувачі можуть отримати доступ до отримання навігаційних карток по будівлі університету.

Таким чином, розробка веб-додатку для перегляду розкладу занять не тільки відповідає сучасним вимогам до управління навчальним процесом, а й надає додаткові можливості для студентів, покращуючи їхній навчальний досвід та підвищуючи зручність та ефективність використання освітніх ресурсів університету "КРОК". Розробка програми для перегляду розкладу занять є важливим кроком до модернізації навчального процесу.

Вона відповідає сучасним вимогам до оперативності, зручності та безпеки доступу до інформації, сприяє підвищенню якості освіти та задоволеності студентів та викладачів, що відповідає на безліч викликів сучасного освітнього

процесу та пропонує значні переваги для всіх учасників навчального процесу в університеті.

Завдання проекту –

Для досягнення мети розробки веб-застосунку для студентів навчального закладу ставляться такі завдання:

1. Проектування системи:
1.1. Проектування архітектури системи, включаючи фронтенд та бекенд компоненти.
1.2. Розробка схеми бази даних для зберігання інформації про нотатки, світловідключення та користувачів.
2. Розробка user-friendly інтерфейсу:
2.1. Створення зручного та інтуїтивно зрозумілого інтерфейсу користувача за допомогою React.
2.2. Забезпечує адаптивний дизайн, щоб додаток коректно відображався на різних пристроях (комп'ютери, планшети, смартфони).
2.3. Розробка основних компонентів інтерфейсу: перегляд розкладу, додавання та відображення нотаток, відображення онлайн зустрічей та карт кабінетів.
3. Додавання функціоналу авторизації та аутентифікації:
3.1. Інтеграція з обліковими записами Microsoft для забезпечення безпечного входу користувачів до системи.
3.2. Реалізація функціоналу реєстрації та аутентифікації студентів та викладачів.
3.3. Забезпечування доступу до персоналізованих функцій програми для авторизованих користувачів.
4. Управління розкладом та нотатками:
4.1. Розробка функціоналу для перегляду та оновлення розкладу занять.

4.2. Реалізація можливості залишення нотаток під парами, які будуть помітні іншим студентам з тієї ж групи.

4.3. Забезпечування функцією редагування та видалення нотаток.

5. Інтеграція з онлайн нарадами та картами кабінетів:

5.1. Відображення всіх запланованих зустрічей, синхронізованих з навчальним розкладом.

5.2. Реалізація функціоналу відображення розташування кабінетів з візуалізацією колії по поверху, включаючи схеми та зображення.

6. Синхронізація із зовнішніми сервісами:

6.1. Розробка функціоналу синхронізації з графіками відключення світла по Україні для відображення інформації про наявність світла у певний проміжок часу.

6.2. Інтеграція із зовнішніми API та джерелами даних для забезпечення актуальності інформації.

7. Розробка та налаштування бекенду:

7.1. Створення та налаштування API на платформі ASP.NET Core 8 для взаємодії з фронтендом та базою даних.

7.2. Реалізація основних функціональних модулів, таких як управління розкладом, нотатками та онлайн зустрічами.

7.3. Забезпечення безпеки API та даних користувачів.

8. Тестування та налагодження:

8.1. Проведення функціонального тестування для перевірки працездатності всіх модулів програми.

8.2. Використання юніт-тестування для забезпечення коректної роботи системи в цілому.

8.3. Налагодження та виправлення виявлених помилок, а також оптимізація продуктивності програми.

9. Розгортання та впровадження:

9.1. Підготовка інфраструктури для розгортання API на хмарній платформі Heroku.

9.2. Розгортання фронтенду на хмарній платформі Github Pages.

10.Документація та підтримка:

10.1. Розробка документації для користувачів API з стандартами OAS 3.0.

10.2. Забезпечення технічної підтримки та регулярних оновлень програми.

Об’єкт дослідження проекту – Об'єктом дослідження у цій дипломній роботі є інформаційна система управління навчальним процесом з фокусом на розробку веб-додатка для перегляду розкладу занять та надання додаткових функцій для студентів. Ця програма має інтегруватися з існуючою екосистемою університету та забезпечувати зручні інструменти для організації навчального процесу.

Ці аспекти дозволяють розробити та впровадити ефективний та зручний веб-додаток для управління розкладом занять в університеті, що підвищує задоволеність студентів та викладачів, покращує організацію навчального процесу та інтеграцію з цифровою екосистемою університету..

Практичне значення одержаних результатів –

Розробка веб-додатків для керування розкладом занять та додатковими функціями для студентів університету обіцяє значні вигоди для всіх учасників навчального процесу.

Насамперед, це суттєве поліпшення організації навчального процесу. Студенти та викладачі отримують можливість швидко та легко отримувати доступ до актуального розкладу занять, що дозволяє їм ефективно планувати свої заняття та інші активності. Завдяки додатковим функціям, таким як можливість залишати нотатки під парами та перегляд онлайн зустрічей, студенти можуть краще організовувати свою навчальну діяльність та співпрацювати один з одним у навчальних проектах.

Програма також забезпечує зручність використання для всіх користувачів. Інтуїтивно зрозумілий інтерфейс робить його доступним навіть менш досвідченим користувачам, а адаптивний дизайн гарантує коректне

відображення на всіх типах пристроїв, що дозволяє використовувати його в будь-який час і в будь-якому місці.

Ефективне управління розкладом та обмін інформацією через додаток знижує навантаження на адміністрацію університету, звільняючи їх від рутинних завдань та дозволяючи зосередитись на важливіших аспектах управління навчальним закладом. Інтеграція із зовнішніми сервісами, такими як графіки відключення світла, збагачує функціонал програми та робить його більш інформативним для користувачів.

У результаті розробка веб-додатку для управління навчальним процесом в університеті не тільки підвищує зручність та ефективність для студентів та викладачів.

РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМИ

1.1. Постановка задачі

У сучасній університетській освіті існує низка проблем, пов'язаних з організацією та управлінням навчальним процесом. Однією з таких проблем є неефективне використання ресурсів через неоптимальну організацію розкладу занять та недостатню інформаційну підтримку студентів та викладачів.

Ефективне управління навчальним процесом є наріжним каменем для забезпечення якісної освіти та досягнення успіху студентів. Невдале планування та організація розкладу занять можуть призвести до перетину графіків, нестачі часу для підготовки, а також створення додаткових стресових ситуацій для всіх учасників освітнього процесу.

Однією з головних проблем, з якими стикаються студенти, є незручність доступу до актуального розкладу занять та додаткової інформації про заняття. Традиційні методи розповсюдження розкладу, такі як паперові буклети або розкладні стенди, можуть бути незручними і не завжди забезпечувати актуальну інформацію. Важливо відзначити, що наявність окремої вкладки з онлайн-розкладом на сайті є кроком уперед у напрямку покращення доступності інформації про розклад занять. Однак, на думку студентів, цей розклад може бути незручним у використанні та важким для швидкого доступу через низку факторів.

Насамперед, інтерфейс онлайн-розкладу може бути неінтуїтивним, що ускладнює навігацію та пошук необхідної інформації. Це може включати незручне розташування елементів, складну систему фільтрації або відсутність функцій пошуку.

Таким чином, незважаючи на наявність онлайн-розкладу на сайті, студенти можуть відчувати труднощі при його використанні через незручний інтерфейс та неоптимальне відображення інформації.

Це може призвести до перепусток занять, недорозуміння часу та місця проведення занять, а також до недостатньої підготовки до них.

Для викладачів також існує проблема незручності у плануванні та організації навчального процесу. Вони можуть зіткнутися з труднощами у доступі до інформації про розклад занять та інших актуальних даних, а також у комунікації з колегами та студентами.

У зв'язку з переліченими вище проблемами виникає необхідність у розробці інформаційної системи, яка б забезпечила ефективне управління навчальним процесом, забезпечуючи студентам та викладачам зручний доступ до актуальної інформації про розклад занять та інші аспекти навчального процесу. Така система має бути інтуїтивно зрозумілою, легкою у використанні та гнучкою для адаптації до різних потреб університету.

1.2. Обґрунтування мети рішення поставленої задачі

Метою розробки зручної та ефективної інформаційної системи для управління розкладом занять та додатковими функціями в університеті є прагнення покращити якість освіти та оптимізацію навчального процесу. Це обґрунтовується кількома важливими аспектами.

По-перше, підвищення доступності інформації стає особливо важливим у сучасній освіті. Студенти та викладачі потребують швидкого та зручного доступу до актуального розкладу занять, додаткових матеріалів та важливих подій. Інформаційна система, що забезпечує такий доступ, зробить навчальний процес більш гнучким та ефективним.

По-друге, покращення організації навчального процесу сприяє більш ефективному використанню часу та ресурсів. Студенти зможуть планувати свої заняття та інші обов'язки більш усвідомлено, уникаючи конфліктів у розкладі та знижуючи рівень стресу. Викладачі також отримають інструменти для більш ефективного ведення занять та координації своєї роботи.

Третій аспект пов'язаний із підвищенням ефективності взаємодії між студентами та викладачами. Обмін нотатками та інформацією про зустрічі сприяє більш активній комунікації, співпраці та обміну знаннями. Це важливо не лише для успішного засвоєння навчального матеріалу, а й для розвитку навичок роботи в колективі, які є важливими у професійній діяльності.

Четвертий аспект стосується зниження навантаження на адміністрацію університету. Автоматизація процесів управління розкладом та обміном інформацією знизить кількість рутинних операцій, звільнивши час та ресурси для більш стратегічно важливих завдань.

Нарешті, задоволеність користувачів відіграє ключову роль успішної реалізації інформаційної системи. Створення зручного інтерфейсу, який легко сприйматиметься і використовуватиметься всіма учасниками навчального процесу, сприяє позитивному сприйняттю системи та її активному використанню. Це, у свою чергу, допоможе досягти поставлених цілей у галузі освіти та управління університетом.

РОЗДІЛ 2. МЕТОДИ ТА ЗАСОБИ РЕАЛІЗАЦІЇ WEB-ЗАСТОСУНКУ

2.1. Вибір програмного забезпечення для програмування

Для розробки програми були використані елементи для зручного планування розробки та програмування, а саме:

JetBrains Rider - це потужний засіб розробника, який можна використовувати для завершення всього циклу розробки. Це комплексне інтегроване середовище розробки (IDE), яке можна використовувати для написання, редагування, налагодження та складання коду, а потім для розгортання програми. Крім редагування та налагодження коду, JetBrains Rider включає компілятори, засоби завершення коду, систему керування версіями, розширення та багато інших функцій для покращення кожного етапу процесу розробки програмного забезпечення.

Реалізація: допомагав у написанні коду програми. Дозволяв складали різні модулі програми, встановлювали залежності, відлагоджував код та тестували його в середовищі. Використовуючи багатофункціональний редактор та інші інструменти, забезпечував продуктивну розробку проекту.

JetBrains WebStorm - це потужне інтегроване середовище розробки (IDE) для веб-розробки, розроблене компанією JetBrains. Вона надає розробникам усі необхідні інструменти для створення сучасних веб-додатків на базі HTML, CSS та JavaScript, а також для роботи з різними фреймворками, такими як React, Angular, Vue.js та іншими.

WebStorm володіє широким набором функцій, включаючи автодоповнення коду, інтегровані інструменти для налагодження та тестування, систему контролю версій, інтеграцію з різними інструментами збирання та плагіни для розширення функціональності. Завдяки своїй багатофункціональності та зручності використання, WebStorm став популярним вибором серед веб-розробників.

Реалізація: Під час розробки веб-програми я активно використовував WebStorm як основне середовище розробки. Я оцінив його інтуїтивно зрозумілий інтерфейс, високу продуктивність та широкий набір інструментів, які значно спростили процес написання коду, налагодження та тестування програми. Завдяки функціональності WebStorm я міг ефективно працювати над проектом і досягати поставленої мети розробки.

JetBrains DataGrip - це інтегроване середовище розробки (IDE), спеціально призначене для роботи з базами даних. Розроблена компанією JetBrains, ця IDE надає розробникам усі необхідні інструменти для управління та маніпулювання даними у різних типах баз даних, включаючи SQL, NoSQL і навіть хмарні послуги зберігання даних.

DataGrip володіє широким набором функцій, які забезпечують зручність та ефективність роботи з базами даних. Серед них автодоповнення коду SQL, інструменти для візуального аналізу даних, підтримка різних SQL-діалектів, можливість роботи з кількома підключеннями одночасно, інтеграція із системами контролю версій та багато іншого.

Реалізація: Під час розробки веб-програми я використовував DataGrip для роботи з базою даних. Ця IDE допомогла мені ефективно створювати та змінювати структуру бази даних, виконувати запити SQL, аналізувати дані та багато іншого. Завдяки зручному інтерфейсу та широким можливостям DataGrip я міг оперативно реагувати на вимоги проекту та ефективно керувати базою даних.

Github Copilot - це інтелектуальний помічник для розробників, який використовує AI для автоматичного генерування коду. Він пропонує розробникам контекстні рекомендації та шаблони коду прямо в їхньому улюбленому середовищі розробки. Ключовою особливостю є те, що copilot аналізує контекст вашої роботи та пропонує зразки коду або фрагменти коду, які можуть бути корисними для завершення поточного завдання, окрім цього він

може також створювати коментарі та документацію, що пояснюють, як працює певний код або функція.

Користуючись Copilot, він вчится створювати специфічний код для вашого проекту, що може покращити якість та продуктивність вашої роботи.

Реалізація: скористався функціями інтелектуального аналізу коду, автоматичного рефакторингу та доповнення, щоб зробити код більш організованим, ефективним та легко зрозумілим. Він допомагав виявляти помилки та пропонував покращення, що сприяло підвищенню якості програмного продукту.

GitHub - це онлайн-платформа для керування версіями коду та спільної роботи над проектами. За допомогою GitHub розробники можуть зберігати свій програмний код у віддалених репозиторіях, використовувати систему контролю версій Git для відстеження змін, спільно працювати над проектами та обмінюватись змінами з іншими учасниками команди. Інтерфейс GitHub дозволяє зручно переглядати, порівнювати та зливати зміни, відкриваючи можливості для ефективної співпраці та розробки програмного коду.

Реалізація: допомагав регулярно завантажувати код на GitHub та зберігати його в репозиторії. Це надавало зручний спосіб зберігання та резервного копіювання коду, а також доступ до нього з будь-якого пристрою.

Trello - це онлайн-платформа для управління завданнями та проектами. З її допомогою команди можуть організовувати свою роботу, створювати завдання, призначати відповідальних та відстежувати прогрес виконання завдань. Trello надає зручний інтерфейс, що базується на картках, які можна переміщати між різними списками (наприклад, "У роботі", "Готово", "На перевірці"). Це забезпечує прозорість процесу роботи та дозволяє ефективно організовувати завдання.

Реалізація: Під час роботи над проектами я активно використовував Trello для відстеження завдань та управління проектними процесами. Я створив

чотири списки завдань для кожного етапу роботи - Зроблено, У процесі потрібно зробити низький пріоритет. Що мені допомагало відслідковувати прогрес. Це дозволяло мені ефективно планувати свою роботу, керувати часом та бути в курсі поточного стану проекту.

2.2. Вибір архітектури програмного комплексу

Для реалізації поставленої задачі було вирішено використовувати триланкову архітектуру, яка складається з таких компонентів: клієнт, сервер і база даних. Схема даної архітектури зображена на рисунку 2.1.

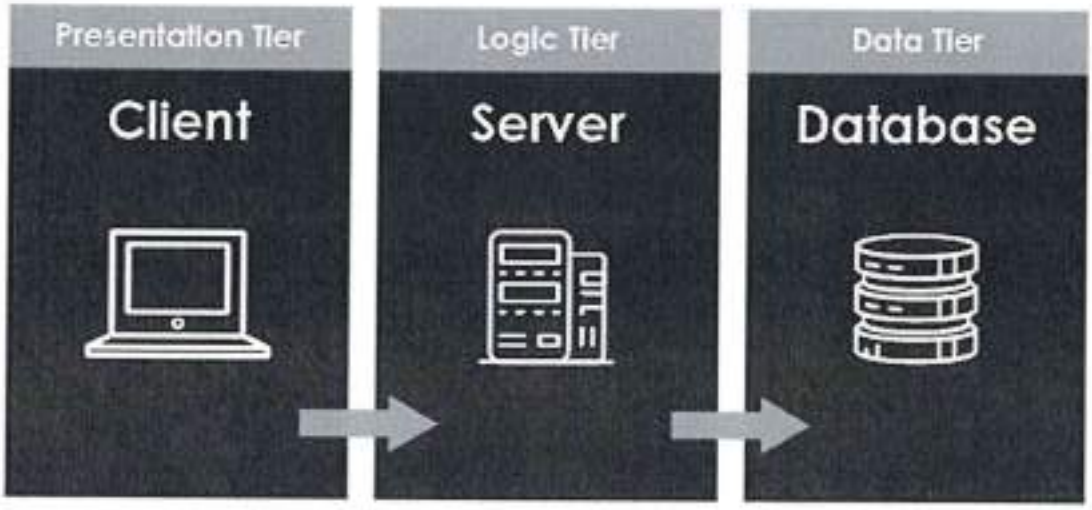

Рисунок 2.1 — Триланкова архітектура програмного комплексу

Головним центром програмного комплексу є сервер. У ньому зосереджена основна бізнес-логіка та логіка доступу до бази даних. За допомогою серверу відбувається ідентифікація користувача для надання індивідуального доступу до програмного застосунку. Сервер є єдиним зв'язком між користувачем та базою даних, щоб унеможливити пошкодження даних та їх використання не за призначенням. Для того, щоб користуватися програмою, потрібно бути авторизованим у системі, тому дана логіка реалізуються на рівні серверу, тому що на рівні користувача можлива підміна прав доступу та інші методи

неконтрольованого доступу до даних. Під час користування програмою, користувач взаємодіє з клієнтським додатком, яким є веб-сайт в даному випадку.

На рівні користувача реалізований інтерфейс, за допомогою якого відбувається налаштування програми та перегляд результатів роботи. Також на користувацькому рівні відбувається попередня обробка даних перед відправленням на сервер і також опрацювання результатів від сервера.

Ще на цьому рівні відбувається перший етап аутентифікації користувача для обмеження неконтрольованого доступу до програми. Важливою задачею рівня бази даних є забезпечення збереження даних, які сервер зберігає для подальшого використання. Також забезпечується цілісність даних за допомогою зовнішніх зв'язків та ключів. На рівні бази даних також можна реалізовувати деяку бізнес-логіку, яка не потребує використання зовнішніх джерел даних окрім самої бази даних та її таблиць.

2.3. Опис архітектури серверу

Для реалізації серверу було використано API, який є інтерфейсом прикладного програмування. Одним з найпоширеніших призначень API є надання набору широко використовуваних функцій, наприклад для малювання вікна чи іконок на екрані. Програмісти використовують переваги API у функціональності, таким чином їм не доводиться розробляти все з нуля. Програмний інтерфейс API є абстрактним поняттям — програмне забезпечення, що пропонує деякий API, часто називають реалізацією даного API. У багатьох випадках API є частиною набору розробки програмного забезпечення, водночас, набір розробки може включати як API, так і інші інструменти/апаратне забезпечення, отже ці два терміни не є взаємозамінювані.

2.3.1. Детальне визначення Web-API

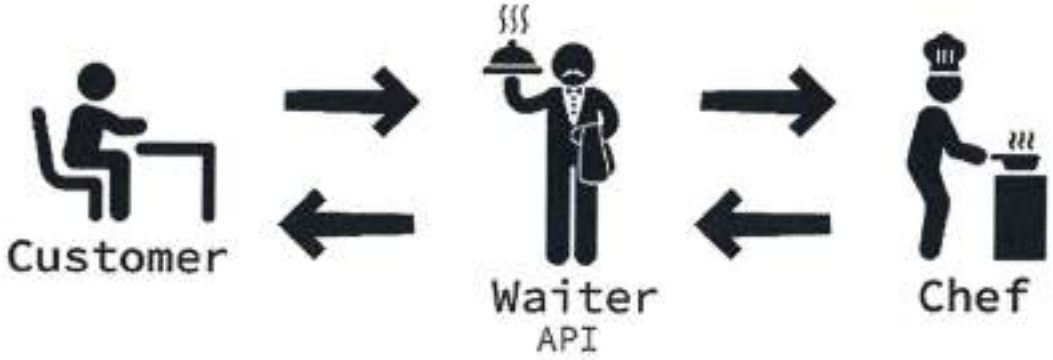

Рисунок 2.2 – Приклад API у реальному житті

API - інтерфейс програмування програми. це набір правил, протоколів та інструментів, які дозволяють різним програмам взаємодіяти одна з одною. Основна ідея полягає в тому, що програми можуть використовувати функції або отримувати доступ до даних інших програм без необхідності знати деталі їхньої внутрішньої реалізації.

API можна порівняти з інтерфейсом користувача, але в контексті програмування. Якщо інтерфейс користувача надає зручний спосіб для людей взаємодіяти з програмою через графічний інтерфейс, то API надає зручний спосіб для програм взаємодіяти між собою.

Як реальний приклад, подумайте про електропостачання у вашому будинку, квартирі чи іншому житлі. Якщо ви хочете використати якийсь прилад у вашому домі, ви вмикаєте його в розетку і він працює. Ви не намагаєтеся підключити його безпосередньо до електромережі - це було б дуже неефективно і, якщо ви не електрик, складно і небезпечно.

Тепер розберемо Web API, це, в свою чергу, програмний інтерфейс, який дозволяє різним програмам або веб-сервісам спілкуватися між собою через Інтернет використовуючи HTTP-протокол.

Розглянемо простий приклад. Припустимо, ви розробляєте магазин електроніки, і у вас є база даних з інформацією про товари. Ви можете створити

Web API, яке надає доступ до цих товарів, таким чином інші сервіси або додатки можуть отримати доступ до списку товарів, їхніх цін, описів тощо.

Web API реалізуються за допомогою HTTP-протоколу, що є стандартом для передачі даних в Інтернеті. Вони приймають HTTP-запити від клієнтів і повертають HTTP-відповіді з результатами запитів. Для зручності розробки та використання Web API можуть підтримувати різні формати обміну даними, такі як JSON або XML. Одні з основних принципів, які дотримуються при розробці Web API, - це принцип REST.

2.3.2. Архітектурний стиль REST API

Що таке REST? Він визначає стандарти для створення легкодоступних, масштабованих і добре структурованих API. Спочатку REST створювали як керівництво для управління взаємодіями в складній мережі, такій як Інтернет. Архітектуру на основі REST можна використовувати для підтримки високопродуктивного і надійного зв'язку в необхідному масштабі. Її можна легко впроваджувати і модифікувати, забезпечуючи прозорість і крос-платформну переносимість будь-якої системи API.

Розробники можуть створювати API з використанням декількох архітектур. API-інтерфейси, що відповідають архітектурному стилю REST, називаються REST API. Веб-служби, що реалізують архітектуру REST, називаються веб-службами RESTful. Як правило, термін RESTful API відноситься до мережевих RESTful API. Однак REST API і RESTful API є взаємозамінними термінами.

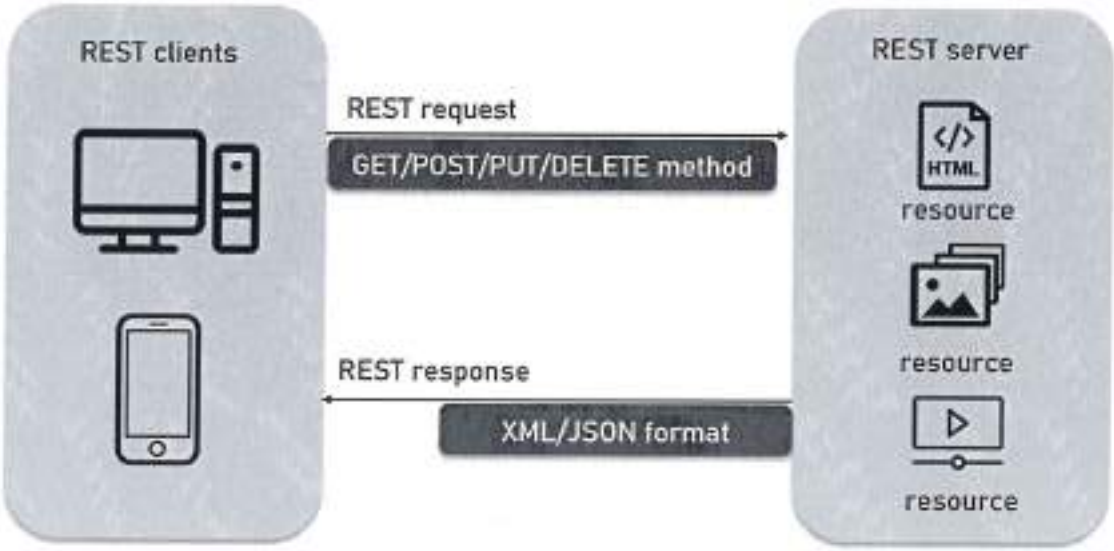

Рисунок 2.3 – Структура REST API

Розглянемо деякі принципи архітектурного стилю REST:

Єдиний інтерфейс – є конструктивною основою будь-якого веб-сервісу RESTful. Це свідчить про те, що сервер передає інформацію у стандартному форматі. Відформатований ресурс у REST називається уявленням. Цей формат може відрізнятись від внутрішнього представлення ресурсу в серверній програмі. Наприклад, сервер може зберігати дані у вигляді тексту, але надсилати їх у форматі подання JSON.

Відсутність збереження стану - В архітектурі REST відсутність збереження стану відноситься до методу зв'язку, при якому сервер виконує кожен запит клієнта незалежно від усіх попередніх запитів. Клієнти можуть вимагати ресурси у будь-якому порядку, і кожен запит або ізольований від інших запитів, або його стан не зберігається. Це конструктивне обмеження REST API передбачає, що сервер може щоразу повністю зрозуміти та виконати запит.

Багаторівнева система - У багаторівневій системній архітектурі клієнт може підключатися до інших авторизованих посередників між клієнтом і сервером і, як і раніше, отримувати відповіді від сервера. Сервери також можуть передавати запити іншим серверам. Ви можете спроектувати свою веб-службу

RESTful для роботи на кількох серверах із кількома рівнями (безпекою, додатками та бізнес-логікою), які спільно виконують клієнтські запити. Ці рівні залишаються невидимими для клієнта.

2.4. Опис архітектури клієнтського застосунку

Архітектура клієнтської частини веб-додатка визначає структуру, організацію та взаємодію компонентів, що відповідають за інтерфейс та користувацький досвід. У контексті нашого проекту, де ми використовуємо JavaScript з бібліотекою React та стилі Bootstrap, архітектура клієнтської частини є ключовою для забезпечення ефективності, розширюваності та масштабованості додатку. Основною частиною буде React, який дозволяє розбивати інтерфейс на невеликі, самодостатні компоненти, які легко керувати та перевикористовувати. Кожен компонент відповідає за свою частину інтерфейсу та має свій власний стан. Компоненти організовані в ієрархічну структуру, де вищорівневі компоненти управляють станом та передають пропси (властивості) нижчорівневим компонентам. Це дозволяє ефективно керувати даними та станом додатку. Для зручного управління станом додатку може використовуватися патерн контейнер-компонент, де стан зберігається на вищому рівні та передається дочірнім компонентам за допомогою пропсів.

І також клієнтська частина відповідає за маршрутизацію сайту. Для навігації між сторінками та відображення відповідних компонентів за адресою URL використовується маршрутизація. У нашому випадку, може використовуватися бібліотека Hash Router для організації маршрутів, такий вибір був зроблений через деякі проблеми з навігацією платформи GitHub Pages, а саме через те, що навігація вже включає до себе назву репозиторія на сайті GitHub.

Клієнтська частина буде взаємодіяти з серверною частиною за допомогою HTTP запитів, таких як GET, POST, PUT та DELETE. Для цього ми будемо використовувати бібліотеку Axios. Для найкращого оформлення та мобільної

оптимізації ми будемо використовувати Bootstrap, який надає готові компоненти та стилі для створення зручного та привабливого користувацького інтерфейсу. Використання стилів Bootstrap спрощує розробку та забезпечує єдність у вигляді додатку.

Узагальнюючи, архітектура клієнтської частини веб-додатка заснована на принципах розбиття на компоненти, ієрархічної організації, маршрутизації та управління станом, що дозволяє створити ефективний, зручний та масштабований інтерфейс для користувачів.

2.5. Опис інструментів розробки

2.5.1. Мова програмування JavaScript та бібліотека React

У цьому розділі буде розглянуто мову програмування JavaScript, яка в основному використовується для розробки нашої клієнтської частини, а також бібліотека React яка надає шикорий спектр інструментів для розробки чуйних та сучасних веб-застосунків.

Мову JavaScript класифікують як прототипну (підмножина об'єктно орієнтованої), скриптову мову програмування з динамічною типізацією. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування (імперативну та частково функціональну) і деякі відповідні архітектурні властивості,зокрема: динамічна та слабка типізація, автоматичне керування пам'яттю, прототипне наслідування, функції як об'єкти першого класу.

Тож, як працює JavaScript знаходячись у браузері? Коли веб-сторінка завантажується, браузер отримує HTML-код та створює дерево об'єктів документа (DOM), що можна побачити на рис. 2.4. У процесі завантаження та парсингу сторінки браузер також знаходить та виконує JavaScript-код. JavaScript може бути вбудований у HTML-документ за допомогою тега <script>, або завантажений із зовнішнього файлу.

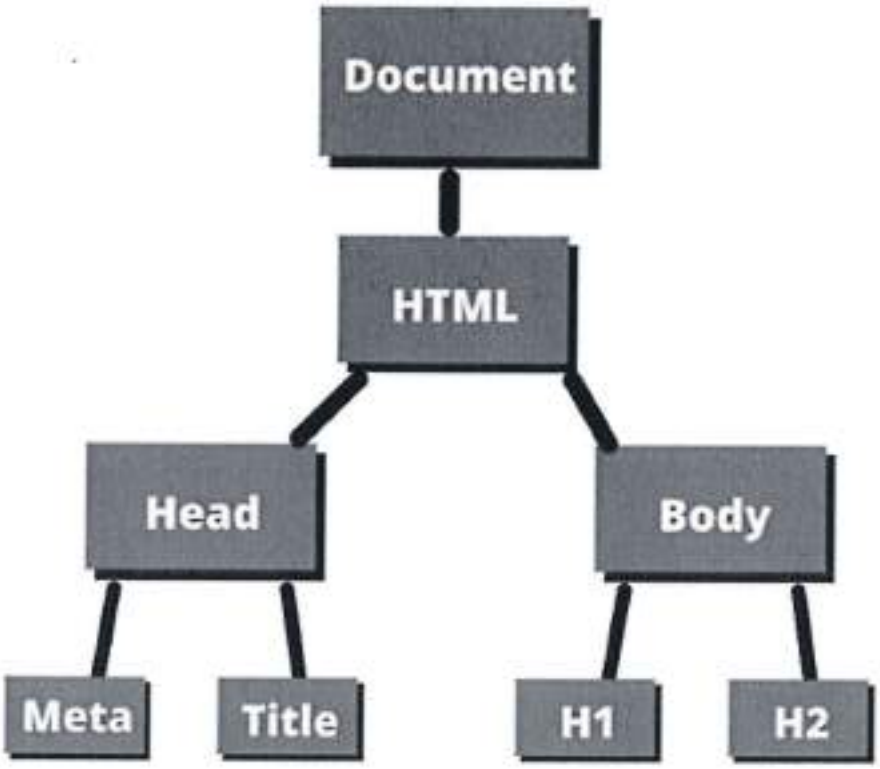

Рисунок 2.4 – DOM-дерево

Першим етапом є парсинг. Двигун JavaScript бере вихідний код і перетворює його на абстрактне синтаксичне дерево (AST). Це дерево є структурованим уявленням коду, де кожен вузол відповідає синтаксичної конструкції мови.

На наступному етапі двигун виконує проміжне уявлення коду, відоме як байт-код. Деякі движки, такі як V8 використовують проміжну компіляцію, де AST компілюється в байт-код, який потім може бути виконаний віртуальною машиною. Віртуальна машина інтерпретує байт-код та виконує відповідні інструкції.

Щоб підвищити продуктивність, сучасні движки JavaScript також використовують Just-In-Time (JIT) компіляцію. JIT-компілятор аналізує байт-код під час виконання і переводить код, що виконується, безпосередньо в машинний код, оптимізуючи його для конкретного процесора. Це дозволяє значно прискорити виконання JavaScript-додатків, оскільки машинний код виконується швидше, ніж байт-код, що інтерпретується.

JavaScript - однопотокова мова, що означає, що він виконує завдання послідовно в одному потоці. Однак для забезпечення асинхронності JavaScript використовує події та зворотні дзвінки (callbacks). Коли виникає асинхронна подія, така як запит до сервера або таймер, вона міститься в чергу подій. Головна петля подій (event loop) постійно перевіряє цю чергу та обробляє події в міру їх надходження, викликаючи відповідні зворотні дзвінки.

Для роботи з асинхронністю також використовуються обіцянки (promises) та async/await. Обіцянки є об'єкти, які можуть бути в одному з трьох станів: очікування, виконання і відхилення. Це дозволяє зручніше працювати з асинхронними операціями, уникаючи вкладеності зворотних викликів. Async/await, надають синтаксичний цукор для роботи з обіцянками, дозволяючи писати асинхронний код, який виглядає як синхронний, що робить його більш читаним та керованим.

JavaScript також взаємодіє з веб-API, що надаються браузером, такими як DOM API, для маніпуляції елементами на сторінці, Fetch API для виконання мережевих запитів та інші API для роботи з графікою, мультимедіа та сховищем даних. Ці API надають інтерфейси для виконання різних завдань від взаємодії з користувачем до керування станом програми.

Далі окремо розглянемо основну бібліотеку, яка буде використовуватись для розробки користувацької частини інтерфейсу. Цією бібліотекою є React, що підтримується та розширюється компанією Meta. React - це популярна JavaScript-бібліотека для створення інтерфейсів користувача. React дозволяє розробникам створювати веб-програми, що складаються з окремих компонентів, які можуть повторно використовуватися та ефективно керуватися. Бібліотека використовується для створення динамічних і високопродуктивних інтерфейсів, завдяки своїй філософії компонентного підходу і віртуальному DOM. React розбиває інтерфейс на невеликі ізольовані компоненти, кожен з яких відповідає за свій власний шматок UI. Це спрощує розробку, тестування та повторне використання коду.

У React для кожного об'єкта справжнього DOM (далі – BDOM) існує відповідний об'єкт VDOM. VDOM – це об'єктне уявлення BDOM, його легковажна копія. VDOM містить ті ж властивості, що й BDOM, але не може безпосередньо впливати на те, що відображається на екрані.

Коли до UI додаються нові елементи, створюється VDOM у вигляді дерева. Кожен елемент є вузлом цього дерева. При зміні стану будь-якого елемента створюється нове дерево. Потім це нове дерево порівнюється (diffed) із старим.

Після цього обчислюється найефективніший метод внесення змін до BDOM. Мета даних обчислень полягає в мінімізації кількості операцій, які здійснюються з BDOM. Тим самим зменшуються накладні витрати, пов'язані з оновленням BDOM.

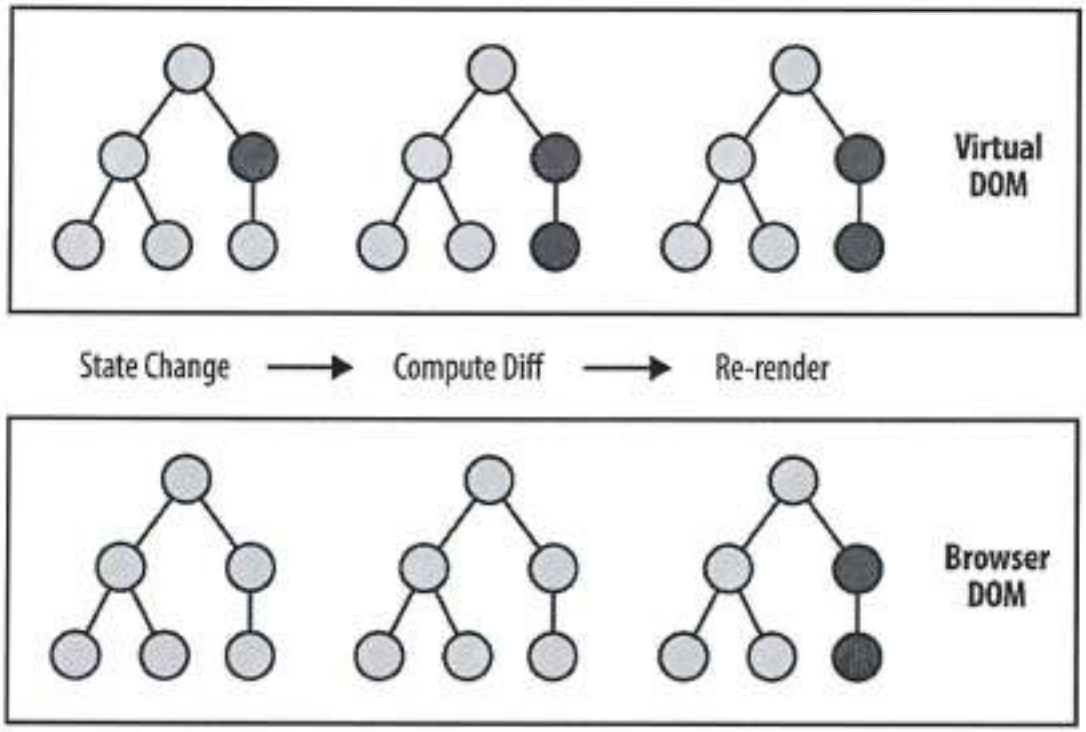

Рисунок 2.5 – Процес узгодження VDOM та BDOM

Але ж як працює механізм узгодження? Узгодження у React – це те, як приймається рішення про повторний рендеринг компонента. У браузерах

перемалювання DOM дерева – дуже важка операція і вимагає багато часу як при монтуванні, так і при розмонтуванні, що, безумовно, впливає на перфоменс, за який ми боремося. Частиною того, що робить React дуже продуктивним, є цей алгоритм.

Ось спрощений варіант його роботи: На початку при запуску програми React з наявних нод-вузлів збирає перше дерево, яке прийнято називати Current tree, і серед Rendering Enviroment на основі пріоритетів малює перше DOM дерево, що ми згодом побачимо як стартову web-сторінку.

Далі, на стадії, коли у відповідь на дію чи подію на нашому сайті відбуваються зміни, будується нове дерево Work-in-progress tree, яке, у свою чергу, відразу порівнюється з Current tree для визначення відмінностей. Ось саме ці відмінності і передаються потім у середу Rendering Enviorement для повторного малювання та оновлення DOM. Тепер Work-in-progress tree перетворюється на Current tree до наступної зміни, де цей цикл повториться.

При вирішенні проблеми трансформації одного дерева в інше команда розробників React Core застосувала алгоритм на основі евристики, а це означає, що алгоритм розв'язання задачі не є гарантовано точним чи оптимальним, але достатньо для вирішення поставленого завдання. Ось це і дозволило значно прискорити розв'язання задачі таким чином, що замість повного обходу вузлів дерева відбувається дуже обмежена кількість порівнянь, що дорівнює кількості нід у дереві. А це дозволяє досягти складності алгоритму O(n) проти $O(n^3)$ за повного обходу.

Цей алгоритм і отримав назву «diffing» або «узгодження» і включає наступні припущення. Щоб оновлення компонентів були передбачуваними і досить швидкими.

Два елементи різних типів виготовлятимуть різні дерева. Заміна кореневого типу div на тип nav у компоненті призведе до розмонтування старого дерева з усіма дочірніми елементами та втратою їхнього стану, а потім

дерево буде змальовано заново. Виконуючи порівняння двох елементів React DOM одного типу React дивиться на атрибути обох, за необхідності оновлює змінені атрибути, наприклад, className.

2.5.2. Мова програмування C#, фреймворки .NET та ASP.NET

У якості мови програмування буде обрано C#. Вона є мовою програмування, яка має багатий набір функцій і можливостей, що дозволить вам ефективно реалізувати потрібну функціональність в програмі. Вона підтримує об'єктно-орієнтований підхід, динамічне зв'язування, делегати, лямбда-вирази та багато інших конструкцій, які полегшують розробку складних застосунків. Також разом с C# буде використовуватись платформа .NET Core 8.

.NET Core - це платформа для розробки та виконання програм, створена компанією Microsoft. Вона надає середовище виконання, бібліотеки та інструменти розробки для створення різних типів програм, включаючи веб-додатки, мобільні програми, консольні програми та багато іншого.

Він відомий своєю високою продуктивністю та швидкодією. Він має оптимізований середовище виконання та конструкції, що дозволяють розробникам створювати ефективні та швидкодіючі додатки. Це особливо важливо, якщо проект потребує обробки великого обсягу даних або високої продуктивності.

В основі .NET Core лежить концепція крос-платформності, що означає, що програми, створені з використанням цієї платформи, можуть працювати на різних операційних системах, таких як Windows, MacOS і Linux. Це досягається завдяки тому, що .NET Core побудований з використанням відкритих стандартів та надає спільні інтерфейси для взаємодії з операційною системою.

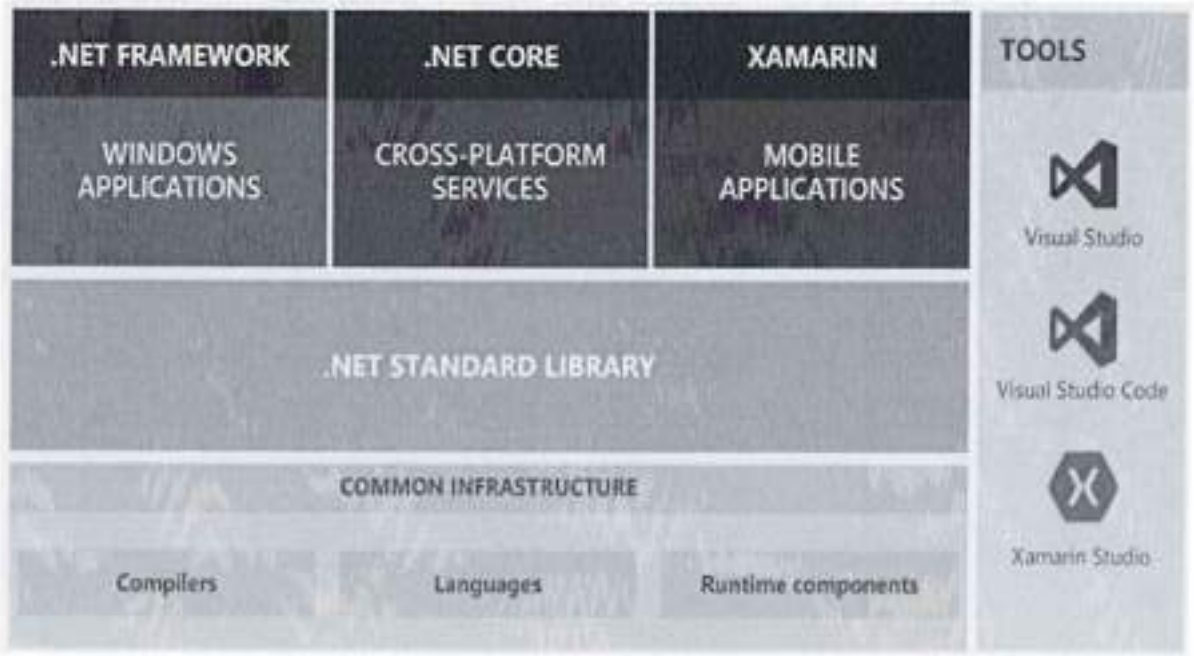

Рисунок 2.6 – Інфраструктура .NET

.NET Core заснований на модульній архітектурі, що дозволяє розробникам вибирати тільки ті компоненти та бібліотеки, які необхідні для їхньої конкретної програми. Це підвищує гнучкість та ефективність розробки, а також зменшує розмір та складність остаточного застосування.

Важливою особливістю .NET Core є його висока продуктивність та масштабованість. Він забезпечує швидке виконання коду завдяки використанню оптимізованої віртуальної машини та JIT-компіляції. Крім того, .NET Core підтримує асинхронне програмування та багатопоточність, що дозволяє створювати швидкі та чуйні програми навіть при високих навантаженнях.

Іншою важливою характеристикою .NET Core є його велика екосистема, яка включає різні інструменти, бібліотеки і фреймворки для розробки додатків. Це дозволяє розробникам створювати високоякісні та інноваційні програми з мінімальними зусиллями та витратами.

В цілому, .NET Core є потужною і гнучкою платформою для розробки сучасних додатків, яка забезпечує високу продуктивність, крос-платформність і багату функціональність. Він активно використовується в різних галузях і

пропонує розробникам широкі можливості для створення інноваційних та ефективних програм.

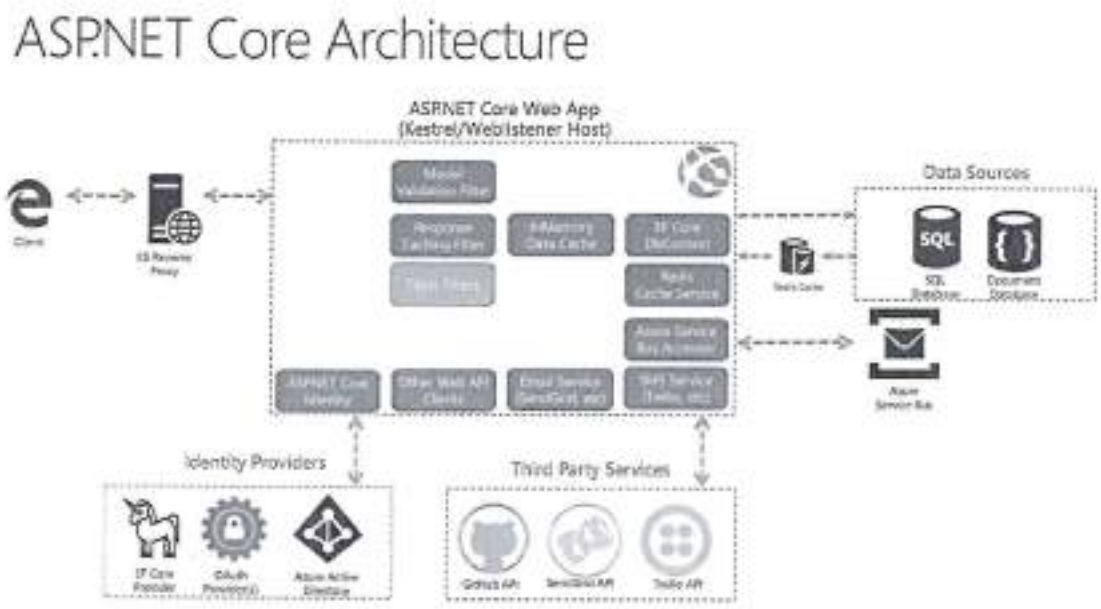

Рисунок 2.7 – Архітектура ASP.NET Core

ASP.NET Core - це фреймворк для розробки веб-застосунків, який є частиною екосистеми .NET Core від Microsoft. Він надає набір інструментів, бібліотек та шаблонів для створення сучасних та масштабованих веб-застосунків, що працюють як на Windows, так і на інших операційних системах.

Основні принципи роботи ASP.NET Core включають високу продуктивність, масштабованість і гнучкість. Він побудований з використанням модульної архітектури, яка дозволяє розробникам вибирати тільки ті компоненти та функціональність, які необхідні для їх застосування. Це спрощує процес розробки та скорочує накладні витрати.

2.5.3. Хмарна платформа Heroku та GitHub Pages

Heroku - це хмарна платформа як сервіс (PaaS), яка надає можливість розгортання, масштабування та управління веб-застосунками. Heroku підтримує широкий спектр технологій та мов програмування, а ті, що не підтримуються, можна використовувати налаштування для розгортання. З її допомогою розробники можуть швидко розгорнути свої програми у хмарі, використовуючи

простий та інтуїтивно зрозумілий інтерфейс керування. Завдяки автоматизованим процесам розгортання та інтеграції, Heroku спрощує життя розробників, дозволяючи їм швидко запускати та масштабувати свої програми без зайвого клопоту. За допомогою студентської підтримки від Heroku, було отримано кошти на баланс аккаунту, задля розробки та тестування застосунків студентами, і завдяки чому була розгорнута серверна частина цього веб-застосунку.

Heroku дозволяє користувачам використовувати аддони для веб-застосунків. Аддони - це сторонні сервіси, які інтегруються з вашим додатком для додавання різних можливостей, таких як бази даних, кешування, моніторинг, черги завдань та багато іншого. Використання аддонів дозволяє розробникам швидко інтегрувати необхідні сервіси без необхідності налаштовувати та керувати ними самостійно.

Moesif – одна з таких аддонів на Heroku, це аналітична платформа для API, яка допомагає компаніям покращити та оптимізувати їх API-стратегії. Вона надає різні інструменти для моніторингу та аналізу трафіку, відстеження продуктивності, виявлення проблем та розуміння використання API кінцевими користувачами.

Moesif надає можливості для моніторингу ключових метрик API, таких як частота запитів, час відповіді, успішні та неуспішні запити, а також детальні дані про користувачів та їх взаємодію з API. Платформа також має можливість аналізу даних для виявлення трендів, проблем та можливостей для покращення застосунку.

Під час розробки веб-програми я інтегрував Moesif для моніторингу та аналізу використання мого API. Це допомогло мені отримати цінний зворотний зв'язок про те, як мої кінцеві користувачі взаємодіють з API, і виявити області для покращення та оптимізації. Завдяки Moesif я міг ефективно відслідковувати продуктивність та надійність мого API, що дозволило мені створити більш

якісний та надійний веб-застосунок. Інтеграція з Heroku значно спростила процес інтеграції Moesif, забезпечуючи швидке та ефективне впровадження аналітичної платформи у мій проект.

GitHub Pages - це безкоштовний сервіс хостингу, що надається GitHub, який дозволяє розміщувати статичні веб-сайти безпосередньо з репозиторіїв на GitHub. Він має простий у використанні інтерфейс і добре інтегрований з системою контролю версій Git, що робить його популярним вибором для розгортання статичних сайтів, документації та інших веб-проектів.

GitHub Pages надає користувачеві можливість розміщувати веб-сторінки безпосередньо в репозиторії GitHub і автоматично розгортати їх на піддомені github.io. Сервіс підтримує різні технології, включаючи HTML, CSS, JavaScript, а також різні фреймворки та бібліотеки, такі як React, Vue.js, Angular та інші.

Під час розробки веб-програми я використовував GitHub Pages для розгортання клієнтської програми з використанням React. Це дозволило мені швидко та зручно розмістити статичні файли програми у репозиторії на GitHub та автоматично розгорнути їх на піддоміні github.io. Завдяки GitHub Pages користувачі могли легко отримати доступ до моєї веб-застосунку без необхідності налаштування окремого хостингу або сервера.

2.5.4. Система керування базами даних PostgreSQL

PostgreSQL (або Postgres) — це потужна, відкрита реляційна система керування базами даних (СКБД), яка має багату функціональність та високу продуктивність. Вона є однією з найпопулярніших СКБД завдяки своїй надійності, масштабованості та підтримці широкого спектра стандартів SQL.

PostgreSQL відповідає більшості стандартів SQL, підтримуючи складні запити, транзакції, підзапити та індекси. Розробники мають можливість створювати власні функції, типи даних, оператори та індекси, що робить систему дуже гнучкою. Підтримка ACID забезпечує високу надійність транзакцій завдяки атомарності, консистентності, ізоляції та надійності. СУБД

підтримує паралельні запити та багатоверсійність, що дозволяє забезпечити конкурентний доступ до даних без блокувань. PostgreSQL може працювати з великими обсягами даних, підтримуючи як вертикальне, так і горизонтальне масштабування. Завдяки асинхронній реплікації, логічній реплікації та кластеризації забезпечується висока доступність і відмовостійкість.

Головна іконка PostgreSQL представляє собою зображення слона, і це не випадково, тому що це символ міцності, надійності та витривалості, адже PostgreSQL відома своєю здатністю обробляти великі обсяги даних та підтримувати високий рівень надійності навіть у найскладніших умовах.

Також, PostgreSQL є проектом з відкритим кодом, що забезпечує доступність та прозорість, а також активну підтримку з боку спільноти розробників. СУБД має клієнтські бібліотеки для багатьох мов програмування, включаючи Python, Java, C#, PHP, Ruby, Node.js та інші. PostgreSQL забезпечує гнучкість в управлінні даними завдяки підтримці складних типів даних, JSON, XML, а також можливості створювати власні типи даних та функції. Система підтримує шифрування даних, аутентифікацію за допомогою паролів, SSL, а також контроль доступу до даних на рівні рядків. Велика спільнота користувачів та розробників, регулярні оновлення та покращення гарантують надійність і розвиток системи.

І звісно, ця база даних підтримується як аддон на Heroku, що дозволяє легко інтегрувати її у застосунок.

Heroku Postgres надає потужні інструменти для моніторингу та аналітики, які допомагають відстежувати продуктивність вашої бази даних. В панелі управління Heroku ви можете переглядати статистику використання ресурсів, наприклад, використання CPU, пам'яті та дискового простору. Також доступні метрики, які показують час виконання запитів, кількість підключень та інші важливі показники. Це дозволяє виявляти потенційні проблеми та оптимізувати роботу вашого додатка.

2.6. Опис інструментів авторизації користувача

2.6.1. Web-API застосунок Microsoft Graph

Microsoft Graph є потужним інструментом для розробників, який дозволяє отримувати доступ до даних і сервісів Microsoft 365. Це єдиний API, що забезпечує інтеграцію з різними продуктами та службами Microsoft, включаючи Azure Active Directory, Outlook, OneDrive, Teams та інші. Завдяки Microsoft Graph, розробники можуть створювати додатки, які забезпечують надійну та безпечну авторизацію користувачів, а також доступ до їхніх даних та ресурсів.

Microsoft Graph дозволяє отримувати доступ до користувацьких даних, таких як профіль, контакти, повідомлення електронної пошти, файли, календарі та інші. Це забезпечує багатий функціонал для створення персоналізованих додатків. За допомогою Microsoft Graph можна інтегрувати функції автентифікації та авторизації користувачів у ваші додатки, використовуючи Azure Active Directory. Це забезпечує високий рівень безпеки та контроль доступу до даних. API дозволяє розробникам отримувати доступ до різних сервісів Microsoft 365 через єдиний кінцевий пункт, що спрощує інтеграцію та зменшує складність коду. Microsoft Graph підтримує сценарії реального часу, такі як отримання сповіщень про зміни в даних, що дозволяє створювати інтерактивні та динамічні додатки. Завдяки підтримці стандартів OAuth 2.0 та OpenID Connect, Microsoft Graph забезпечує безпечну та надійну автентифікацію користувачів.

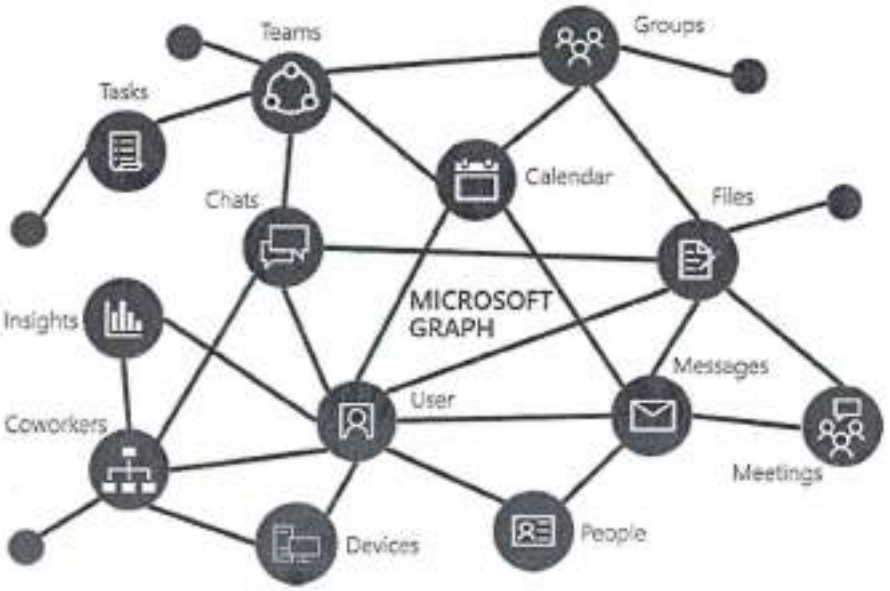

Рисунок 2.8 – Сервіси Microsoft Graph

Технологія Microsoft Graph є надзвичайно актуальною для проекту, оскільки еко-система університету «КРОК» побудована на акаунтах Microsoft. Це означає, що університет використовує різноманітні сервіси Microsoft 365, такі як Outlook, Teams та інші, які інтегруються з Microsoft Graph. Це створює міцний захист від зловмисників, оскільки Microsoft 365 відомий своїми високими стандартами безпеки та захисту даних.

За допомогою цього сервісу, проект може отримати доступ до різноманітної інформації про користувачів університету, такої як їхні профілі, календарі, контакти та інші дані. У розробці застосунку буде використовуватися список онлайн-нарад користувача для створення персоналізованих розкладів. Це дозволить створити застосунок, який буде надійно захищений та забезпечить зручний та ефективний досвід користувачів університету «КРОК».

2.6.2. Автентифікаця OpenID та OBO потік

Автентифікація - це термін, який позначає процес доведення того, що певний факт або документ є справжнім. У комп'ютерних науках цей термін зазвичай асоціюється з підтвердженням особи користувача. Зазвичай користувач підтверджує свою особу, надаючи свої облікові дані, тобто узгоджену інформацію, якою обмінюються користувач і система.

Авторизація - це процес визначення прав доступу користувача до певних ресурсів чи функціональності в системі після того, як користувач був ідентифікований. Цей процес встановлює, чи має користувач право на доступ до певних областей системи, файлів, даних чи функціональності.

У кожній системі авторизація виконується згідно з встановленими правилами та політиками безпеки. Користувачі можуть мати різні рівні доступу залежно від їхнього статусу, ролі або інших факторів. Ідентифікація - користувач вводить свої ідентифікаційні дані, такі як ім'я користувача та пароль, щоб підтвердити свою особу. Перевірка прав доступу - система перевіряє ці дані та визначає, які ресурси чи функціональність доступні користувачеві. Надання доступу - якщо користувач має необхідні права доступу, він отримує дозвіл на використання відповідних ресурсів чи функціональності. Відмова в доступі - у випадку, якщо користувач не має відповідних прав доступу, він отримує відмову в доступі та не може скористатися цими ресурсами чи функціональністю.

Авторизація важлива для забезпечення безпеки та контролю доступу до систем та даних. Вона дозволяє адміністраторам систем контролювати, які користувачі мають доступ до різних ресурсів, і захищати конфіденційні дані від несанкціонованого доступу.

Авторизація OAuth 2.0 - це протокол, який дозволяє користувачам дозволяти третім сторонам обмежений доступ до їхніх ресурсів без необхідності передавати їм свій пароль. Цей протокол широко використовується для авторизації користувачів в веб-додатках та API, забезпечуючи зручну та безпечну інтеграцію з іншими системами.

Потік OBO - це частина протоколу аутентифікації OAuth 2.0, яка дозволяє службам (наприклад, веб-застосункам або мікрослужбам) отримувати доступ до ресурсів в ім'я користувача, якого вони представляють. Коли веб-застосунок потребує доступу до захищених ресурсів в ім'я користувача, він може

використовувати потік OBO для отримання дозволу на доступ від сервера авторизації. Після успішної аутентифікації додаток отримує токен доступу, який може використовуватися для доступу до ресурсів в ім'я користувача.

Зазвичай потік OBO використовується в контексті виконання дій, які вимагають права доступу до ресурсів, що належать користувачеві. Наприклад, веб-застосунок для електронної пошти може отримати доступ до календаря користувача для створення або оновлення подій. Основна ідея потоку OBO полягає в тому, щоб дозволити додаткам діяти з ім'ям користувача, забезпечуючи при цьому безпеку та контроль доступу до ресурсів. Це дозволяє розробникам створювати потужні та гнучкі застосунки, які можуть виконувати дії від імені користувача з його дозволу.

OpenID Connect (OIDC) - це протокол аутентифікації, який побудований на основі протоколу OAuth 2.0 і надає розширені можливості для ідентифікації користувачів в мережі Інтернет. Він створений з метою забезпечення безпеки та простоти взаємодії між веб-застосунками та сервісами, які вимагають автентифікації. OIDC є стандартизованим протоколом, що дозволяє різним сервісам та додаткам спілкуватися між собою за єдиною автентифікаційною моделлю.

Під час автентифікації сервер авторизації генерує JWT, який містить інформацію про ідентифікованого користувача. Цей токен містить у собі дані про ідентифікацію, а також інформацію про аутентифікацію. Використання JWT дозволяє забезпечити безпеку аутентифікації та запобігти атакам, таким як перехоплення та підробка токенів. Протокол забезпечує простий та зручний процес аутентифікації для користувачів, що дозволяє швидко та безпечно увійти до системи.

Розглянемо роботу автентентифікації від Google, кожного разу при вході на більшість сайтів Вам пропонується використати акаунт Google для входу, але як це працює? Це і є OIDC. Сайт перенаправляє користувача на сервер

авторизації Google за допомогою OpenID Connect. Користувач вводить свої облікові дані на сторінці авторизації Google. Сервер Google перевіряє ці дані та ідентифікує користувача, після успішної авторизації сервер Google видає токен доступу, який містить інформацію про користувача. Користувач повертається до сторінки, а токен доступу передається нам. Тепер ми можемо використовувати цей токен, щоб отримати інформацію про користувача з сервера Google та дозволити йому використовувати дані користувача.

2.7. Обгрунтування вибору програмної реалізації

При розробці масштабного веб-застосунку, особливо в якості студента, важливо обрати технології та інструменти, які не лише відповідають вимогам проекту, а й дозволять ефективно виконувати завдання, враховуючи обмеження ресурсів і часу. У цьому контексті моїм вибором стали JavaScript з бібліотекою React для клієнтської частини, та C# з .NET Core і ASP.NET Core для серверної частини, базою даних обрав PostgreSQL, а для розгортання обрав хмарну платформу Heroku.

JavaScript є однією з найпопулярніших мов програмування для розробки веб-додатків, а React - однією з найбільш використовуваних бібліотек для створення інтерфейсів користувача. Використання React дозволить створити швидкий, реактивний та масштабований інтерфейс, що полегшить роботу з компонентами та управлінням станом додатку.

C# є мовою програмування, яка відома своєю продуктивністю, надійністю та широким спектром функцій. .NET Core і ASP.NET Core забезпечують швидку розробку, високу продуктивність та підтримку кросплатформенності. Використання цих технологій дозволить побудувати потужний та масштабований сервер, який буде готовий до обробки запитів від клієнтів.

PostgreSQL є потужною та надійною реляційною базою даних з відкритим вихідним кодом. Вона надає широкий набір функцій, які дозволяють ефективно

зберігати та обробляти дані. Використання PostgreSQL забезпечить стабільну роботу серверної частини додатку та надійність зберігання інформації.

Heroku є потужною та легкою у використанні хмарною платформою, яка надає безкоштовні можливості для студентів. Її простий інтерфейс дозволить швидко розгорнути веб-додаток та надати доступ до нього з будь-якого пристрою. Використання Heroku спростило процес розгортання та підтримки додатку, дозволяючи сконцентруватися на розробці функціональності.

Загалом, обрана комбінація технологій та інструментів відповідає потребам проекту, забезпечуючи ефективну розробку та надійну роботу додатку, а також використовує доступні безкоштовні ресурси для студентів, що є важливим аспектом при обмежених можливостях фінансування.

РОЗДІЛ 3. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ WEB-ЗАСТОСУНКУ

3.1. Опис бази даних

Для забезпечення ефективного зберігання та управління даними у нашому веб-застосунку ми обираємо реляційну базу даних PostgreSQL. PostgreSQL є потужною та надійною системою управління базами даних з відкритим вихідним кодом, яка надає широкий набір функцій та можливостей для зберігання та обробки даних. Однією з ключових переваг використання PostgreSQL є його широкий функціонал, який включає підтримку розширених типів даних, оптимізацію для високої навантаженості, масштабованість та високу надійність. Крім того, PostgreSQL має відкритий вихідний код та активну спільноту, що забезпечує швидку виправлення помилок та підтримку нових функцій.

Для побудови моделі бази даних та роботи з нею ми використовуватимемо ORM (Object-Relational Mapping) під назвою Entity Framework. Entity Framework є частиною технологічного стеку .NET Core та ASP.NET Core і надає можливість робити реляційне взаємозв'язування даних у форматі об'єктів, що спрощує розробку та підтримку додатку.

Ми обираємо підхід code-first для побудови моделі бази даних. Це означає, що спочатку ми визначимо моделі даних у вигляді класів C#, а потім Entity Framework автоматично згенерує відповідні таблиці бази даних, виходячи з цих класів. Цей підхід дозволяє нам працювати з даними на рівні об'єктів, що спрощує розробку та робить код більш зрозумілим та легким у підтримці. Окрім, перевагою використання Entity Framework у режимі code-first є можливість автоматичного створення або оновлення структури бази даних на основі змін у моделі даних, що робить процес розробки та супроводу додатку більш гнучким і ефективним. І Entity Framework надає можливість використовувати міграції, які дозволяють зберігати та контролювати версії бази

даних, забезпечуючи безпечну роботу з даними навіть у випадку змін у структурі або схемі даних.

Загалом, використання PostgreSQL як основи для зберігання даних разом з ORM Entity Framework у режимі code-first дозволить нам створити ефективну та розширювану модель бази даних, яка відповідатиме потребам нашого веб-застосунку і забезпечить надійну та швидку роботу з даними.

У зв'язку з тим, що ми отримуємо всі найважливіші дані за допомогою зв'язків з Graph API та іншими API, розмір нашої бази даних буде невеликим.

Табл. 3.1.

Поля таблиці "Users"

Ім'я поля	Тип та розмір поля	Опис поля
Id	uuid	Первинний ключ, ідентифікатор акаунту Microsoft
FullName	varchar(120)	Повне ім'я користувача з Microsoft акаунту
Group	varchar(60)	Навчальна група користувача з Microsoft акаунту
IsRestricted	boolean	Чи має користувач обмеження щодо опубліковування нотаток
CreationDate	timestamp	Дата створення користувача

Для зберігання нотаток, створених користувачами до навчальних пар, які будуть відображатись тільки в межах однієї групи користувачів, ми плануємо створити окрему таблицю "Notes"

Табл. 3.2.

Поля таблиці "Notes"

Ім'я поля	Тип та розмір поля	Опис поля
NoteId	uuid	Первинний ключ, ідентифікатор нотатки
LessonId	uuid	Ідентифікатор пари, на яку зроблено нотатку
Message	varchar(256)	Повний текст нотатки до пари
AuthorId	uuid	Ідентифікатор акаунту користувача автора нотатки, зв'язано с табл. 3.1.
CreationDate	timestamp	Дата створення нотатки користувачем

І також окремою буде таблиця для збереження завантажених даних по світловідключеням по містам.

Табл. 3.3.

Поля таблиці "Outages"

Ім'я поля	Тип та розмір поля	Опис поля
City	varchar(50)	Складовий ключ, місто для якого створено відключення
Group	varchar(50)	Складовий ключ, група для якої створено відключення
DayOfWeek	integer	Складовий ключ, день неділі для якої створено відключення
Outages	text	Текст масиву з відключеннями, який за допомогою EF буде десеріалізуватись у список

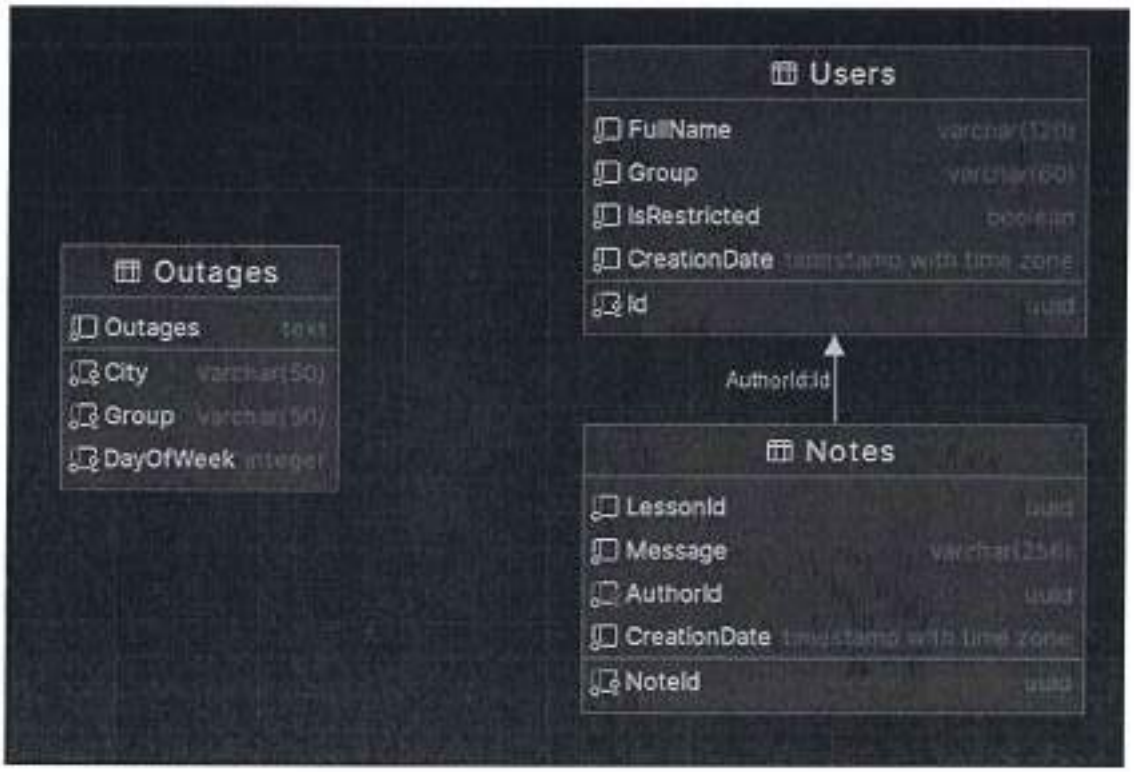

Рисунок 3.1 – Концептуальна модель бази даних

3.2. Програмна реалізація клієнтської частини

3.2.1. Використані пакетні модулі

@azure/msal-browser і @azure/msal-react: Ці пакети від Microsoft дозволяють інтегрувати аутентифікацію та авторизацію за допомогою Azure Active Directory у клієнтську частину додатка. @azure/msal-browser надає API для взаємодії з MSAL (Microsoft Authentication Library) у браузері, в той час як @azure/msal-react надає React-специфічні компоненти та хуки для зручної інтеграції аутентифікації у React-додаток.

axios: Цей пакет дозволяє здійснювати HTTP-запити з клієнтської частини додатка. Він надає простий та зручний інтерфейс для взаємодії зі сторонніми API та виконання асинхронних запитів.

bootstrap і react-bootstrap: Bootstrap є потужним фреймворком для розробки користувацького інтерфейсу, а react-bootstrap - це набір React-компонентів, які побудовані на базі Bootstrap. Вони дозволяють швидко та просто створювати стильні та респонсивні інтерфейси у React-додатках.

font-awesome: Цей пакет містить набір значків та іконок, які можна використовувати для покращення користувацького інтерфейсу. Він надає доступ до великого набору графічних елементів, які можна легко вставляти в додаток.

react і react-dom: React – це основна бібліотека JavaScript для розробки інтерфейсів, а react-dom - це пакет, який дозволяє взаємодіяти з DOM веб-сторінки у React-додатку. Вони дозволяють створювати динамічні та ефективні інтерфейси у веб-додатках.

react-router-dom: Цей пакет надає навігацію між сторінками та маршрутизацію у React-додатках. Він дозволяє визначати маршрути та зв'язувати їх з відповідними компонентами, що дозволяє створювати багатосторінкові додатки та зручно керувати навігацією.

3.2.2. Структура проекту

Для структурування клієнтської частини нашого веб-застосунку, створеного з використанням React, ми будемо організовувати наш код в кілька основних папок. Кожна папка матиме свою специфічну функцію, що дозволить нам підтримувати чистоту та впорядкованість коду, а також спростить його підтримку та масштабування. Ось опис кожної з основних папок:

У кореневому каталозі проекту, містяться конфігураційні файли, такі як vite.config.js, package.json, і інші, такі як .gitignore, README.md тощо. Файл vite.config.js є конфігураційним файлом для Vite. Він експортує об'єкт конфігурації, який визначає базову URL-адресу проекту (/SmartTimetable/), використовувані плагіни (в даному випадку React) і конфігурацію сервера.

Components – ця папка міститиме всі React-компоненти, які використовуються у додатку. React-компоненти – це фундаментальні будівельні блоки React-додатків. Вони являють собою ізольовані, повторно використовувані частини інтерфейсу користувача, які можуть містити як логіку, так і уявлення. Компоненти дозволяють розбивати інтерфейс на невеликі, керовані частини, які легше розробляти, тестувати та підтримувати. Компоненти є ключовою концепцією React, що дозволяє будувати складні та динамічні інтерфейси з використанням невеликих та повторно використовуваних частин. Це спрощує процес розробки, тестування та підтримки додатків, роблячи їх більш модульними та масштабованими. Кожен компонент зберігатиметься в окремій підпапці, що спростить його пошук та підтримку.

Hooks – тут зберігатимуться користувальницькі хуки. Хуки – це функції, що дозволяють використовувати стан та інші можливості React-функціональних компонентів. Хуки роблять функціональні компоненти більш потужними та гнучкими, спрощуючи написання та керування кодом. Хуки працюють, дозволяючи вам «підключатися» до можливостей React, таких як стан, життєві цикли та контекст усередині функціональних компонентів. Це спрощує логіку компонента і робить код більш читаним та підтримуваним. Користувацькі хуки – це хуки, створені для повторного використання логіки стану або побічних ефектів у різних компонентах.

Node_modules – у цьому каталозі зберігаються залежності проекту. Вони встановлюються за допомогою npm, як зазначено у файлі package.json.

Pages – папка, в якій зберігатимуться компоненти сторінок програми. Кожна сторінка є основним розділом веб-додатку, таким як головна сторінка, розклад вчителя, розклад студента та налаштування.

Assets – ця папка міститиме статичні ресурси, такі як зображення, шрифти та файли CSS. Тут ми зберігатимемо будь-які медіафайли та стилі, які використовуються в додатку.

Utils – ця папка міститиме допоміжні функції та утиліти, які можуть бути використані в різних частинах програми. Це можуть бути функції форматування дат, роботи з локальним сховищем та інші корисні утиліти.

Layout – ця папка міститиме компоненти, пов'язані із загальним макетом та структурою програми. Тут зберігатимуться компоненти, які визначають основні шаблони та каркаси для різних сторінок та розділів програми. Компоненти в цій папці зазвичай включають навігаційні елементи, бічні панелі, футери та інші структурні елементи, які повторюються на різних сторінках програми.

3.2.3. Детальна реалізація клієнтської частини

В основному файлі main.jsx наш React-додаток бере свій початок і стартує свій життєвий цикл. Цей файл відіграє ключову роль у конфігурації та початковому налаштуванні нашої програми, забезпечуючи його правильне функціонування та взаємодію з навколишнім середовищем.

Ініціалізація програми включає не тільки впровадження основних компонентів, таких як роутер і контекст аутентифікації, але й налаштування обробників подій і завантаження зовнішніх ресурсів. Наприклад, ми можемо встановити обробники подій для обробки ситуацій, коли користувач успішно автентифікується або коли виникають помилки під час завантаження даних.

Крім ініціалізації основних компонентів, файл main.jsx також є місцем, де ми можемо підключати додаткові бібліотеки та ресурси, необхідні для роботи нашої програми. Наприклад, ми можемо імпортувати стилі або скрипти, визначити глобальні змінні або константи, які будуть доступні у всьому додатку, наприклад, саме тут відбувається підключення бібліотеки стилів Bootstrap.

У контексті роутера, ми визначаємо структуру маршрутизації програми, задаючи відповідність між шляхами URL-адреси та компонентами, які мають бути відображені для кожного шляху. Це дозволяє нам створювати багатосторінкові програми зі зручною навігацією та вмістом, що динамічно змінюється.

Інтеграція з бібліотекою автентифікації Майкрософт (MSAL) дозволяє нам забезпечити безпеку та захищений доступ до нашої програми. MSAL обробляє процес автентифікації користувача, надаючи доступ до токенів та можливості автентифікації за допомогою облікових даних Microsoft.

Таким чином, основний файл main.jsx є вихідною точкою для налаштування та запуску нашої програми React, де ми визначаємо його основні компоненти і забезпечуємо початкову конфігурацію для правильної роботи програми. Це місце, де зустрічаються різні частини нашої програми та де починається його життєвий цикл, наповнюючи його функціональністю та інтерактивністю.

У структурі нашого проекту ми звертаємо увагу до різні файли, кожен із яких виконує свою певну роль.

У папці utils розміщуються допоміжні функції, призначені для різних завдань. Наприклад, файл Requests.js містить функції для виконання запитів до API. Тут ми можемо знайти різні типи запитів, такі як запити отримання даних, оновлення інформації чи видалення записів. Ці функції надають зручний інтерфейс для взаємодії із зовнішніми даними, дозволяючи компонентам програми отримувати дані асинхронно та обробляти їх відповідно до потреб.

Інші файли в папці utils можуть включати функції для форматування тексту. Ці функції допомагають зробити код більш модульним, читаним та підтримуваним, розділяючи логіку додатка на дрібніші та зрозуміліші частини.

Використання допоміжних функцій в окремих модулях дозволяє підвищити ефективність та перевикористання коду, оскільки вони можуть бути легко імпортовані та використані в різних частинах програми.

Далі маємо папку hooks, у якій знаходяться кастомні хуки, які являють собою набір функцій, що полегшують взаємодію компонентів програми з різними аспектами його функціональності.

Один із ключових кастомних хуків - useLocalStorage.Основне його завдання полягає в тому, щоб полегшити процес збереження стану програми між різними сеансами користувача або для зберігання налаштувань користувача і переваг. У додатку використовується useLocalStorage для збереження інформації про збережені фільтри при виборі групи або викладача, або стан програми під час його закриття.

Основна перевага useLocalStorage полягає в тому, що він забезпечує збереження даних між сеансами користувача навіть після закриття браузера або перезавантаження сторінки. Це робить його відмінним вибором для збереження налаштувань та переваг користувачів, що сприяє покращенню користувальницького досвіду та загальної зручності використання програми.

Хук useIsMobile є інструментом для визначення типу пристрою, на якому запущено веб-додаток. Цей хук дозволяє розробникам створювати адаптивні інтерфейси, які автоматично реагують на розмір екрану та тип пристрою користувача, покращуючи таким чином досвід користувача.

Під час виклику useIsMobile він повертає логічне значення, що вказує, чи є поточний пристрій мобільним чи ні. Це значення можна використовувати в умовах або під час створення компонентів, щоб реагувати на тип пристрою та змінювати відображення інтерфейсу відповідним чином.

Переваги використання useIsMobile включають можливість створення адаптивних інтерфейсів, які оптимально відображаються на різних пристроях, підвищуючи цим зручність використання програми для користувачів. Це

дозволяє розробникам створювати більш гнучкі та доступні веб-програми, які можуть ефективно працювати на широкому спектрі пристроїв та екранів.

Хук useConfiguration являє собою зручний спосіб отримання основних налаштуваннями користувача, такими як обрані фільтри, статус авторизації та інші. Це дозволяє легко налаштовувати програму під індивідуальні уподобання користувачів і створювати персоналізований досвід користувача.

Нарешті, хук useSchedule надає дані про розклад занять користувача. Він є основним інструментом роботи з розкладом занять. Він надає зручний інтерфейс для отримання списку занять на певну дату та управління ними, що робить його корисним інструментом для програм, пов'язаних з освітнім процесом або організацією робочого часу.

Основне завдання полягає в тому, щоб надати користувачеві доступ до інформації про його розклад занять на певну дату. Для цього хук може використовувати дані із сервера та оновлювати його за потреби.

При виклику useSchedule він повертає список занять на вказану дату, а також функції оновлення цього списку при зміні дати або інших параметрів. Це дозволяє легко відображати розклад занять користувачеві та надавати можливість планувати свій час, переваги використання включають можливість створення зручного та інтуїтивно зрозумілого інтерфейсу для перегляду розкладу занять, покращуючи тим самим досвід користувача. Це також робить програму більш функціональною та корисною для користувачів, дозволяючи їм ефективно організовувати свій час та планувати свої заняття.

Проект є комплексним рішенням, що поєднує в собі 20 різних компонентів, які тісно взаємодіють один з одним, утворюючи смарт-розклад для навчальних або робочих завдань. Ці компоненти охоплюють широкий спектр функціональності та відповідальні за різні аспекти роботи програми

3.2.4. Модуль авторизації

У реактивних додатках авторизація за допомогою Microsoft Graph зазвичай відбувається через використання бібліотеки @azure/msal-react для інтеграції з Azure Active Directory та MSAL (Microsoft Authentication Library) для JavaScript.

Процес авторизації буде починатись з ініціалізації MSAL, зпочатку додаток ініціалізує бібліотеку MSAL для взаємодії з Azure Active Directory. Під час ініціалізації визначаються параметри клієнта, такі як ідентифікатор клієнта, авторитет та URL-адреса перенаправлення. Після цього йде аутентифікація користувача у системі, після ініціалізації MSAL, користувач переходить до процесу аутентифікації. Це зазвичай включає введення облікових даних користувача та надання дозволу на доступ до його облікового запису.

Після успішної аутентифікації MSAL генерує токен доступу, який надається клієнтському веб-застосунку. З отриманим токеном доступу додаток може здійснювати запити до Microsoft Graph API для отримання, оновлення або видалення даних з облікових записів користувачів. Після виклику API додаток обробляє отриману відповідь і відображає або виконує відповідні дії відповідно до бізнес-логіки додатку.

Оновлення токену доступу: Токен доступу має обмежений термін дії. Після закінчення терміну дії токен потрібно оновити для продовження доступу до API без повторної аутентифікації користувача. Він дозволяє забезпечити безпеку та захищеність облікових записів користувачів, а також надає можливість взаємодії з різними сервісами та даними, які знаходяться у Microsoft.

3.3. Програмна реалізація серверної частини

3.3.1. Використані пакетні модулі

AspNet.Versioning.Mvc: пакет дозволяє вам легко налаштувати версіонування API у вашому ASP.NET Core додатку. Він надає можливість

визначити правила маршрутизації для кожної версії API та забезпечує сумісність з існуючими проектами.

FluentValidation: Цей пакет дозволяє вам валідувати дані у вашому ASP.NET Core додатку за допомогою Fluent API. Він надає зручний та ефективний спосіб визначення правил валідації для вашої моделі даних.

Microsoft.AspNetCore.OpenApi: пакет дозволяє автоматично створювати документацію API з вашого ASP.NET Core додатку за допомогою стандарту OpenAPI (раніше відомого як Swagger). Він надає інтерактивну документацію з можливістю пробування запитів безпосередньо з браузера.

Microsoft.EntityFrameworkCore: пакет надає доступ до Entity Framework Core, який є ORM (Object-Relational Mapping) для роботи з базою даних у ASP.NET Core додатках. Він дозволяє легко взаємодіяти з базою даних за допомогою коду C#.

Microsoft.Identity.Web: пакет надає підтримку аутентифікації та авторизації у ASP.NET Core додатках за допомогою Azure Active Directory. Він дозволяє легко і безпечно інтегрувати Azure AD у ваш додаток та взаємодіяти з Microsoft Graph API.

Moesif.Middleware: пакет надає можливість відстежувати та аналізувати HTTP-запити та відповіді у вашому ASP.NET Core додатку. Він дозволяє збирати метрики про додаток та користувачів для поліпшення продуктивності та ефективності.

Quartz.AspNetCore: пакет дозволяє вам планувати та виконувати завдання у вашому ASP.NET Core додатку за допомогою бібліотеки планувальника робіт Quartz.NET.

Riok.Mapperly: пакет надає можливість мапування об'єктів у вашому ASP.NET Core додатку за допомогою декларативних конфігурацій.

SharpGrip.FluentValidation.AutoValidation.Mvc:пакет дозволяє автоматично валідувати вхідні моделі даних у вашому ASP.NET Core додатку за допомогою FluentValidation і без необхідності вручного визначення правил валідації.

Microsoft.Graph: пакет надає зручний спосіб взаємодії з Microsoft Graph API у вашому .NET додатку. Він дозволяє вам взаємодіяти з різними сервісами та ресурсами, що знаходяться у Microsoft 365, такими як пошта, календар, контакти, файли та багато іншого.

Newtonsoft.Json: пакет є одним з найпопулярніших бібліотек для роботи з JSON у .NET додатках. Він дозволяє серіалізувати та десеріалізувати об'єкти .NET у формат JSON та навпаки. Newtonsoft.Json широко використовується для обміну даними між сервером та клієнтом, а також для зберігання даних у форматі JSON.

Npgsql.EntityFrameworkCore.PostgreSQL: пакет надає підтримку PostgreSQL у вашому .NET додатку за допомогою Entity Framework Core. Він дозволяє вам використовувати PostgreSQL як базу даних для зберігання та обробки даних у вашому додатку.

Quartz.Jobs: пакет містить різноманітні роботи для використання у планувальнику завдань Quartz.NET. Він надає можливість планувати та виконувати різні завдання у вашому додатку, такі як відправлення листів, оновлення даних та багато іншого.

FluentAssertions: пакет дозволяє вам писати зрозумілі та зрозумілі тести у вашому .NET додатку. Він надає зручний та експресивний спосіб визначення тестових умов та перевірки їх виконання.

Moq: пакет надає можливість створювати та налаштовувати підроблені об'єкти для тестування у вашому .NET додатку. Він дозволяє вам ефективно виконувати юніт-тести та перевіряти поведінку вашого коду.

xunit: пакет надає потужний фреймворк для створення та виконання юніт-тестів у вашому .NET додатку. Він дозволяє вам легко створювати та виконувати тести для перевірки правильності вашого коду.

3.3.2. Структура проекту

Проект серверної частини організований з використанням трьох підпроектів, кожен із яких виконує певні функції та забезпечує конкретні аспекти роботи програми.

Перший підпроект WebTimetable.Api є основним інтерфейсом для взаємодії з клієнтською частиною програми. Він відповідає за прийом HTTP-запитів від клієнтів, їх валідацію, автентифікацію та авторизацію користувачів, а також за надсилання відповідних відповідей. Цей підпроект містить контролери, маршрутизацію запитів та інші компоненти, необхідні обробки запитів і управління потоком даних між клієнтом і сервером.

Другий підпроект, WebTimetable.Application, є центром бізнес-логіки програми. Тут містяться сервіси, обробники запитів, моделі даних та інші компоненти, що відповідають за виконання бізнес-правил, обробку даних та взаємодію з базою даних. Цей підпроект відповідає за основну функціональність програми, включаючи обробку даних користувачів, виконання операцій та оновлення стану програми.

Третій підпроект, WebTimetable.Contracts, містить моделі-контракти, які визначають структуру даних, що використовуються у додатку. Ці контракти визначають формати даних, які передаються між клієнтом та сервером, а також між різними компонентами програми. Вони є важливим засобом обміну інформацією та забезпечують її структуроване уявлення на різних рівнях програми.

Четвертий підпроект включає модульні тести для перевірки функціональності серверної частини програми. У цьому проекті зберігаються тести, що перевіряють коректність роботи API, валідацію даних, а також інші

аспекти роботи сервера. Тести дозволяють виявляти помилки та вразливості у додатку на ранніх етапах розробки, забезпечуючи його надійність, стабільність та якість.

Такий поділ відповідальності дозволяє ефективно організувати роботу серверної частини програми, спростити її розробку та супровід, а також підвищити надійність та масштабованість кодової бази. Кожен підпроект фокусується на своїй галузі функціональності та взаємодіє з іншими підпроектами через чітко визначені інтерфейси, що забезпечує узгодженість та цілісність роботи всього додатка.

3.3.3. Об'єкто-реляційне відображення з Entity Framework

У нашому проекті аза даних моделюється за допомогою Entity Framework Core, який забезпечує об'єктно-реляційне відображення між об'єктами вашого додатку та таблицями у базі даних.

Модель бази даних знаходиться у проекті WebTimetable.Application у папці Entities. Сутності бази даних в цій папці відображають структуру даних вашого додатку. Кожна сутність представляє собою таблицю у базі даних, а кожне поле сутності відображає колонку у таблиці. Ці сутності дозволяють створювати і взаємодіяти з об'єктами застосунку у базі даних. Ви можете створювати, зчитувати, оновлювати та видаляти записи в цих таблицях за допомогою Entity Framework Core та DbContext. Це дозволяє легко управляти даними у вашому додатку та забезпечує гнучкість та ефективність у роботі з базою даних.

Міграції, які забезпечують автоматичне оновлення схеми бази даних при зміні моделі даних, знаходяться у папці Migrations. Кожна міграція відповідає певному зміні у моделі даних та генерує відповідний SQL-скрипт для оновлення бази даних.

Основний клас, що описує поведінку створення та управління базою даних, DbContext, знаходиться у корневій папці. Він є основним інтерфейсом

між застосунком та базою даних, і відповідає за виконання запитів до бази даних, зберігання та оновлення даних. Клас DbContext включає в себе набір DbSet властивостей, які представляють собою колекції сутностей бази даних, а також методи для взаємодії з цими колекціями. Крім того, ви можете конфігурувати різні аспекти взаємодії з базою даних, такі як відносини між сутностями, індекси, налаштування відслідковування змін тощо.

3.3.4. Детальна реалізація серверної частини

У нашому проекті серверна програма реалізована як Web-API, архітектурно поділена на два основні компоненти: безпосередньо API та застосунок, який маніпулює даними. Цей підхід дозволяє чітко розмежувати обов'язки та забезпечити гнучкість і масштабованість системи.

Коли клієнтський запит надходить на сервер, він обробляється за допомогою ASP.NET Core. Платформа ASP.NET Core відповідає за маніпуляцію з вхідними запитами та направлення їх до відповідних контролерів. Це відбувається завдяки вбудованій маршрутизації, яка дозволяє визначити, який саме контролер та метод повинні обробити конкретний запит. ASP.NET Core це, буквально, ядро нашого серверного додатку.

Коли клієнтський запит надходить на сервер, він обробляється ASP.NET Core. Спочатку запит проходить через середовище проміжного програмного забезпечення (middleware), де може бути виконано логування, автентифікація, авторизація та інші попередні обробки. Далі запит передається до маршрутизатора, який визначає відповідний контролер та метод для обробки запиту.

Контролер отримує запит і, за потреби, викликає відповідні методи сервісів для обробки даних. Сервіси, в свою чергу, взаємодіють з репозиторіями для виконання необхідних операцій з базою даних. Після обробки запиту контролер повертає результат клієнту у вигляді HTTP-відповіді. ASP.NET Core є потужним та гнучким фреймворком для створення веб-додатків та API. Він

забезпечує високу продуктивність, безпеку та легку інтеграцію з іншими сервісами. Усі запити від фреймворка переходять до контроллерів. Контролери є основними обробниками HTTP-запитів. Кожен контролер відповідає за певний набір функціональності і взаємодіє з відповідними сервісами та репозиторіями для виконання запитів. У нашому проекті ми маємо п'ять основних контролерів: Cabinets, Notes, Settings, Students та Teachers. Вони забезпечують обробку запитів, пов'язаних з різними аспектами роботи системи. Сервіси, у свою чергу, реалізують бізнес-логіку застосунку. Вони забезпечують обробку даних та взаємодію з репозиторіями для виконання необхідних операцій. Сервіси дозволяють відокремити бізнес-логіку від контролерів, що спрощує підтримку та тестування коду.

CabinetsController відповідає за управління інформацією про кабінети в будівлі університету. Його основне завдання - надання шляхів та фільтрів для отримання даних про наявні кабінети. Наприклад, користувач може запитати список всіх кабінетів, фільтрувати їх за поверхом, типом або іншими критеріями.

Контролер включає методи для виконання різних запитів:

- Отримання списку всіх кабінетів.
- Отримання детальної інформації про конкретний кабінет.
- Фільтрація кабінетів за різними параметрами.

Усі запити до CabinetsController повинні бути авторизовані, що забезпечує безпеку доступу до інформації. Модуль авторизації перевіряє, чи має користувач необхідні права для виконання запиту.

NotesController займається управлінням нотатками, створеними користувачами. Його основне завдання - створення та видалення нотаток, пов'язаних з навчальними заняттями. Нотатки можуть бути створені студентами

для зберігання важливої інформації, яка буде відображатися лише в межах групи користувачів.

Контролер включає методи для виконання таких запитів:

- Створення нової нотатки.
- Видалення існуючої нотатки.

NotesController також захищений авторизацією, що забезпечує безпеку та приватність нотаток. Модуль авторизації перевіряє права користувача на виконання кожної операції.

StudentsController та TeachersController займаються найголовнішим завданням - видачею навчальних розкладів та фільтрів для налаштування. Вони дозволяють студентам та викладачам отримувати актуальні розклади занять, а також налаштовувати фільтри для зручного доступу до потрібної інформації.

SettingsController відповідає за управління налаштуваннями системи, зокрема за видачу інформації про групи відключень світла. Це дозволяє користувачам отримувати актуальну інформацію про можливі відключення та планувати свої дії відповідно до цих даних.

Розглянемо окремо як працює один з найважливіших частей застосунку, це оновлення графіків відключення.

Модуль відключень є важливим компонентом нашого серверного додатку. Його основне завдання - забезпечення актуальної інформації про графіки електро відключення, що є критично важливим для користувачів нашої системи.

Кожні чотири години модуль відправляє запит до сайту ДТЕК, який містить інформацію про відключення електроенергії. Цей процес реалізований за допомогою веб-скрапінгу, що дозволяє отримувати необхідні дані безпосередньо зі сторінки сайту.

Отримані дані обробляються та зберігаються у базі даних. Для цього використовується підхід атомарних операцій, що гарантує цілісність даних. Атомарність процесу означає, що оновлення бази даних відбувається як єдина неділима операція. Якщо з будь-якої причини програма не зможе отримати дані з сайту ДТЕК, наприклад, у разі недоступності сайту або помилки в процесі веб-скрапінгу, у базі даних залишаються попередні дані. Це гарантує, що користувачі завжди матимуть доступ до інформації, навіть якщо вона не є найактуальнішою.

Процес оновлення даних виконується автоматично за допомогою планувальника задач, такого як Quartz.NET. Кожні чотири години планувальник запускає завдання, яке виконує запит до сайту ДТЕК, обробляє отримані дані та оновлює базу даних. Якщо в процесі оновлення виникають помилки, попередні дані залишаються незмінними, що забезпечує стабільність системи.

3.3.5. Модуль авторизації

У цьому проекті Microsoft Graph використовується для автентифікації та авторизації, зокрема, за допомогою потоку On-Behalf-Of (OBO). Цей потік використовується, коли програма викликає службу/веб-API, яка, у свою чергу, має викликати іншу службу/веб-API.

Ідея полягає в тому, щоб поширювати делеговану ідентифікацію користувача та дозволи через ланцюжок запитів. Коли користувач намагається отримати доступ до програми у клієнтській частині, він перенаправляється на сторінку входу, якою керує Azure AD. Користувач вводить свої облікові дані, які потім надсилаються в Azure AD для перевірки. Далі, якщо облікові дані дійсні, Azure AD видає маркер доступу та ідентифікатор. Маркер доступу використовується для автентифікації користувача для кожного наступного запиту, який він робить до програми. Ідентифікаційний маркер містить відомості про користувача, наприклад його ім'я та адресу електронної пошти. Далі від клієнтської частини цей маркер доступу переходить до серверної

частини, серверна частина має можливість користуватись маркером саме через потік OBO, коли програмі потрібно викликати іншу службу або веб-API, вона може використовувати цей потік даних.

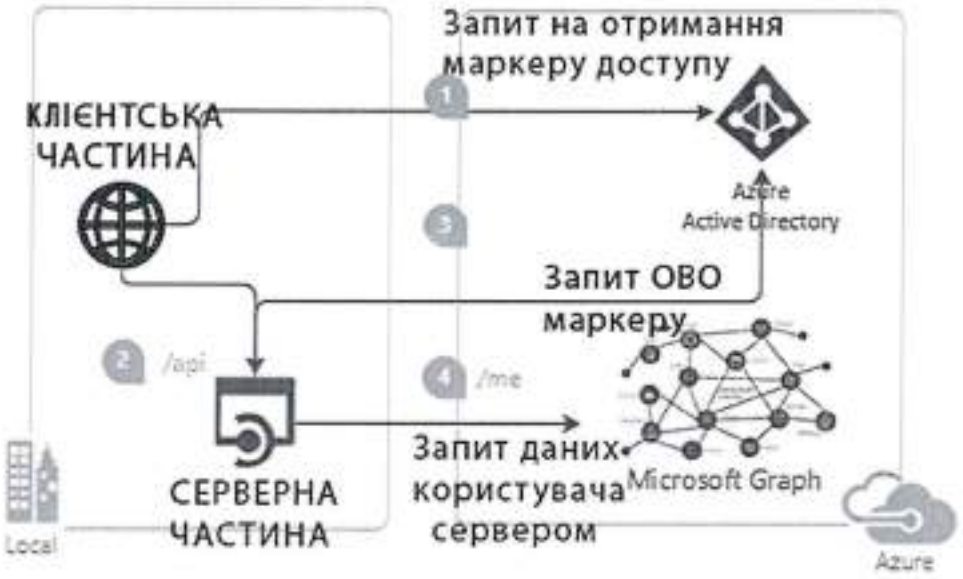

Рисунок 3.2 – Модель перетіку запиту токену

Програма надсилає запит до Azure AD, включаючи маркер доступу користувача. Azure AD перевіряє цей маркер і, якщо він дійсний, видає новий маркер доступу для використання програмою під час виклику другої служби або веб-API. Цей новий маркер містить дозволи користувача, тому друга служба або веб-API може знати, хто такий користувач і що йому дозволено робити., який надав дозвіл на користування ним.

3.3.6. Тестування

Тестування серверної частини є критично важливою частиною розробки веб-застосунків, оскільки воно дозволяє забезпечити стабільність, надійність та безпеку додатку. У вашому проекті ви використовуєте два основні пакети для тестування: xUnit і FluentAssertions. Використовуючи ці інструменти, ви намагаєтеся покрити найбільшу кількість коду тестами, що є хорошою практикою для забезпечення якості програмного забезпечення. Задля

правильної структури тестів було використано конвенцію найменування "Назва классу _Метод классу_Очікуваний результат".

Для правильної структури тестування було задіяно методику AAA.

Методика Arrange-Act-Assert (AAA) є загальноприйнятим підходом до написання юніт-тестів, що допомагає структурувати тести і зробити їх більш читабельними та зрозумілими. У вашому проекті, де використовуються xUnit та FluentAssertions, методика AAA дозволяє створювати добре організовані тести, які легко підтримувати та розуміти.

Arrange: На цьому етапі ви готуєте всі необхідні дані та залежності для тесту. Це може включати створення об'єктів, налаштування вхідних даних, конфігурацію мока (підробки) для залежностей і все інше, що потрібно для виконання тесту. Мета цього етапу - створити середовище, необхідне для виконання тесту.

Act: На цьому етапі ви виконуєте дію, яку хочете протестувати. Це зазвичай означає виклик методу чи функції, яку ви перевіряєте. Мета цього етапу - виконати конкретну операцію, щоб можна було перевірити її результат.

Assert: На цьому етапі ви перевіряєте результат виконаної дії. Використовуючи твердження (assertions), ви перевіряєте, чи відповідає результат очікуваному. Це може включати перевірку значень, станів об'єктів, виключень та інших умов. Мета цього етапу - впевнитися, що результат виконання тестованої дії відповідає очікуванням.

xUnit є одним з найпопулярніших фреймворків для тестування у .NET. Він дозволяє створювати і виконувати юніт-тести для перевірки правильності роботи вашого коду. xUnit пропонує простий і зрозумілий синтаксис для написання тестів, а також потужний набір інструментів для організації та виконання тестів. Однією з ключових особливостей xUnit є те, що він підтримує принцип "один тест - один метод", що дозволяє легко організовувати і підтримувати тестовий код.

FluentAssertions є бібліотекою для написання тверджень у тестах на більш зрозумілій та читабельній мові. Вона дозволяє писати тести таким чином, що вони стають легшими для читання і розуміння. FluentAssertions пропонує широкий набір методів для перевірки різних умов, включаючи перевірку значень, колекцій, виключень та багато іншого. Це дозволяє вам створювати детальні та точні тести для вашого коду.

При тестуванні серверної частини була ціль намагатись покрити якнайбільше аспектів вашого коду.

Було проведено тестування контролерів. Перевірка правильності роботи контролерів, включаючи обробку HTTP-запитів, виклик сервісних методів та повернення коректних HTTP-відповідей. Було перевірено, чи контролери правильно реагують на різні типи запитів і чи повертають очікувані результати.

Тестування основних сервісів які оброблюють запити від користувачей має бути приорітетною задачею. Перевірка логіки бізнес-рівня, яка зазвичай реалізується у сервісних класах. Було перевірено, чи сервіси коректно обробляють дані, взаємодіють з репозиторіями та виконують потрібні операції.

Тестування репозиторіїв: Перевірка правильності роботи репозиторіїв, включаючи виконання запитів до бази даних, збереження та оновлення даних. Ви переконуєтесь, що репозиторії коректно взаємодіють з базою даних та виконують необхідні операції.

При написанні тестів використано xUnit для організації та виконання тестів, а FluentAssertions для написання читабельних та зрозумілих тверджень. Наприклад, ви можете перевіряти, чи повертає метод очікуваний результат, чи генерується виключення при неправильному вводу, чи правильно взаємодіють різні компоненти додатку.

Тестування серверної частини дозволяє вам виявляти та виправляти помилки на ранніх стадіях розробки, забезпечувати стабільність та надійність додатку, а також зменшувати ризики виникнення проблем у продакшн

середовищі. Використання xUnit та FluentAssertions дозволяє вам ефективно організовувати процес тестування та забезпечувати високу якість вашого коду.

ВИСНОВОК

Насамперед метою було створення інтуїтивно зрозумілого та зручного інтерфейсу, який дозволяв би студентам та викладачам легко орієнтуватися у розкладі. Головною цілью було те, щоб користувачі могли швидко знаходити потрібну інформацію та отримувати доступ до розкладу у будь-який час та з будь-якого пристрою. Використовуючи сучасні технології, такі як React та Bootstrap. Додаток став простим у використанні та візуально привабливим.

Також у цілях було забезпечити функціональність, яка б виходила за рамки простого перегляду розкладу. Було інтегровано можливості взаємодії студентам друг з одним. Студенти можуть залишати нотатки під заняттями, обмінюватися корисною інформацією та стежити за запланованими онлайн-зустрічами. Викладачі, своєю чергою, отримали можливість краще планувати свої заняття.

Одним із важливих завдань було забезпечення надійності та безпеки даних. Ми хотіли, щоб користувачі могли бути впевнені у збереженні своєї особистої інформації та даних про розклад. Використовуючи .NET Core та ASP.NET Core для створення серверної частини, ми забезпечили високу продуктивність та безпеку нашої програми.

Ще однією значущою метою було створення функціоналу для адаптації до зовнішніх умов, таких як світловідключення. Інтеграція з графіками відключень світла по Україні дозволяє користувачам бути в курсі можливих перебоїв та планувати свій час відповідно.

У ході розробки були ретельно вивчені потреби студентів та викладачів університету, а також особливості та вимоги до веб-додатку для управління розкладом занять. Було приділено увагу кожної деталі, прагнучи створити продукт, який відповідає очікуванням користувачів, а й перевершує їх.

У процесі роботи над клієнтською частиною сили були зосереджені на розробці інтуїтивно зрозумілого та комфортного інтерфейсу. Були використані сучасні технології, такі як React та Bootstrap, щоб забезпечити хороший користувальницький досвід та ефективну роботу з додатком. Кожна сторінка була розроблена з урахуванням потреб користувачів, забезпечуючи простоту навігації та доступність інформації.

У серверній частині ми дотримувалися принципів надійності, збереження та масштабованості. Використовуючи .NET Core та ASP.NET Core, було створено стабільне та ефективне API, здатне обробляти запити користувачів з високою продуктивністю та безпекою. Також було передбачено механізми автентифікації та авторизації, що забезпечують захист конфіденційної інформації користувачів.

Програма стала не просто інструментом для перегляду розкладу, а й справжнім центром управління навчальним процесом. Його багатофункціональність дозволяє студентам не лише легко орієнтуватися у своєму розкладі, а й активно взаємодіяти з одногрупниками. Вони можуть ділитися нотатками, обмінюватися корисною інформацією, що сприяє більш ефективному та продуктивному навчальному процесу.

Викладачі також отримують значні переваги завдяки функціональності програми. Вони можуть ефективно управляти своїм розкладом. Завдяки можливості швидкого доступу до інформації про розклад та інструменти для планування, викладачі можуть проводити заняття більш структуровано та продуктивно, що сприяє підвищенню якості освіти.

Проект був успішно опублікований на хостингах, що дозволяє користувачам легко та швидко отримати доступ до програми з будь-якого місця та в будь-який час.

Всі цілі, поставлені перед нами у процесі розробки, були успішно досягнуті. Було створено зручний, функціональний та безпечний веб-додаток

для управління розкладом, який надає студентам та викладачам університету всі необхідні інструменти для ефективної взаємодії та планування навчального процесу. Додаток не тільки полегшує доступ до інформації про розклад, а й сприяє покращенню якості освіти за рахунок додаткових можливостей для взаємодії та адаптації до зовнішніх умов.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Архітектура програмного забезпечення [Електронний ресурс]. – Режим доступу: https://wezom.com.ua/ua/blog/arhitektura-programmnogo-obespecheniya
2. Розробка та аналіз вимог до програмного забезпечення [Електронний ресурс]. – Режим доступу: https://ela.kpi.ua/bitstream/123456789/38101/1/Rozrobka_ta_analiz_KP.pdf
3. Як зробити технічне завдання [Електронний ресурс]. – Режим доступу: https://blog.jungo.dev/uk/2021/05/yak-zrobyty-tehnichne-zavdannya/
4. Архітектура та проектування програмного забезпечення [Електронний ресурс]. – Режим доступу: http://dspace.wunu.edu.ua/jspui/bitstream/316497/24194/1/опорний%20конспект%20лекцій.pdf
5. Алгоритмізація та розробка програмного забезпечення [Електронний ресурс]. – Режим доступу: https://jait.donnu.edu.ua/article/view/12257
6. .NET fundamentals documentation [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/dotnet/fundamentals/
7. ASP.NET Documentation [Електронний ресурс]. – Режим доступу:
8. https://learn.microsoft.com/ru-ru/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-8.0
9. Github Copilot - Your AI Programmer [Електронний ресурс]. – Режим доступу: https://github.com/features/copilot
10. Azure DevOps Services [Електронний ресурс]. – Режим доступу: https://azure.microsoft.com/en-us/products/devops
11. Introduction to web APIs [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Introduction
12. What is RESTful API [Електронний ресурс]. – Режим доступу: https://aws.amazon.com/what-is/restful-api/
13. Create a web API with ASP NET Core [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/aspnet/core/tutorials/first-web-api?view=aspnetcore-8.0&tabs=visual-studio
14. JavaScript programming language [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/JavaScript
15. What is the DOM? Document Object Model Meaning in JavaScript [Електронний ресурс]. – Режим доступу: https://www.freecodecamp.org/news/what-is-the-dom-document-object-model-meaning-in-javascript/

16. About React [Електронний ресурс]. – Режим доступу: https://react.dev
17. React's Virtual DOM [Електронний ресурс]. – Режим доступу: https://medium.com/@BharathkumarV/reacts-virtual-dom-17fdcb290a10
18. What is Heroku [Електронний ресурс]. – Режим доступу: https://www.heroku.com/what
19. About Moesif [Електронний ресурс]. – Режим доступу: https://www.moesif.com/about
20. PostgreSQL Advantages and Disadvantages [Електронний ресурс]. – Режим доступу: https://www.aalpha.net/blog/pros-and-cons-of-using-postgresql-for-application-development/
21. Bootstrap examples [Електронний ресурс]. – Режим доступу: https://getbootstrap.com/docs/5.3/examples/
22. All React hooks in one short examples [Електронний ресурс]. – Режим доступу: https://medium.com/@AbidKazmi/all-react-hooks-in-one-short-4b0ed4b5a6e4
23. Microsoft Authentication Library for JavaScript (MSAL.js) Disadvantages [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/javascript/api/overview/msal-overview
24. React single-page application using MSAL Graph [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/samples/azure-samples/ms-identity-ciam-javascript-tutorial/ms-identity-ciam-javascript-tutorial-1-sign-in-react/
25. Overview of Microsoft Graph [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/graph/overview
26. Configuring Microsoft Graph [Електронний ресурс]. – Режим доступу: https://docs.logrhythm.com/OCbeats/docs/configure-microsoft-graph-api
27. Authentication basics Graph [Електронний ресурс]. – Режим доступу: https://auth0.com/intro-to-iam/what-is-authentication
28. OpenID vs OAuth: Understanding API Security Protocols [Електронний ресурс]. – Режим доступу: https://konghq.com/blog/engineering/openid-vs-oauth-what-is-the-difference
29. On-behalf-of flows with MSAL.NET [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/entra/msal/dotnet/acquiring-tokens/web-apps-apis/on-behalf-of-flow
30. What is Entity Framework? [Електронний ресурс]. – Режим доступу: https://www.entityframeworktutorial.net/entityframework6/what-is-entityframework.aspx
31. Entity Framework Migrations Overview [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/ef/core/managing-schemas/migrations

32. Arrange-Act-Assert [Електронний ресурс]. – Режим доступу: https://docs.telerik.com/devtools/justmock/basic-usage/arrange-act-assert

ДОДАТКИ

Додаток А

Скріншоти клієнтської частини веб-застосунку

Рисунок 1 – Головна сторінка

Рисунок 2 – Розклад студента

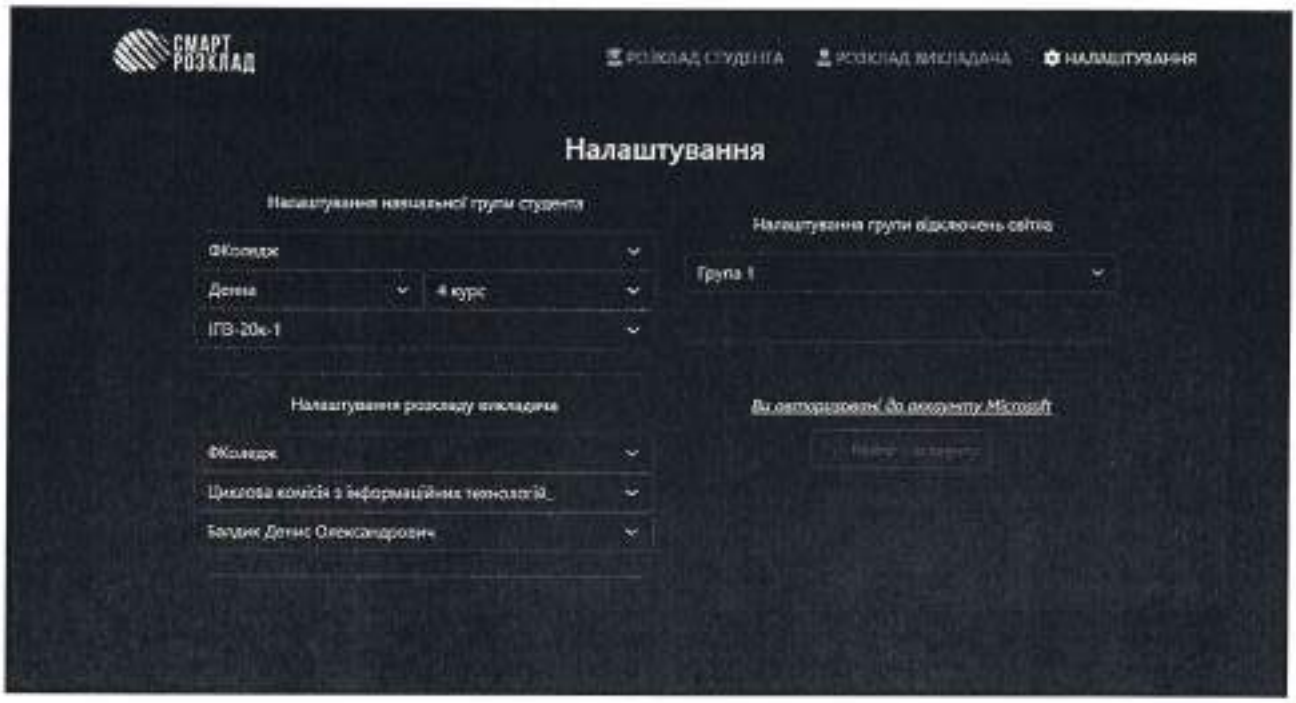

Рисунок 3 – Сторінка налаштувань

Додаток Б

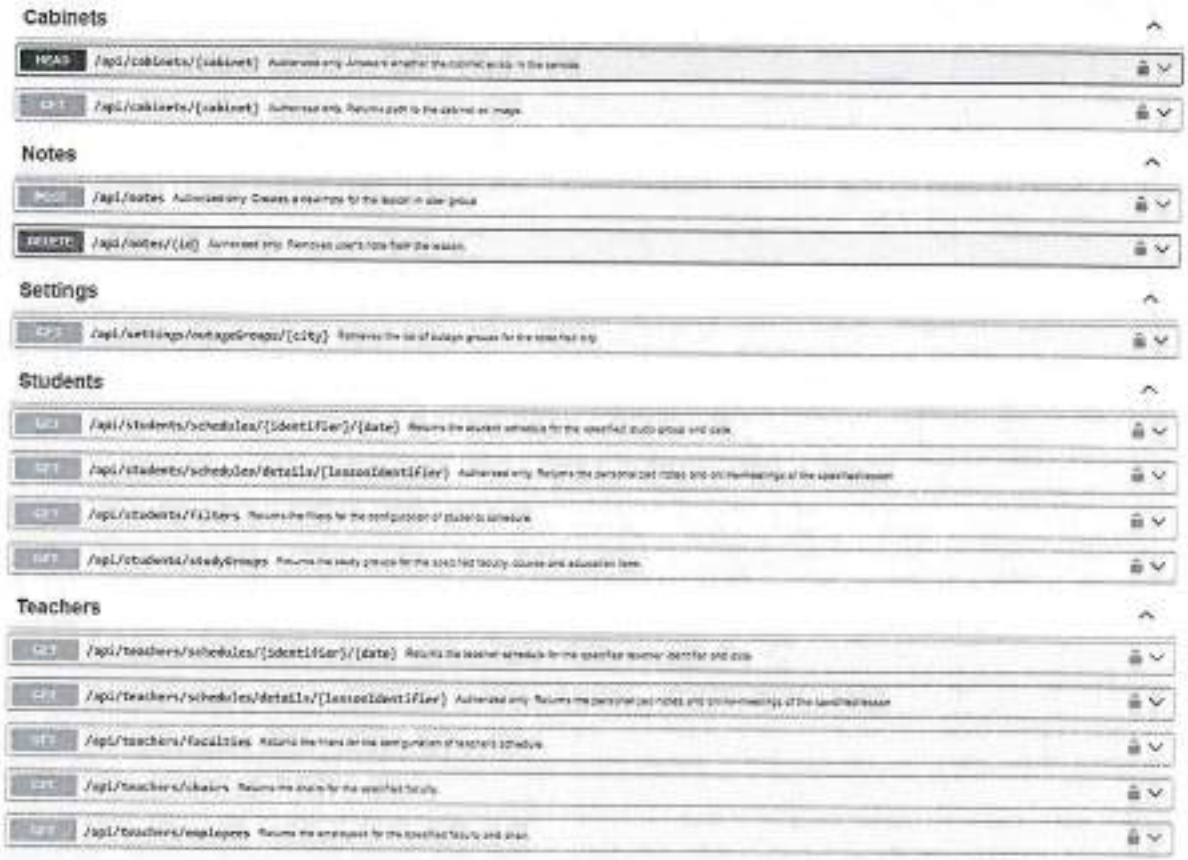

Рисунок 4 – Ендпоінти API у Swagger