Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

Навчально-науковий інститут комп'ютерних наук
та інформаційних технологій
(повне найменування інституту, назва факультету(відділення))

Кафедра комп'ютерних наук
(повна назва кафедри (предметної циклової комісії))

Магістерська кваліфікаційна робота

другий (магістерський)
(рівень вищої освіти)

на тему: «Інтелектуальна цифрова платформа для управління освітнім процесом»

Виконала студентка 6 курсу, групи КН-61
спеціальності:
122 „Комп'ютерні науки"
(шифр і назва напрямку підготовки спеціальності)

Шевчук Оксана Зеновіївна
(прізвище, ім'я, по батькові)

Керівник: Флуд Л. О.
(прізвище, ініціали)

Рецензент: Прусак Ю.В
(прізвище, ініціали)

Львів-2025

Національний лісотехнічний університет України

(повне найменування вищого навчального закладу)

ННІ комп'ютерних наук та інформаційних технологій

Кафедра комп'ютерних наук

Рівень вищої освіти другий (магістерський)

Спеціальність 122 "Комп'ютерні науки"

ЗАТВЕРДЖУЮ:

Завідувачка кафедри КН

_____Борецька І.Б.

„10" грудня 2025 р.

ЗАВДАННЯ

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Шевчук Оксана Зеновіївна

(прізвище, ім'я, по батькові)

1.Тема магістерської роботи: " Інтелектуальна цифрова платформа для управління освітнім процесом"

керівник роботи Флуд Л.О., к.т.н., доцент,

затверджені наказом вищого навчального закладу від "29" квітня 2025 року, № С-288.

2.Термін подання студентом роботи 10 грудня 2025 р.

3. Вихідні дані до роботи: Проєкт реалізувати за допомогою Java, TypeScript, HTML, CSS, SQL.

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Стан проблемної області

Інформаційне забезпечення

Математичне забезпечення

Програмне забезпечення

Розроблення стартап-проєкту

Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Підготовка матеріалу до доповіді.

6. Дата видачі завдання 01.05.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Етапи магістерської роботи	Термін виконання	Відмітка про виконання
1.	Збір та опрацювання потрібних матеріалів	02.05.2025. – 15.05.2025.	виконано
2.	Аналіз поставленої задачі та написання вступу	16.05.2025. – 20.05.2025.	виконано
3.	Вибір та обґрунтування методів та технологій для створення системи.	21.05.2025. – 15.06.2025.	виконано
4.	Виявлення переваг майбутньої системи. Написання першого розділу	16.06.2025. – 15.07.2025.	виконано
5.	Вибір користувацького інтерфейсу. Написання другого розділу	16.07.2025. – 19.08.2025.	виконано
6.	Програмна реалізація системи. Написання третього розділу	20.08.2025. – 15.09.2025.	виконано
7.	Написання четвертого розділу	16.09.2025. – 15.10.2025.	виконано
8.	Аналіз перспектив стартапу та план його подальшого розвитку. Написання п’ятого розділу	16.10.2025. – 15.11.2025.	виконано
9.	Аналіз отриманих результатів та написання висновків	16.11.2025. – 09.12.2025.	виконано
10.	Здача пояснювальної записки на перевірку керівнику, виправлення помилок та здача роботи рецензенту	10.12.2025 р	виконано

Керівник роботи: Флуд Л.О. _____

(підпис)

Студент: Шевчук О.З. _____

(підпис)

АНОТАЦІЯ

Дана дипломна робота містить 72 сторінки поянювальної записки, 34 рисунки, 8 таблиць, 11 формул, 4 додаткки і 12 використаних джерел.

В даній роботі реалізовано платформу для управління освітнім процесом “Schoolify”. В процесі реалізації проведено аналіз стану проблемної області, сформовано чіткий технічний план та розроблено вебзастосунок.

Для розробки вебзастосунку використано мови програмування Java та TypeScript, мову розмітки HTML та мову стилів CSS.

Розроблений вебзастосунок дає змогу ознайомитися з інформацією про учнів, вчителів, викладачів, класи закладів освіти, документи користувачів та навчальних закладів.

***Ключові слова**: вебзастосунок, освітній процес, освіта, цифровізація, учень, Java.*

ABSTRACT

This thesis contains 72 pages of explanatory text, 34 figures, 8 tables, 11 formulas, 4 appendices and 12 references. In this work, the **“**Schoolify**”** platform for managing the educational process has been developed. During the implementation, an analysis of the problem domain was conducted, a clear technical plan was formulated, and a web application was created. The web application was developed using the Java and TypeScript programming languages, as well as HTML for markup and CSS for styling. The developed web application provides access to information about students, teachers, lecturers, school classes, user documents, and educational institution documents.

***Keywords:** web application, educational process, education, digitalization, student, Java.*

ТЕХНІЧНЕ ЗАВДАННЯ

Необхідно розробити інтерактивний вебзастосунок, який забезпечить ефективне управління освітнім процесом у навчальних закладах. Застосунок має надати можливість користувачам (адміністрації, викладачам, учням та батькам) переглядати та взаємодіяти з інформацією про освітній процес, включаючи розклади занять, списки учнів і класів, електронні документи та навчальні матеріали.

Система повинна включати функціонал реєстрації та авторизації користувачів з різними рівнями доступу, особистий кабінет для кожної ролі (учень, викладач, адміністратор), а також модуль адміністрування для керування користувачами, класами, розкладом та внутрішньою документацією. Платформа повинна підтримувати обмін повідомленнями, завантаження документів, автоматичне формування звітів, а також можливість інтеграції з іншими сервісами (наприклад, електронним журналом чи тестовими системами).

Інтерфейс вебзастосунку має бути адаптивним, інтуїтивно зрозумілим і відповідати сучасним стандартам UX/UI дизайну, забезпечуючи комфортну роботу як з комп’ютера, так і з мобільних пристроїв.

Код реалізувати на мовах Java, CSS, HTML та TypeScript в застосунках IntelliJ IDEA і WebStorm.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

SQL (Structured Query Language) – мова структурованих запитів;

HTML (HyperText Markup Language) – мова розмітки гіпертексту;

CSS(Cascading Style Sheets) – мова стилів;

POM (Project Object Model) – модуль Maven – модель об’єкту;

API (Application Programming Interface) – прикладний програмний інтерфейс;

JPA (Java Persistence API / Jakarta Persistence) – стандартизований інтерфейс для Java ORM фреймворків;

XML (Extensible Markup Language) – розширювана мова розмітки.

ВСТУП

Якщо запитати людей, що є найважливішим для майбутнього суспільства, більшість із них відповість — освіта. Освітній процес був і залишається ключовим чинником формування особистості, розвитку суспільства та інноваційного зростання. З розвитком цифрових технологій та глобалізаційних змін виникла гостра потреба у трансформації традиційної системи освіти. Сучасний світ потребує ефективних інструментів, які дозволяють забезпечити якісну, доступну та гнучку освіту. У цьому контексті інтелектуальні цифрові платформи стають основним засобом оптимізації навчального процесу, забезпечення прозорості, автоматизації повсякденних завдань і покращення взаємодії між усіма учасниками освітнього процесу.

Актуальність теми обумовлена потребою в удосконаленні управління освітнім процесом через впровадження сучасних цифрових платформ, які забезпечують автоматизацію документообігу, складання навчальних планів, контроль за виконанням навчальних програм та моніторинг результатів навчання.

Об’єктом дослідження є процес розробки інтелектуальної цифрової платформи для управління освітнім процесом.

Предмет дослідження — засоби, методи та технології розробки вебзастосунку, орієнтованого на автоматизацію управління навчальними закладами, зокрема з використанням Java, SQL, CSS та HTML.

Метою роботи є створення інтелектуальної цифрової платформи, яка дозволяє ефективно керувати навчальним процесом, спрощувати взаємодію між адміністрацією, викладачами, студентами й батьками, а також забезпечити безперервний доступ до актуальної інформації про освітні заходи та результати навчання.

Наукова новизна дослідження полягає у створенні інтегрованої цифрової платформи, орієнтованої на автоматизацію управління освітнім процесом із застосуванням сучасних вебтехнологій. Запропоноване рішення відрізняється комплексним підходом до цифровізації навчального закладу, що охоплює документообіг, планування, контроль виконання навчальних програм та комунікацію

між усіма учасниками освітнього процесу. Особливістю є поєднання функціональності, доступності та гнучкої архітектури, яка дозволяє легко адаптувати систему до потреб конкретного закладу.

Практичне значення розробки полягає в створенні зручного онлайн-інструменту, який дозволяє навчальним закладам автоматизувати ключові освітні та адміністративні процеси, підвищити ефективність управління, зменшити паперову бюрократію та покращити якість надання освітніх послуг.

РОЗДІЛ 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

1.1 Причини актуальності цифровізації освітнього процесу

Цифровізація освітнього процесу є одним із ключових напрямів розвитку сучасної освіти, що обумовлено потребою адаптації навчальних систем до динамічного інформаційного середовища. Впровадження цифрових технологій дозволяє значно підвищити ефективність навчання, зробити його більш індивідуалізованим та доступним, а також оптимізувати управлінські процеси в закладах освіти. Крім того, цифрові інструменти створюють умови для інтерактивної взаємодії між учнями, викладачами та адміністрацією, що сприяє покращенню якості освітніх послуг.

Актуальність цифровізації освітнього процесу також зумовлена глобальними викликами, зокрема необхідністю забезпечення безперервності навчання в умовах надзвичайних ситуацій, таких як пандемія COVID-19. Цифрові технології стали основою для впровадження дистанційного та змішаного навчання, що дозволяє забезпечити доступ до освіти незалежно від географічних чи соціальних обмежень. Таким чином, цифровізація виступає не лише інструментом модернізації освітньої сфери, а й важливою умовою її сталого розвитку та конкурентоспроможності на світовому рівні.

Причини актуальності цифровізації освітнього процесу:

1. Покращення якості освіти

Покращення якості освіти є однією з ключових причин цифровізації освітнього процесу, оскільки цифрові технології дозволяють значно розширити можливості викладання та навчання. Використання інтерактивних платформ, візуалізацій, симуляцій, електронних підручників та онлайн-курсів сприяє кращому засвоєнню матеріалу, посиленню інтересу до навчання та формуванню критичного мислення. Крім того, цифрові інструменти дають змогу вчителям краще диференціювати навчальний матеріал відповідно до рівня знань і потреб учнів, а також здійснювати оперативний зворотний зв’язок, що підвищує ефективність засвоєння знань. Завдяки

цьому навчальний процес стає більш динамічним, гнучким і результативним, що безпосередньо впливає на загальну якість освіти.

2. Доступність освіти

Доступність освіти є важливим аспектом, який цифровізація здатна суттєво покращити. Завдяки впровадженню онлайн-платформ, дистанційного навчання та цифрових ресурсів, освітні послуги стають доступними для широкого кола учнів незалежно від їхнього місця проживання, соціального статусу чи фізичних можливостей. Це особливо актуально для дітей із сільських або віддалених регіонів, осіб з інвалідністю, а також тих, хто навчається у складних соціальних чи економічних умовах. Цифрові технології дозволяють отримувати знання у зручному темпі, у будь-який час і з будь-якого пристрою, що робить освіту більш інклюзивною та відкритою для всіх верств населення. Таким чином, цифровізація виступає потужним інструментом у подоланні освітньої нерівності.

3. Автоматизація адміністративних процесів

Автоматизація адміністративних процесів у закладах освіти є одним із ключових напрямів цифровізації, що сприяє підвищенню ефективності управління та зменшенню навантаження на педагогів. Впровадження електронних журналів, систем електронного документообігу, автоматизованих розкладів, обліку відвідуваності та оцінювання дозволяє значно скоротити час на буденні операції, мінімізувати помилки, пов’язані з людським фактором, і зосередити увагу працівників на педагогічній діяльності. Крім того, такі системи забезпечують прозорість управлінських рішень, полегшують моніторинг навчального процесу та забезпечують оперативний доступ до актуальної інформації для адміністрації, вчителів, учнів і батьків. У результаті автоматизація сприяє підвищенню якості освітнього менеджменту та оптимізації внутрішніх процесів у навчальних закладах.

4. Підготовка до сучасного цифрового світу

Підготовка до сучасного цифрового світу є однією з найважливіших задач освіти в умовах стрімкого технологічного розвитку. Сучасне суспільство

вимагає від людини володіння цифровими навичками, здатності ефективно користуватися інформаційними технологіями, критично мислити та швидко адаптуватися до нових умов. Цифровізація освітнього процесу сприяє формуванню цих компетентностей з раннього віку, забезпечуючи учнів практичними інструментами для роботи з інформацією, онлайн-комунікації, дистанційної співпраці та самонавчання. Освітні платформи, цифрові лабораторії, симуляції та проєктна діяльність готують молодь до реалій сучасного ринку праці, де IT-грамотність є базовою вимогою. Таким чином, цифрове середовище в освіті виконує не лише навчальну, а й соціалізуючу функцію, формуючи покоління, готове до викликів цифрової економіки.

5. Ефективність управління ресурсами

Ефективне управління ресурсами є важливою складовою функціонування сучасного закладу освіти, і цифровізація суттєво підвищує його якість. Завдяки використанню цифрових систем адміністрування можна оптимізувати розподіл матеріальних, фінансових та людських ресурсів, уникати дублювання функцій і своєчасно виявляти потреби в оновленні чи перерозподілі. Наприклад, електронні платформи дозволяють швидко формувати розклади, відстежувати навантаження викладачів, планувати закупівлі та контролювати використання інфраструктурних об’єктів. Також можливість аналітики на основі даних сприяє прийняттю більш обґрунтованих управлінських рішень. У результаті цифрові інструменти роблять управління закладом освіти більш прозорим, гнучким і орієнтованим на результат.

6. Відкритість та прозорість

Відкритість та прозорість освітнього процесу значно посилюються завдяки цифровим технологіям, які забезпечують вільний і зручний доступ до інформації для всіх учасників освітнього середовища. Електронні щоденники, онлайн-журнали, цифрові платформи для комунікації між учителями, учнями та батьками створюють умови для постійного зворотного зв’язку, моніторингу успішності та своєчасного реагування на проблеми. Такі інструменти дозволяють уникати необ’єктивності в оцінюванні, забезпечують контроль за виконанням навчальних планів і сприяють формуванню довіри до системи освіти. Крім того, відкритий доступ до статистичних

даних і звітності підвищує відповідальність адміністрації навчальних закладів і сприяє більш справедливому прийняттю управлінських рішень.

1.2 Проблеми вибору цифрової платформи

У сучасних умовах цифрової трансформації освіти зростає потреба у впровадженні ефективних інструментів для управління освітнім процесом. Цифрові платформи дозволяють автоматизувати буденні адміністративні завдання, організовувати дистанційне та змішане навчання, підвищувати прозорість оцінювання, а також забезпечувати зручний доступ до навчальних матеріалів для всіх учасників освітнього процесу. Проте вибір відповідної цифрової платформи є складним і багатогранним завданням, яке потребує ретельного аналізу технічних, функціональних, юридичних та організаційних аспектів.

Наявність широкого спектра цифрових рішень, які суттєво відрізняються за можливостями, вартістю, умовами впровадження та рівнем технічної підтримки, створює низку викликів для керівників закладів освіти. Особливої уваги потребують питання безпеки персональних даних, інтеграції з існуючими системами та адаптації до специфіки конкретного навчального закладу. У цьому контексті виникає потреба у ґрунтовному вивченні проблем вибору цифрової платформи, що дозволяє забезпечити ефективне та безперебійне управління освітнім процесом.

Основні проблеми вибору цифрової платформи:

1. Невідповідність функціоналу потребам закладу освіти

Однією з ключових проблем при виборі цифрової платформи для управління освітнім процесом є невідповідність її функціоналу реальним потребам конкретного закладу освіти. Більшість готових рішень орієнтовані на універсальний підхід і не враховують особливості організації навчального процесу в різних типах закладів — загальноосвітніх, професійно-технічних, вищих або спеціалізованих. Це може проявлятися у відсутності необхідних модулів (наприклад, інструментів для ведення навчальних планів, складання

розкладу, індивідуального навчання), обмеженій гнучкості налаштувань або незручному інтерфейсі для викладачів та студентів. Як результат, платформа не забезпечує повноцінну підтримку освітнього процесу, що змушує працівників шукати додаткові рішення або повертатися до паперової документації.

2. Нестабільність та обмеження платформи

Нестабільність та обмеження цифрової платформи є суттєвим бар'єром для ефективного управління освітнім процесом. Деякі платформи можуть працювати з перебоями, мати часті технічні збої або оновлюватися без попередження, що ускладнює користування та створює ризики втрати даних. Крім того, безкоштовні або умовно безкоштовні версії часто мають обмеження щодо кількості користувачів, обсягу збереженої інформації чи доступних функцій, що робить їх непридатними для повноцінного використання у великих або багатофункціональних освітніх установах. Такі обмеження знижують загальну ефективність платформи та створюють додаткове навантаження на адміністрацію закладу.

3. Безпека та захист персональних даних

Безпека та захист персональних даних є критично важливими аспектами при виборі цифрової платформи для управління освітнім процесом. Освітні заклади працюють з великим обсягом конфіденційної інформації — персональними даними учнів, студентів, викладачів, а також результатами їх навчальної діяльності. Недостатній рівень захисту може призвести до витоку або несанкціонованого доступу до цих даних, що порушує вимоги законодавства про захист персональних даних і ставить під загрозу довіру користувачів. Тому обрана платформа повинна відповідати чинним нормативам, мати належні механізми шифрування, контролю доступу та аудит безпеки, що забезпечить надійний захист інформації в процесі її зберігання та обробки.

4. Складність впровадження для користувачів

Складність впровадження цифрової платформи для управління освітнім процесом часто пов’язана з недостатнім рівнем технічної підготовки користувачів та незручністю інтерфейсу системи. Викладачі, адміністративний персонал і студенти можуть стикатися з труднощами у засвоєнні нових інструментів через відсутність

чітких інструкцій, локалізації мовою користувачів або підтримки з боку розробників. Це призводить до зниження мотивації до використання платформи, помилок при роботі та необхідності додаткового часу на навчання, що ускладнює повноцінне впровадження системи і негативно впливає на ефективність управління освітнім процесом.

5. Висока вартість впровадження та підтримки

Висока вартість впровадження та подальшої підтримки цифрової платформи є значущою перепоною для багатьох закладів освіти, особливо з обмеженим бюджетом. Окрім початкових витрат на придбання ліцензій або підписку, необхідно враховувати додаткові витрати на налаштування системи, навчання персоналу, технічну підтримку та оновлення програмного забезпечення. Часто ці витрати можуть перевищувати фінансові можливості навчального закладу, що обмежує вибір платформ або змушує шукати компроміси у функціональності та якості обслуговування. Внаслідок цього цифрова трансформація освітнього процесу може затягуватися або реалізовуватися не повною мірою.

Висновки до розділу

Цифровізація освітнього процесу є невід’ємною складовою модернізації сучасної освіти, оскільки вона дозволяє підвищити якість навчання, забезпечити доступність освітніх послуг, автоматизувати управлінські процеси та сформувати необхідні цифрові компетентності в учасників освітнього середовища. Вона виступає не лише інструментом підвищення ефективності, а й відповіддю на глобальні виклики, зокрема забезпечення безперервності навчання в умовах кризових ситуацій. Проте для повноцінної реалізації потенціалу цифровізації необхідно враховувати не лише її переваги, а й складнощі, пов’язані з впровадженням цифрових платформ.

Вибір цифрової платформи для управління освітнім процесом потребує ретельного аналізу, оскільки неправильне рішення може ускладнити, а не полегшити роботу закладу освіти. Проблеми, пов’язані з невідповідністю

функціоналу, нестабільністю роботи, безпекою даних, складністю впровадження та високими витратами, можуть суттєво вплинути на ефективність цифрової трансформації. Тому підхід до вибору платформи має бути комплексним, з урахуванням як технічних і функціональних, так і організаційних можливостей конкретного навчального закладу.

РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Мова програмування Java

Java — це об’єктно-орієнтована мова програмування, створена компанією Sun Microsystems та широко використовувана для розробки вебдодатків, серверних систем, мобільних застосунків (особливо під Android) та корпоративного програмного забезпечення. Її ключовою ідеєю є принцип *“Write once, run anywhere”* — код, написаний на Java, може виконуватись на будь-якій платформі, де є встановлена Java Virtual Machine (JVM). Завдяки цьому Java стала однією з найпоширеніших мов у світі, а її екосистема включає величезну кількість бібліотек, фреймворків та інструментів для розробки [2].

Переваги Java. До основних переваг Java належить висока портативність, стабільність та безпека, що робить її ідеальним вибором для великих корпоративних систем. Вона має потужну систему керування пам’яттю та автоматичне збирання сміття, що значно знижує ризик помилок. Мова також підтримує багатопотоковість, що дозволяє створювати високопродуктивні застосунки. Велика спільнота та численні інструменти, такі як Spring, Maven, Hibernate, IntelliJ IDEA, роблять розробку ефективнішою та зручнішою.

Недоліки Java. Недоліками Java вважають її порівняно високу “ваговитість”: програми на Java можуть споживати більше пам’яті та працювати повільніше, ніж аналоги, написані на більш низькорівневих мовах. Синтаксис Java є досить багатослівним, що іноді ускладнює написання стислого коду. Також запуск на JVM додає додатковий шар абстракції, який може впливати на продуктивність. Попри це, Java продовжує залишатися однією з найнадійніших та найпопулярніших мов програмування завдяки своїй універсальності та перевіреній часом ефективності. [11].

Фреймворки використані при реалізації проєкту:

1. Spring Boot: це фреймворк що значно спрощує запуск і конфігурацію Java-додатків, автоматично налаштовуючи більшість компонентів і залежностей. Він дозволяє швидко створювати готові до роботи сервіси без

складних XML-конфігурацій, використовуючи вбудований сервер (Tomcat/Jetty) [12].

2. Spring Data JPA: це модуль фреймворку Spring, який суттєво спрощує роботу з базами даних, автоматизуючи більшість операцій CRUD та дозволяючи взаємодіяти з даними через інтерфейси репозиторіїв без написання SQL. Він інтегрується з JPA та ORM-інструментами, такими як Hibernate, забезпечуючи гнучке та зручне керування сутностями й запитами.

3. Spring Web: модуль Spring Framework, який забезпечує створення вебдодатків і REST-контролерів, обробку HTTP-запитів та маршрутизацію. Він дозволяє легко реалізовувати API, керувати параметрами, заголовками та статусами відповідей.

4. Spring Security: це потужний фреймворк для захисту застосунків, який забезпечує автентифікацію, авторизацію та фільтрацію запитів. Він дозволяє налаштовувати ролі, права доступу, валідацію користувачів і захист від типових атак (CSRF, Brute Force тощо).

5. JWT Token (JSON Web Tokens): це компактний криптографічно підписаний токен, який передає інформацію про користувача між клієнтом і сервером у безпечній формі. Його використовують для авторизації: після входу користувач отримує токен і надалі надсилає його з кожним запитом, що дозволяє серверу працювати без збереження сесій.

6. Redis Cache — це швидкий in-memory кеш (працює в оперативній пам’яті), який використовується для тимчасового зберігання часто запитуваних даних з мінімальною затримкою, щоб зменшити навантаження на базу даних і прискорити роботу застосунків; він зберігає дані у форматі key–value, підтримує різні структури (string, hash, list, set тощо), має вбудований механізм TTL (час життя записів) і автоматичне очищення, добре масштабується та підходить для розподілених систем, але не є заміною основної бази даних і повинен використовуватися саме як кеш або тимчасове сховище.

7. RabbitMQ — це брокер повідомлень (message broker), який використовується для асинхронного обміну повідомленнями між різними частинами системи: він приймає повідомлення від продюсерів, зберігає їх у чергах і доставляє споживачам, дозволяючи розвантажити основні сервіси, підвищити масштабованість і надійність

застосунків; RabbitMQ працює за протоколом AMQP, підтримує маршрутизацію через exchange-и, гарантії доставки, acknowledge-повідомлення, повторні спроби, відкладені та відмовостійкі повідомлення, і широко використовується в мікросервісних архітектурах для фонового виконання задач, інтеграції сервісів та обробки подій.

8. OpenAI API — це програмний інтерфейс, який надає доступ до великих мовних і мультимодальних моделей штучного інтелекту для інтеграції в застосунки, сервіси та бекенд-системи; за допомогою OpenAI API можна обробляти та генерувати текст, аналізувати документи, працювати із зображеннями й файлами, виконувати семантичний пошук, узагальнення, переклад, класифікацію, а також створювати інтелектуальних асистентів і автоматизувати бізнес-процеси, при цьому взаємодія відбувається через HTTP-запити з чіткою структурою, що дозволяє легко використовувати API з різних мов програмування.

9. Spring Data MongoDB — це модуль екосистеми Spring, який спрощує роботу з документоорієнтованою базою даних MongoDB у Java-застосунках: він надає репозиторії з готовими CRUD-операціями, автоматичне мапінґування Java-об’єктів у BSON-документи, підтримує запити через методи інтерфейсів або анотації, агрегації, індекси й реактивну модель доступу до даних, що дозволяє швидко та зручно взаємодіяти з MongoDB без написання великої кількості низькорівневого коду.

10. OpenID Connect (OIDC) — це протокол автентифікації, побудований на основі OAuth 2.0, який дозволяє безпечно підтверджувати особу користувача у веб або мобільних застосунках. Він додає до OAuth 2.0 спеціальний «ID токен», що містить інформацію про користувача (наприклад, ім’я, електронну пошту, унікальний ідентифікатор), і дозволяє застосункам отримувати цю інформацію без необхідності зберігати пароль користувача. OIDC забезпечує стандартизований спосіб аутентифікації та одночасно підтримує делегований доступ до ресурсів через OAuth 2.0, що робить його популярним вибором для єдиного входу (SSO) та інтеграції з різними сервісами.

11. Keycloak — це відкритий сервер ідентифікації та управління доступом, який реалізує протоколи OIDC, OAuth 2.0 і SAML. Він дозволяє організаціям централізовано керувати користувачами, ролями, групами та правами доступу до різних застосунків, забезпечуючи одночасно автентифікацію та авторизацію. Keycloak підтримує єдиний вхід (SSO), соціальні логіни (Google, Facebook тощо), багатофакторну автентифікацію, а також інтеграцію з корпоративними каталогами (LDAP, Active Directory). Його можна розгорнути як окремий сервер або у контейнері, і він надає REST API та адаптери для легкого підключення до веб та мобільних застосунків.

2.2 Мова програмування TypeScript

TypeScript — це мова програмування, створена як надбудова над JavaScript, що додає статичну типізацію та розширює можливості стандартного JS. Вона компілюється у звичайний JavaScript, завдяки чому може працювати в будь-якому середовищі, де JS вже підтримується — у браузерах, серверних платформах на кшталт Node.js та гібридних мобільних додатках. TypeScript робить розробку більш передбачуваною, дозволяючи чітко визначати структуру даних та запобігати помилкам ще до запуску програми [9].

Крім типів, TypeScript надає сучасний синтаксис та можливості, які часто випереджають офіційні стандарти JavaScript. Він підтримує дженерики, інтерфейси, модулі, покращену роботу з класами та інші інструменти, які роблять код організованішим і зручнішим у масштабуванні. Завдяки цьому мова особливо популярна у великих командах та корпоративних проєктах, де потрібна висока структурованість та передбачуваність [5].

Переваги TypeScript. Головною перевагою TypeScript є статична типізація, яка дозволяє уникати безлічі помилок на ранніх етапах розробки та робить код більш надійним і легшим у підтримці. Крім того, вона забезпечує кращу інтеграцію з інструментами розробки — автодоповнення, підказки типів та автоматичний рефакторинг працюють більш точно, що значно підвищує ефективність роботи.

Недоліки TypeScript. Серед недоліків TypeScript виділяють необхідність додаткового процесу компіляції та складність початкової конфігурації проєкту. Також розробники часто витрачають більше часу на опис типів, а складні системи типізації можуть бути незрозумілими для новачків. Хоча ці мінуси існують, у практиці вони зазвичай перекриваються перевагами, які TypeScript приносить у довгостроковій перспективі[8].

2.3 Мова стилів CSS

CSS - це мова стилів, призначена для візуального оформлення вебсторінок і вебдодатків. Вона дозволяє описувати вигляд HTML-елементів: кольори, шрифти, відступи, розміри, розташування та анімації. CSS відокремлює структуру сторінки від її презентації, що робить код більш організованим і зручним у підтримці. Завдяки цьому розробники можуть створювати сучасні, гнучкі та адаптивні інтерфейси.

До особливостей CSS належать каскадність, специфічність та успадкування правил — ці механізми визначають, як саме браузер застосовує різні стилі до елементів. CSS також підтримує медіа-запити, що дозволяють створювати адаптивний дизайн для різних розмірів екрану, та сучасні функції, такі як Flexbox, Grid, змінні (custom properties) і анімації. Такі можливості роблять мову надзвичайно потужною для верстки вебінтерфейсів будь-якої складності.

Основними перевагами CSS є легкість у вивченні, можливість створення адаптивних і привабливих інтерфейсів без використання JavaScript, а також висока продуктивність роботи в браузерах. Крім того, CSS дозволяє розділяти дизайн та логіку, що спрощує командну роботу та робить код більш чистим. Широка підтримка у всіх браузерах і постійний розвиток стандартів роблять CSS невід’ємною частиною веброзробки.

До недоліків CSS можна віднести складність управління стилями у великих проєктах через каскадність і специфічність — конфлікти стилів зустрічаються досить часто. Також мова не має повноцінної логіки, тому іноді

базових можливостей недостатньо, і доводиться використовувати препроцесори (SASS, LESS) або додаткові інструменти. У різних браузерах інколи виникають відмінності у відображенні, що потребує додаткової перевірки та коригування. Попри ці недоліки, CSS залишається ключовим інструментом для створення сучасного вебдизайну [4].

2.4 Мова розмітки HTML

HTML - це мова розмітки, яка визначає структуру та зміст вебсторінок. Вона описує елементи сторінки за допомогою тегів: заголовки, абзаци, зображення, кнопки, форми, таблиці та інші компоненти інтерфейсу. HTML не є мовою програмування — він не виконує логіку, а лише задає каркас, на якому будуються стилі (CSS) та функціональність (JavaScript). Завдяки своїй простоті HTML став базою для всієї веброзробки.

Особливістю HTML є його семантичність — наявність тегів, які не лише відображаються, а й описують зміст: <header>, <article>, <nav>, <footer> тощо. Завдяки цьому пошукові системи краще розуміють структуру сайту, а користувачі, включно з людьми, які використовують спеціальні допоміжні технології, отримують більш доступний контент. HTML також підтримує мультимедійні елементи, інтеграцію скриптів та можливість роботи з формами для взаємодії з сервером [1].

Переваги HTML полягають у його простоті, легкому вивченні та універсальності — він працює у всіх браузерах без додаткових налаштувань. HTML дозволяє швидко створювати структуру сторінки, легко інтегрується з CSS і JavaScript, та є абсолютно безкоштовним і відкритим стандартом. Постійний розвиток (HTML5) додав підтримку відео, аудіо, Canvas, локального збереження даних і семантичних тегів, що значно розширило його можливості.

До недоліків HTML належать відсутність логічних конструкцій та обмежена функціональність — без CSS і JavaScript він не може створювати динамічні або стильні інтерфейси. Також HTML схильний до хаосу в руках недосвідчених розробників: неправильна структура та відсутність семантики ускладнюють

підтримку проєкту. Попри це, HTML залишається фундаментом вебу, без якого не працював би жоден сучасний сайт.

У загальному підсумку , HTML є базовою та незамінною складовою веброзробки, оскільки забезпечує структуру і фундамент для побудови будь-якої вебсторінки. Завдяки своїй простоті, універсальності та здатності легко поєднуватися з іншими технологіями він залишається ключовим інструментом для розробників на всіх етапах навчання й роботи. Опановування HTML стає першим важливим кроком у створенні власних сайтів та у подальшому розумінні складніших вебтехнологій [5].

Висновки до розділу

У ході аналізу мов програмування та технологій, використаних у проєкті, можна зробити висновок, що кожна з них відіграє важливу роль у створенні сучасних, масштабованих та безпечних програмних рішень. **Java** забезпечує надійність і кросплатформенність для бекенд-розробки завдяки потужним фреймворкам, таким як Spring Boot, які суттєво спрощують конфігурацію, безпеку та роботу з базами даних. **TypeScript**, у свою чергу, додає до JavaScript статичну типізацію, що покращує якість коду і зменшує кількість помилок, особливо у великих фронтенд-проєктах. Використання **CSS** та **HTML** забезпечує структуровану та візуально привабливу презентацію вебінтерфейсів, що є необхідним для ефективної взаємодії користувача з системою.

Комбінація цих технологій дозволяє створювати повноцінні вебдодатки з чітким поділом відповідальностей між фронтендом і бекендом. Такий підхід забезпечує високу якість розробки, полегшує підтримку проєкту та дозволяє ефективно масштабувати систему відповідно до зростаючих вимог.

РОЗДІЛ 3. МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Загальний алгоритм імпортування даних у форматі Excel

1. *Завантаження файлу в систему.* Процес імпорту починається з того, що користувач надсилає файл Excel Fчерез вебінтерфейс за допомогою HTTP POST-запиту. Файл може містити декілька таблиць:

$$F = \{T_1, T_2, \dots, T_k\}, \quad T_i \in R^{n_i * m_i} \tag{3.1}$$

де T_i - це окрема таблиця з n_i рядків і m_i колонок. Кожна таблиця містить структуровані дані, які потрібно перевірити та зберегти у базі даних. Використання HTTP POST дозволяє передавати великі файли і забезпечує сумісність з вебінтерфейсами.

2. *Валідація користувача на права для імпорту.* Перед обробкою файлу система перевіряє, чи має користувач права на імпорт даних. Це необхідно для забезпечення безпеки та контролю доступу до критичних даних. Якщо користувач не має достатніх прав, процес імпорту припиняється, і надсилається повідомлення про помилку доступу.

3. *Валідація таблиць та їх вмісту.* Кожна таблиця перевіряється на відповідність підтримуваним форматам у системі. Це включає:

- перевірку унікальності значень ключових колонок,
- відповідність типів даних у колонках (числові, текстові, дати тощо),
- перевірку на наявність обов’язкових полів.

4. *Повідомлення користувачу про стан імпорту*. Після завершення початкової перевірки система надсилає повідомлення користувачу про початок імпортування або про виявлені помилки. Це повідомлення відправляється у рамках HTTP-відповіді, що дозволяє завершити синхронний запит і звільнити з’єднання для подальших операцій.

5. *Перехід в асинхронний режим обробки*. Для обробки великих файлів імпорт відбувається асинхронно. Усі рядки таблиць передаються у чергу обробки Q, яка забезпечує незалежну та послідовну роботу з даними:

$$Q = \{r_1, r_2, \dots, r_n\}, \quad r_i \in \bigcup T_k \tag{3.2}$$

6. *Обробка рядків та акумуляція у структуру даних*. Кожен рядок з черги обробляється послідовно і зберігається у зв'язний список:

$$L = [r_1 \to r_2 \to \cdots \to r_n] \quad (3.3)$$

Ця структура дозволяє ефективно виконувати додаткові операції над даними перед їх остаточним збереженням, наприклад, сортування, фільтрацію або перевірку додаткових умов.

7. *Зберігання даних у базі.* Кожен рядок зберігається у базі даних у окремій транзакції, що дозволяє забезпечити цілісність даних і уникнути втрати інформації у випадку помилок. Файл імпорту разом зі статусом та тривалістю процесу фіксується у MongoDB для подальшого аудиту та відстеження історії імпортів.

8. *Повідомлення користувачу про завершення імпорту.* Після завершення обробки всіх рядків користувачу надсилається повідомлення через WebSocket про успішність імпорту. Це дозволяє отримати миттєвий зворотний зв'язок у вебінтерфейсі без необхідності оновлювати сторінку.

3.2 Алгоритм фільтрування даних по освітніх закладах

1. *Формування запиту користувача.* Процес починається з того, що користувач надсилає запит на отримання даних, наприклад, список користувачів певних освітніх закладів. Формально запит можна подати у вигляді функції:

$$R(u)=getUsers(u,C(u)) \quad (3.4)$$

де u - це поточний користувач системи, а $C(u)$ - область видимості компаній або освітніх закладів, на які користувач має доступ. Тобто, запит повертає тільки ті дані, які користувач має право бачити відповідно до налаштувань доступу та ролей у системі.

2. *Отримання списку освітніх закладів через кеш.* Для підвищення продуктивності система спершу звертається до Redis Cache на серверній частині, щоб отримати список ідентифікаторів закладів, доступних користувачу. Якщо кеш повертає порожній масив, виконується запит на

основний сервіс закладів, де отримується актуальний список id компаній. Після цього список зберігається в кеші для майбутніх запитів, що мінімізує повторні звернення та зменшує навантаження на сервер. Формально це можна записати так:

$$C(u) = \begin{cases} Redis(u), & \text{якщо } Redis(u) \neq 0 \\ API(u), & \text{інакше } Redis(u) \leftarrow API(u) \end{cases} \quad (3.5)$$

Таким чином, кешування дозволяє забезпечити високу швидкість відповіді на запити та ефективне використання ресурсів сервера.

3. *Визначення області видимості компаній.* Область видимості компаній може задаватися в кожному сервісі окремо, залежно від бізнес-вимог або політик безпеки. Умовно, ієрархія освітніх закладів може бути подана у вигляді дерева: $G = (V, E)$, де V - множина компаній або закладів, а E - зв'язки між ними (типу "батьківська-дочірня компанія"). Для кожного користувача система визначає підмножину вузлів дерева, яка становить його область видимості:

$$C(u) = Scope_{type}(v_i) \quad (3.6)$$

де $v_i \in V$ - поточна компанія користувача, а $Scope_{type}$ - політика видимості (наприклад, всі нащадки, один рівень вниз, усі батьківські вузли тощо). Це дозволяє гнучко управляти доступом та забезпечує відповідність даних бізнес-логіці компанії.

4. *Фільтрування даних та безпека запитів.* Отриманий список id компаній інтегрується в SQL-запит для отримання конкретних даних (наприклад, користувачів або груп) лише з тих закладів, на які користувач має доступ. Такий підхід гарантує безпеку даних, оскільки ніяка інформація поза областю видимості користувача не буде повернута. Крім того, фільтрування на рівні SQL забезпечує масштабованість системи, дозволяючи ефективно працювати з великими обсягами даних, і відкриває додаткові можливості для реалізації складних функцій, таких як агрегації, сортування або звітність по групах освітніх закладів.

3.3 Алгоритм оцінювання завдання учня

1. *Структура завдання та задач.* Кожне завдання складається з набору окремих задач, де кожна задача має свою оцінку та ваговий коефіцієнт, що визначає її внесок у підсумкову оцінку. Формально завдання можна подати як множину задач:

$$Z = \{z_1, z_2, \dots, z_n\} \tag{3.7}$$

де z_i - це окрема задача. Для кожної задачі визначається фактична оцінка учня s_i та її вага ω_i, яка показує важливість задачі щодо всього завдання.

2. *Обчислення внеску кожної задачі у підсумкову оцінку.* Оцінка кожної задачі не використовується безпосередньо, а множиться на її вагу. Це дозволяє більш складним або більш важливим задачам мати більший вплив на фінальний результат. Математично внесок окремої задачі визначається так:

$$\mathrm{C}_i = s_i \times \omega_i \tag{3.8}$$

де C_i - зважена оцінка задачі, s_i - фактична оцінка за задачу, w_i - ваговий коефіцієнт.

3. *Формування підсумкової оцінки за завдання.* Підсумкова оцінка учня за завдання визначається як сума всіх зважених оцінок задач. Таким чином, фінальна оцінка є агрегатом усіх результатів, з урахуванням їх ваг. Формально фінальна оцінка розраховується за формулою:

$$S = \sum_{i=1}^{n}(s_i \times \omega_i) \tag{3.9}$$

де S - загальна оцінка за завдання. Такий підхід забезпечує справедливий розподіл оцінювання, при якому сильніші або важливіші задачі роблять більший внесок у підсумковий результат.

4. *Нормалізація та варіації алгоритму (опціонально).* У деяких випадках система може вимагати нормалізації ваг, наприклад, якщо сума всіх ваг повинна дорівнювати 1, або якщо потрібно привести всі оцінки до певного діапазону. У такому випадку ваги можуть нормуватися:

$$\omega_i' = \frac{\omega_i}{\sum_{j=1}^{n}\omega_i} \tag{3.10}$$

і тоді формула набуває вигляду:

$$S = \sum_{i=1}^{n}(s_i \times \omega_i') \quad (3.11)$$

Це використовується у випадках стандартизації або коли потрібно дати можливість порівнювати оцінки між різними типами завдань (додаток Г).

Висновки до розділу

У цьому розділі було представлено математичне та алгоритмічне підґрунтя ключових процесів системи автоматизації освітнього менеджменту. Розглянуті моделі демонструють цілісність та узгодженість логіки роботи сервісів, забезпечуючи надійну, масштабовану та безпечну обробку даних.

Алгоритм імпорту даних у форматі Excel описує повний цикл обробки файлу — від моменту завантаження та первинної валідації до асинхронної обробки записів і фіксації результатів у базі даних. Запропоновані математичні моделі відображають структуру даних, механізм черги та спосіб накопичення інформації перед збереженням, що гарантує цілісність, контроль доступу та можливість обробки великих обсягів інформації без втрати продуктивності системи.

Алгоритм фільтрування даних за освітніми закладами забезпечує коректну та безпечну вибірку інформації відповідно до прав доступу користувача. Формалізація процесу через множини доступних закладів, кешування в Redis та ієрархічну структуру організацій дозволяє ефективно реалізувати механізми контролю доступу та оптимізувати виконання запитів у розподіленій архітектурі.

Алгоритм оцінювання завдання учня формує математично обґрунтовану модель зваженого підрахунку результатів, що забезпечує справедливість та прозорість оцінювання. Використання вагових коефіцієнтів дозволяє варіювати вплив окремих задач на підсумковий результат, а можливість нормалізації ваг робить систему гнучкою та адаптованою до різних типів навчальних сценаріїв.

Загалом, наведене математичне забезпечення створює фундамент для коректної, безпечної та ефективної роботи програмного комплексу, забезпечуючи його надійність, масштабованість та відповідність сучасним вимогам до інформаційних систем у сфері освіти.

РОЗДІЛ 4. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

4.1. Середовища розробки платформи

IntelliJ IDEA - це потужне середовище розробки програмного забезпечення, створене компанією JetBrains. Воно широко використовується для написання коду на різних мовах програмування, таких як Java, Kotlin, Groovy та інших. IntelliJ IDEA допомагає розробникам писати код швидше і з меншими помилками завдяки інтелектуальному редактору, який автоматично підказує, доповнює та виправляє код.

Однією з ключових особливостей IntelliJ IDEA є інтеграція з системами контролю версій, такими як Git, що дозволяє легко керувати змінами в коді. Середовище також підтримує безліч плагінів, які розширюють його функціональність, та працює з популярними фреймворками і технологіями, такими як Spring і Hibernate.

Завдяки цим можливостям IntelliJ IDEA є популярним вибором серед розробників, пропонуючи зручні інструменти для створення якісного програмного забезпечення. Це середовище значно полегшує процес розробки, роблячи його більш ефективним і приємним [10].

WebStorm - це інтегроване середовище розробки (IDE), розроблене компанією JetBrains, яке спеціалізується на веброзробці. Воно підтримує такі мови програмування, як JavaScript, TypeScript, HTML, і CSS, а також популярні фреймворки, такі як React, Angular, Vue.js та Node.js. WebStorm допомагає розробникам писати код швидше і з меншими помилками завдяки інтелектуальному редактору, який надає автодоповнення, перевірку синтаксису та інші корисні підказки.

Однією з ключових переваг WebStorm є його потужна інтеграція з системами контролю версій, такими як Git і SVN. Це дозволяє розробникам легко керувати своїм кодом, відстежувати зміни та співпрацювати з іншими членами команди. Крім того, WebStorm підтримує численні плагіни, які

розширюють функціональність IDE та дозволяють налаштувати його під конкретні потреби проєкту.

WebStorm також пропонує зручні інструменти для налагодження та тестування коду, що робить процес розробки більш ефективним. Завдяки своїм можливостям та зручному інтерфейсу, WebStorm є відмінним вибором для веброзробників, які прагнуть створювати якісні та масштабовані вебдодатки.

Docker Desktop — це офіційний застосунок для Windows та macOS, який надає зручний графічний інтерфейс і набір інструментів для роботи з контейнерами Docker. Він включає Docker Engine, Docker CLI, Docker Compose та Kubernetes (опційно), дозволяючи розробникам створювати, запускати та керувати контейнеризованими застосунками локально. Docker Desktop автоматично налаштовує віртуалізацію, мережу та зберігання даних для контейнерів, спрощує тестування багатокомпонентних систем і інтеграцію з CI/CD. Це зручний інструмент для розробки, відлагодження та навчання контейнеризації без потреби вручну налаштовувати серверні інстанси Docker.

Jenkins — це відкритий сервер автоматизації, який використовується для безперервної інтеграції (CI) та безперервного розгортання (CD) програмного забезпечення. Він дозволяє автоматизувати збірку, тестування та доставку застосунків, інтегруючись з різними системами контролю версій (Git, SVN), інструментами для тестування та середовищами розгортання. Jenkins підтримує розширювану архітектуру плагінів, що дозволяє додавати функціональність під специфічні потреби проєкту, такі як нотифікації, аналіз коду або інтеграція з хмарними сервісами. Завдяки вебінтерфейсу та можливості налаштовувати пайплайни як код (Jenkinsfile), він робить процес розробки більш надійним, повторюваним і контрольованим.

4.2 Структура бази даних

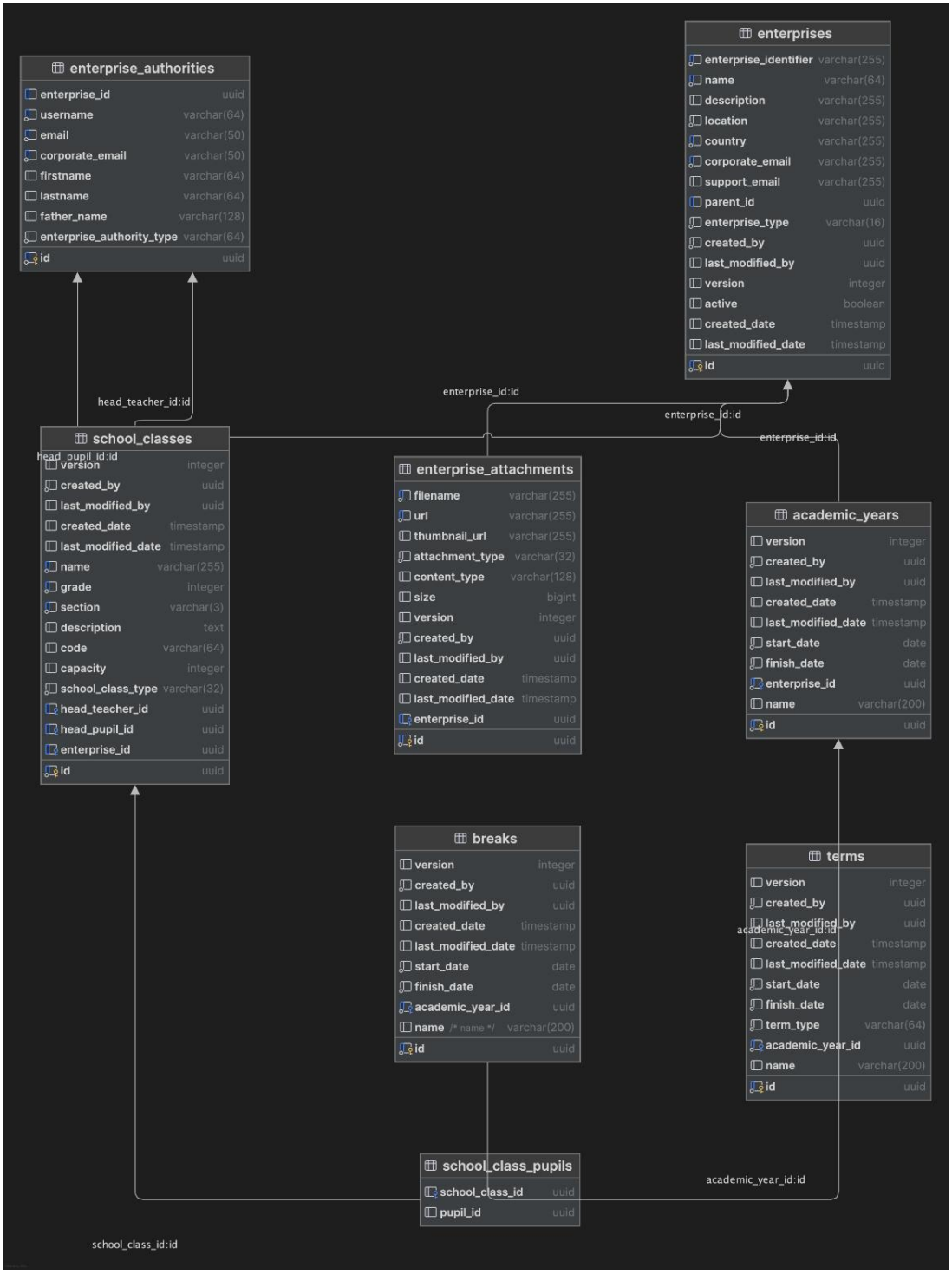


Рисунок 4.1 - ER-Діаграма бази даних закладів

enterprises - ця сутність зберігає інформацію про навчальні заклади . Атрибути цієї сутності включають:

- name: varchar(64): назва;
- support_email: varchar(50): emeil технічної підтримки;
- corporate_email: varchar(50): корпоративний emeil;

- Enterprise_identification: varchar(64): користувацький ідентифікатор закладу;
- active: boolean стан закладу;
- version: integer: версія;
- created_by: uuid: ким створенно;
- last_modified_by: uuid: ким змінено;
- created_date: timestamp: дата створення;
- last_modified_date: timestamp: дата зміни;
- description: text опис закладу;
- enterprise_type: varchar(64): тип закладу;
- Location: varchar(64): місцезнаходження;
- Country: varchar(32): країна;
- id: uuid (PK): ідентифікатор таблиці.

enterprise_authorities (рисунок 4.1) **-** ця сутність зберігає інформацію про посади користувачів. Атрибути цієї сутності включають:

- enterprise_id: uuid (FK → enterprise_data.id): id закладу;
- username: varchar(64): користувач;
- email: varchar(50): email;
- corporate_email: varchar(50): корпоративний email;
- firstname: varchar(64): ім’я;
- lastname: varchar(64) прізвище;
- father_name: varchar(128) по батькові;
- enterprise_authority_type: varchar(64): тип посади;
- id: uuid (PK): ідентифікатор таблиці.

school_classes - ця сутність зберігає інформацію про шкільні класи. Атрибути цієї сутності включають:

- version: integer: версія;
- created_by: uuid: ким створенно;
- last_modified_by: uuid: ким змінено;

- created_date: timestamp: дата створення;
- last_modified_date: timestamp: дата зміни;
- name: varchar(255): назва;
- grade: integer: ступінь;
- section: varchar(3): секція;
- description: text: опис;
- code: varchar(64): код;
- capacity: integer: місткість;
- school_class_type: varchar(32): тип класу;
- head_teacher_id: uuid (FK → enterprise_authorities.id): айді класного керівника;
- head_pupil_id: uuid (FK → pupils.id): id старости;
- enterprise_id: uuid (FK → enterprise_data.id): id закладу;
- id: uuid (PK): ідентифікатор таблиці.

enterprise_attachments - ця сутність зберігає інформацію про документи навчального закладу. Атрибути цієї сутності включають:

- filename: varchar(255): назва файлу;
- url: varchar(255): посилання на файл;
- thumbnail_url: varchar(255): посилання на стиснутий файл;
- attachment_type: varchar(32): тип файлу;
- content_type: varchar(128): тип контенту;
- size: bigint: розмір;
- version: integer: версія;
- created_by: uuid: ким створено;
- last_modified_by: uuid: ким змінено;
- created_date: timestamp: дата створення;
- last_modified_date: timestamp: дата зміни;
- enterprise_id: uuid (FK → enterprise_data.id): id закладу;
- id: uuid (PK): ідентифікатор таблиці.

Breaks - ця сутність зберігає інформацію про канікули. Атрибути цієї сутності включають:

- version: integer: версію;
- created_by: uuid: ким створено;
- last_modified_by: uuid: ким змінено;
- created_date: timestamp: дата створення;
- last_modified_date: timestamp: дата зміни;
- start_date: date: початок канікул;
- finish_date: date: кінець канікул;
- academic_year_id: uuid (FK → academic_years.id): id навчального року;
- name: varchar(200): назва;
- id: uuid (PK): ідентифіктор таблиці .

school_class_pupils (Junction таблиця) **-** ця сутність зберігає інформацію про учнів класу. Атрибути цієї сутності включають:

- school_class_id: uuid (FK → school_classes.id): id класу;
- pupil_id: uuid (FK → pupils.id): id учня.

academic_years - ця сутність зберігає інформацію про навчальні роки. Атрибути цієї сутності включають:

- version: integer: версія;
- created_by: uuid: ким створено;
- last_modified_by: uuid: ким змінино;
- created_date: timestamp: дата створення;
- last_modified_date: timestamp: дата зміни;
- start_date: date: початок навчального року;
- finish_date: date: кінець навчального року;
- enterprise_id: uuid (FK → enterprise_data.id): id закладу;
- name: varchar(200): назва;
- id: uuid (PK): індентифікатор таблиці.

terms - ця сутність зберігає інформацію про семестри. Атрибути цієї сутності включають:

- version: integer: версія;
- created_by: uuid: ким створено;
- last_modified_by: uuid: ким змінено;
- created_date: timestamp: дата створення;
- last_modified_date: timestamp: дата створення;
- start_date: date: початок семестру;
- finish_date: date: кінець семестру;
- term_type: varchar(64): тип семестру;
- academic_year_id: uuid (FK → academic_years.id): id навчального року;
- name: varchar(200): назва;
- id: uuid (PK): індентифікатор таблиці.

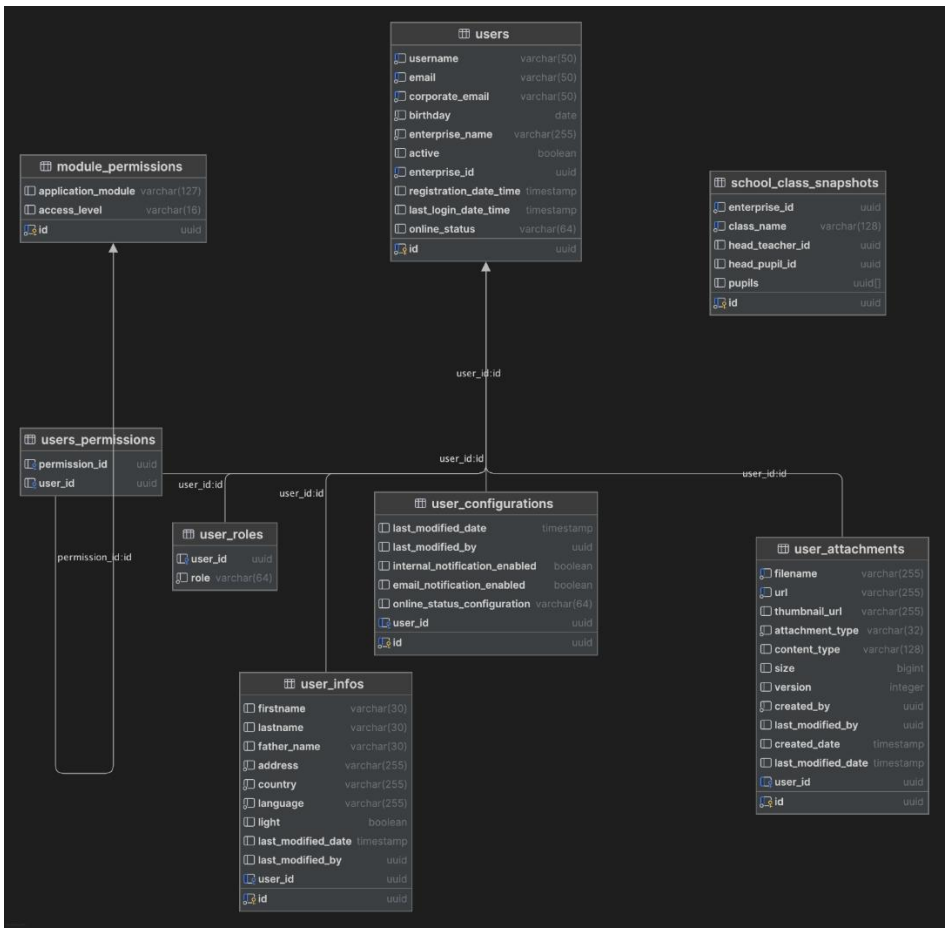


Рисунок 4.2 - ER-Діаграма бази даних користувачів

Users - ця сутність (рисунок 4.2) зберігає інформацію про користувачів. Атрибути цієї сутності включають:

- enterprise_id: uuid (FK → enterprise_data.id): id закладу;
- Enterprise_name: varchar(255) назва закладу;
- Birthday: date: дата народження;
- Registration_date_time: timestamp: дата реєстрації;
- Last_login_date_time: timestamp: дата останнього входу в систему;
- Online_status: varchar(64): мережевий статус;
- username: varchar(64): користувач;
- email: varchar(50): email;
- corporate_email: varchar(50): корпоративний email;
- enterprise_authority_type: varchar(64): тип посади;
- id: uuid (PK): ідентифікатор таблиці.

module_permissions - ця сутність зберігає інформацію про доступи до модулів. Атрибути цієї сутності включають:

- application_module: varchar(127): модуль програми;
- access_level: varchar(16): рівень доступу;
- id: uuid (PK): індентифікатор таблиці.

users_permissions (Junction таблиця) - ця сутність зберігає інформацію про зв`язок між користувачами та дозволами. Атрибути цієї сутності включають:

- permission_id: uuid (FK → module_permissions.id): id модуля;
- user_id: uuid (FK → users.id): id користувача.

user_roles - ця сутність зберігає інформацію про ролі користувача. Атрибути цієї сутності включають:

- user_id: uuid (FK → users.id): id користувача;
- role: varchar(64): назва ролі.

users_configurations - ця сутність зберігає інформацію про настройки користувача. Атрибути цієї сутності включають:

- last_modified_date: timestamp: остання дата зміни;

- last_modified_by: uuid: ким змінино;
- internal_notification_enabled: boolean: увімкнення внунтрішніх сповіщень;
- email_notification_enabled: boolean: увімкнення емейл сповіщень;
- online_status_configuration: varchar(64): відображення онлайн статусу;
- user_id: uuid (FK → users.id): id користувача.

user_info - ця сутність зберігає інформацію про відомості користувача. Атрибути цієї сутності включають:

- firstname: varchar(30): ім’я;
- lastname: varchar(30): прізвище;
- father_name: varchar(30): по батькові;
- address: varchar(255): адреса;
- country: varchar(255): країна;
- language: varchar(255): мова;
- light: boolean: світла чи темна тема;
- last_modified_date: timestamp: остання дата зміни;
- last_modified_by: uuid: ким змінено;
- user_id: uuid (FK → users.id): id користувача.

school_class_snapshots - ця сутність зберігає інформацію про шкільні класи. Атрибути цієї сутності включають:

- enterprise_id: uuid (FK → enterprise_data.id): id закладу;
- class_name: varchar(128): назва класу;
- head_teacher_id: uuid (FK → users.id): id класного керівника;
- head_pupil_id: uuid (FK → users.id): id старости;
- pupils: uuid (FK → users.id, масив): id учнів;
- id: uuid (PK): індетифікатор таблиці.

user_attachments - ця сутність зберігає інформацію про документи користувача. Атрибути цієї сутності включають:

- filename: varchar(255): назва файлу;
- url: varchar(255): посилання;

- thumbnail_url: varchar(255): посилання на стиснутий варіант;
- attachment_type: varchar(32): тип;
- content_type: varchar(128): тип контенту;
- size: bigint: розмір;
- version: integer: версія;
- created_by: uuid: ким створено;
- last_modified_by: uuid ким змінено;
- created_date: timestamp: дата створення;
- last_modified_date: timestamp: дата зміни;
- user_id: uuid (FK → users.id): id користувача;
- id: uuid (PK): індетифікатор таблиці.

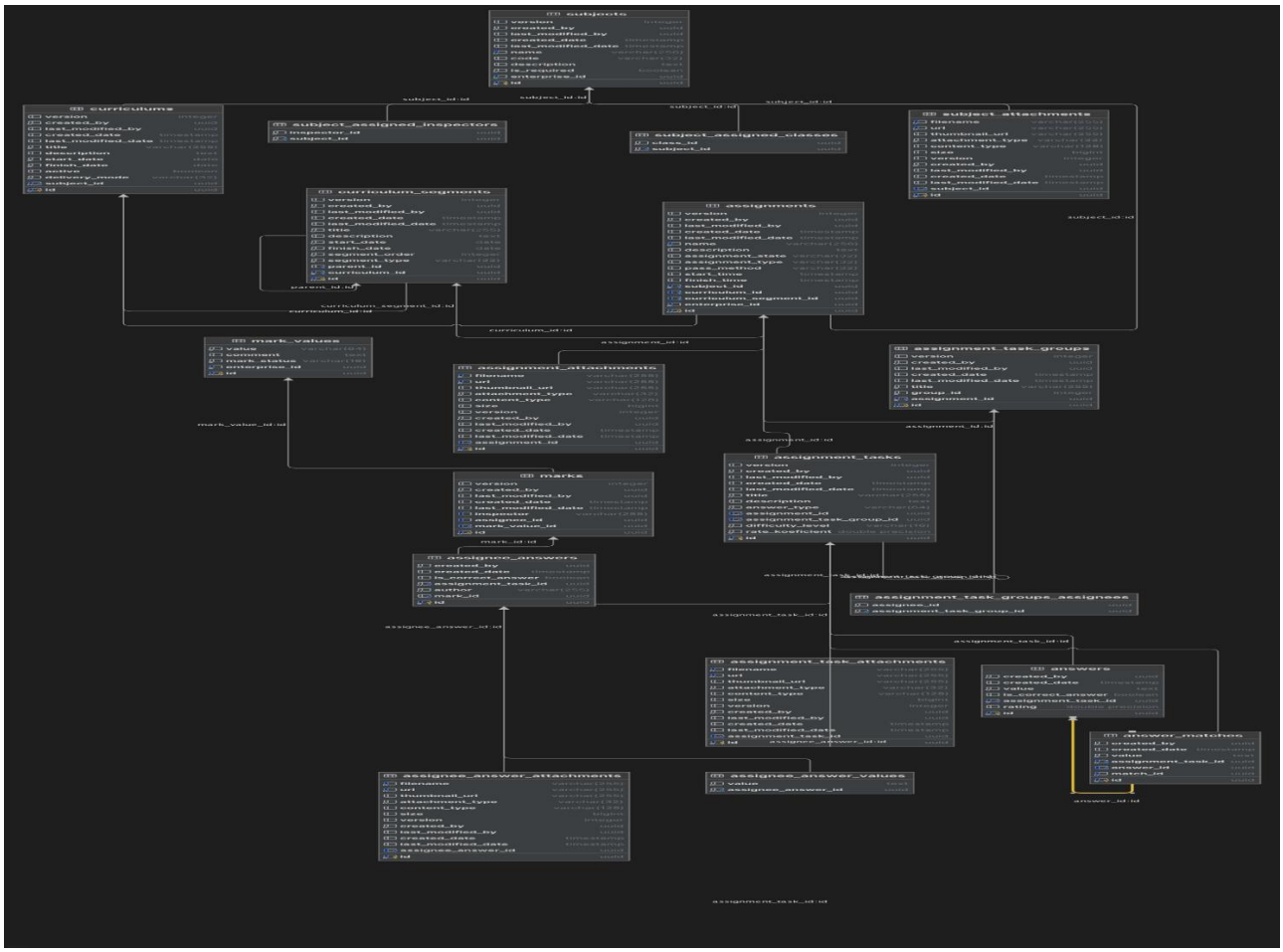


Рисунок 4.3 - ER-Діаграма бази даних завдань і предметів

subjects – ця сутність зберігає інформацію про предмети:

- version: integer: версія;
- created_by: uuid: ким створено;
- last_modified_by: uuid ким змінено;
- created_date: timestamp: дата створення;
- last_modified_date: timestamp: дата зміни;
- id: uuid (PK): індетифікатор таблиці;
- enterprise_id: айді зайкладу;
- name: varchar: назва предмету;
- description: text: опис предмету.

assignments – ця сутність зберігає інформацію про завдання:

- version: integer: версія;
- created_by: uuid: ким створено;
- last_modified_by: uuid ким змінено;
- created_date: timestamp: дата створення;
- last_modified_date: timestamp: дата зміни;
- id: uuid (PK): індетифікатор таблиці;
- enterprise_id: uuid: айді зайкладу;
- subject_id: uuid: айді предмету;
- start_time: timestamp: початок;
- finish_time: timestamp: кінець;
- assignment_state: varchar: стан завдання(розпочато/закінчено);
- curriculum_segment_id: uuid: айді частини навчальної програми.

assignment_tasks – ця сутність зберігає інформацію про задачу із завдання:

- version: integer: версія;
- created_by: uuid: ким створено;
- last_modified_by: uuid ким змінено;
- created_date: timestamp: дата створення;

- last_modified_date: timestamp: дата зміни;
- id: uuid (PK): індетифікатор таблиці;
- title: varchar: назва задачі;
- description: text: опис завдання;
- answer_type: varchar: тип відповіді;
- rate_koefecient: float: вага задачі;
- assignment_id: uuid: айді завдання;
- assignment_task_group_id: uuid: айді групи/варіанту.

answers – ця сутність зберігає інформацію про запропоновані відповіді :

- id: uuid (PK): індетифікатор таблиці;
- created_by: uuid: ким створено;
- created_date: timestamp: дата створення;
- value: значення;
- is_correct_answer: чи правильна відповідь;
- rating: float: вага відповіді;
- assignment_task_id: uuid: айді задачі.

assignee_answers – ця сутність зберігає інформацію про відповіді надіслані відповіді виконавцями:

- id: uuid (PK): індетифікатор таблиці;
- created_by: uuid: ким створено;
- created_date: timestamp: дата сторення;
- author: varchar: імя прізвище автора;
- is_correct_answer: boolean чи правильна відповідь;
- value: значення відповіді;
- answer_id: uuid: айді оцінки;
- assignment_task_id: uuid: айді задачі.

mark_values – ця сутність зберігає оцінки які зареєстрованні в закладі:

- id: uuid (PK): індетифікатор таблиці;
- enterprise_id: uuid: айді зайкладу;

- value: string: значення;
- comment: text: коментар/опис.

mark – ця сутність зберігає оцінки поставлені вчителями/викладачами:

- version: integer: версія;
- created_by: uuid: ким створено;
- last_modified_by: uuid ким змінено;
- created_date: timestamp: дата створення;
- last_modified_date: timestamp: дата зміни;
- id: uuid (PK): індетифікатор таблиці;
- inspector: string: перевіряючий (імя і прізвище або емейл);
- assignee_id: uuid: айді виконавця завдання;
- mark_value_id: uuid: айді оцінки зареєстрованої в системі.

subject_attachments - ця сутність зберігає інформацію про документи до предмету(программу, підручники в електронному форматі). Атрибути цієї сутності включають:

- filename: varchar(255): назва файлу;
- url: varchar(255): посилання;
- thumbnail_url: varchar(255): посилання на стиснутий варіант;
- attachment_type: varchar(32): тип;
- content_type: varchar(128): тип контенту;
- size: bigint: розмір;
- version: integer: версія;
- created_by: uuid: ким створено;
- last_modified_by: uuid ким змінено;
- created_date: timestamp: дата створення;
- last_modified_date: timestamp: дата зміни;
- subject_id: uuid (FK → users.id): id користувача;
- id: uuid (PK): індетифікатор таблиці.

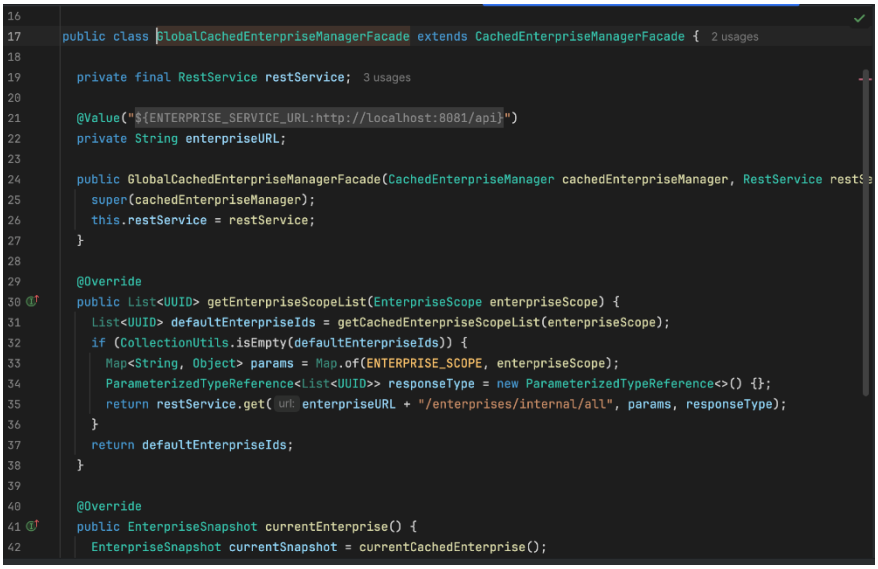
assignment_attachments - ця сутність зберігає інформацію про документи до завдання. Атрибути цієї сутності включають:

- filename: varchar(255): назва файлу;
- url: varchar(255): посилання;
- thumbnail_url: varchar(255): посилання на стиснутий варіант;
- attachment_type: varchar(32): тип;
- content_type: varchar(128): тип контенту;
- size: bigint: розмір;
- created_by: uuid: ким створено;
- last_modified_by: uuid ким змінено;
- created_date: timestamp: дата створення;
- last_modified_date: timestamp: дата зміни;
- assignemnt_id: uuid (FK → users.id): id користувача;
- id: uuid (PK): індетифікатор таблиці.

assignee_answer_attachments - ця сутність зберігає інформацію про документи прикріпленні учнем до завдання. Атрибути цієї сутності включають:

- filename: varchar(255): назва файлу;
- url: varchar(255): посилання;
- thumbnail_url: varchar(255): посилання на стиснутий варіант;
- attachment_type: varchar(32): тип;
- content_type: varchar(128): тип контенту;
- size: bigint: розмір;
- version: integer: версія;
- created_by: uuid: ким створено;
- last_modified_by: uuid ким змінено;
- created_date: timestamp: дата створення;
- last_modified_date: timestamp: дата зміни;
- assignemnt_id: uuid (FK → users.id): id користувача;
- id: uuid (PK): індетифікатор таблиці.

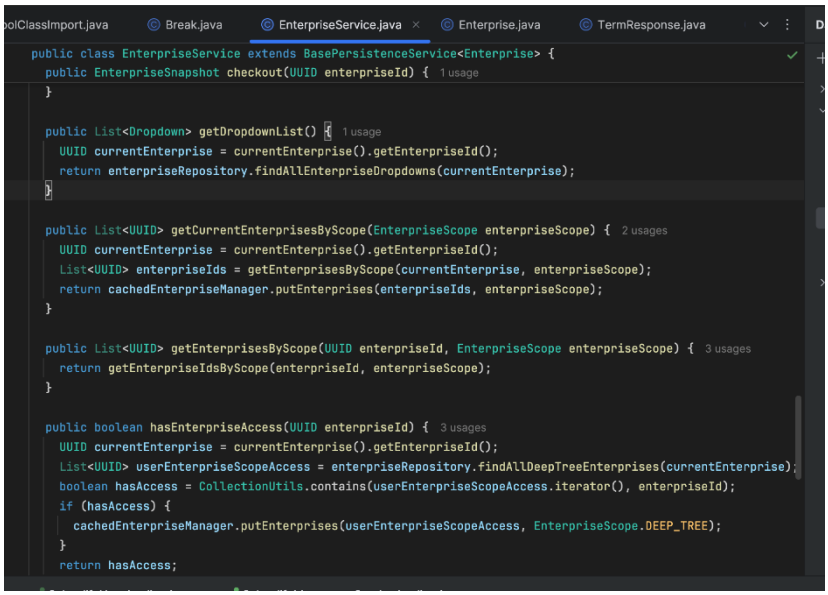
4.3 Реалізація отримання області видимості id закладів



```
public class GlobalCachedEnterpriseManagerFacade extends CachedEnterpriseManagerFacade {  2 usages

  private final RestService restService;  3 usages

  @Value("${ENTERPRISE_SERVICE_URL:http://localhost:8081/api}")
  private String enterpriseURL;

  public GlobalCachedEnterpriseManagerFacade(CachedEnterpriseManager cachedEnterpriseManager, RestService restS
    super(cachedEnterpriseManager);
    this.restService = restService;
  }

  @Override
  public List<UUID> getEnterpriseScopeList(EnterpriseScope enterpriseScope) {
    List<UUID> defaultEnterpriseIds = getCachedEnterpriseScopeList(enterpriseScope);
    if (CollectionUtils.isEmpty(defaultEnterpriseIds)) {
      Map<String, Object> params = Map.of(ENTERPRISE_SCOPE, enterpriseScope);
      ParameterizedTypeReference<List<UUID>> responseType = new ParameterizedTypeReference<>() {};
      return restService.get( url: enterpriseURL + "/enterprises/internal/all", params, responseType);
    }
    return defaultEnterpriseIds;
  }

  @Override
  public EnterpriseSnapshot currentEnterprise() {
    EnterpriseSnapshot currentSnapshot = currentCachedEnterprise();
```

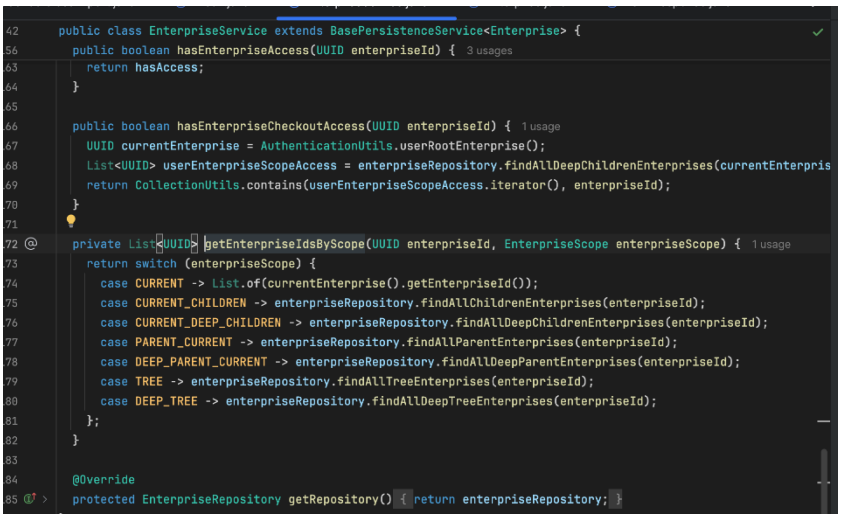
Рисунок 4.4 - Отримання області видимості через Redis Cache

Подаємо запит в Redis (рисунок 4.4) для отримання id закладів, у разі пустого списку, робимо HTTP запит на сервіс із закладами.



```
olClassImport.java   Break.java   EnterpriseService.java   Enterprise.java   TermResponse.java

public class EnterpriseService extends BasePersistenceService<Enterprise> {
  public EnterpriseSnapshot checkout(UUID enterpriseId) {  1 usage
  }

  public List<Dropdown> getDropdownList() {  1 usage
    UUID currentEnterprise = currentEnterprise().getEnterpriseId();
    return enterpriseRepository.findAllEnterpriseDropdowns(currentEnterprise);
  }

  public List<UUID> getCurrentEnterprisesByScope(EnterpriseScope enterpriseScope) {  2 usages
    UUID currentEnterprise = currentEnterprise().getEnterpriseId();
    List<UUID> enterpriseIds = getEnterprisesByScope(currentEnterprise, enterpriseScope);
    return cachedEnterpriseManager.putEnterprises(enterpriseIds, enterpriseScope);
  }

  public List<UUID> getEnterprisesByScope(UUID enterpriseId, EnterpriseScope enterpriseScope) {  3 usages
    return getEnterpriseIdsByScope(enterpriseId, enterpriseScope);
  }

  public boolean hasEnterpriseAccess(UUID enterpriseId) {  3 usages
    UUID currentEnterprise = currentEnterprise().getEnterpriseId();
    List<UUID> userEnterpriseScopeAccess = enterpriseRepository.findAllDeepTreeEnterprises(currentEnterprise);
    boolean hasAccess = CollectionUtils.contains(userEnterpriseScopeAccess.iterator(), enterpriseId);
    if (hasAccess) {
      cachedEnterpriseManager.putEnterprises(userEnterpriseScopeAccess, EnterpriseScope.DEEP_TREE);
    }
    return hasAccess;
```

Рисунок 4.5 - Отримання список Id через Сервіс закладів

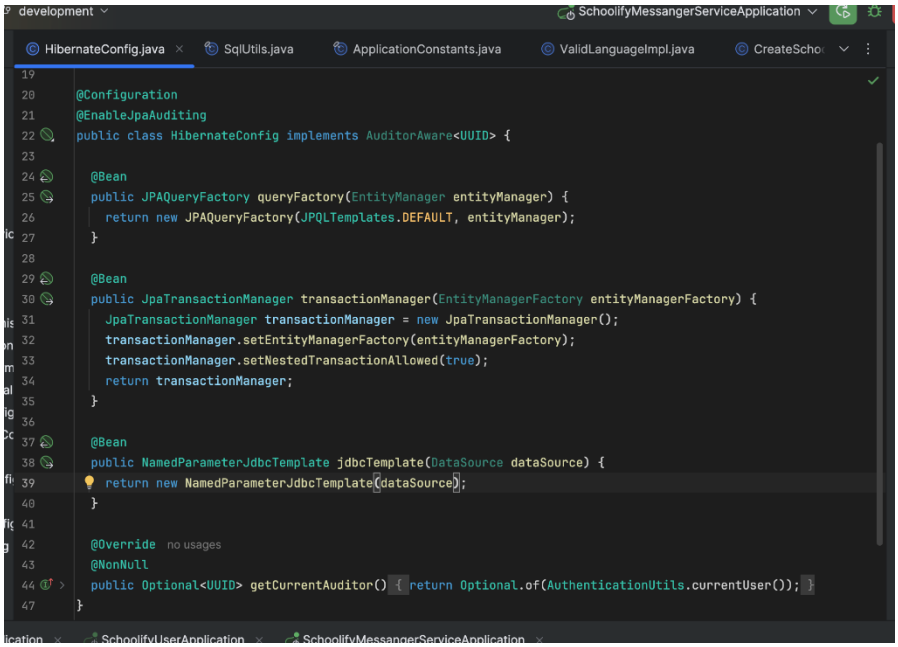


```
public class EnterpriseService extends BasePersistenceService<Enterprise> {
  public boolean hasEnterpriseAccess(UUID enterpriseId) {  3 usages
    return hasAccess;
  }

  public boolean hasEnterpriseCheckoutAccess(UUID enterpriseId) {  1 usage
    UUID currentEnterprise = AuthenticationUtils.userRootEnterprise();
    List<UUID> userEnterpriseScopeAccess = enterpriseRepository.findAllDeepChildrenEnterprises(currentEnterpris
    return CollectionUtils.contains(userEnterpriseScopeAccess.iterator(), enterpriseId);
  }

  private List<UUID> getEnterpriseIdsByScope(UUID enterpriseId, EnterpriseScope enterpriseScope) {  1 usage
    return switch (enterpriseScope) {
      case CURRENT -> List.of(currentEnterprise().getEnterpriseId());
      case CURRENT_CHILDREN -> enterpriseRepository.findAllChildrenEnterprises(enterpriseId);
      case CURRENT_DEEP_CHILDREN -> enterpriseRepository.findAllDeepChildrenEnterprises(enterpriseId);
      case PARENT_CURRENT -> enterpriseRepository.findAllParentEnterprises(enterpriseId);
      case DEEP_PARENT_CURRENT -> enterpriseRepository.findAllDeepParentEnterprises(enterpriseId);
      case TREE -> enterpriseRepository.findAllTreeEnterprises(enterpriseId);
      case DEEP_TREE -> enterpriseRepository.findAllDeepTreeEnterprises(enterpriseId);
    };
  }

  @Override
  protected EnterpriseRepository getRepository() { return enterpriseRepository; }
```

Рисунок 4.6 - Отримання списку id компаній з бази даних

На вище вказаних рисунках 4.5 і 4.6 реалізовано функції витягування списку id з бази даних і внесення їх в Redis Cache для повторного використання (додаток В).

4.4 Конфігурація основних частин серверного проєкту



```
@Configuration
@EnableJpaAuditing
public class HibernateConfig implements AuditorAware<UUID> {

  @Bean
  public JPAQueryFactory queryFactory(EntityManager entityManager) {
    return new JPAQueryFactory(JPQLTemplates.DEFAULT, entityManager);
  }

  @Bean
  public JpaTransactionManager transactionManager(EntityManagerFactory entityManagerFactory) {
    JpaTransactionManager transactionManager = new JpaTransactionManager();
    transactionManager.setEntityManagerFactory(entityManagerFactory);
    transactionManager.setNestedTransactionAllowed(true);
    return transactionManager;
  }

  @Bean
  public NamedParameterJdbcTemplate jdbcTemplate(DataSource dataSource) {
    return new NamedParameterJdbcTemplate(dataSource);
  }

  @Override  no usages
  @NonNull
  public Optional<UUID> getCurrentAuditor() { return Optional.of(AuthenticationUtils.currentUser()); }
}
```

Рисунок 4.7 - Конфігурація JPA Та JDBC Template

Тут сконфігуровано Hibernate (рисунок 4.7) та обгортку SQL Query DSL для зручного користування і JDBC темплейт для імплементації ефективного CQRS паттерну.

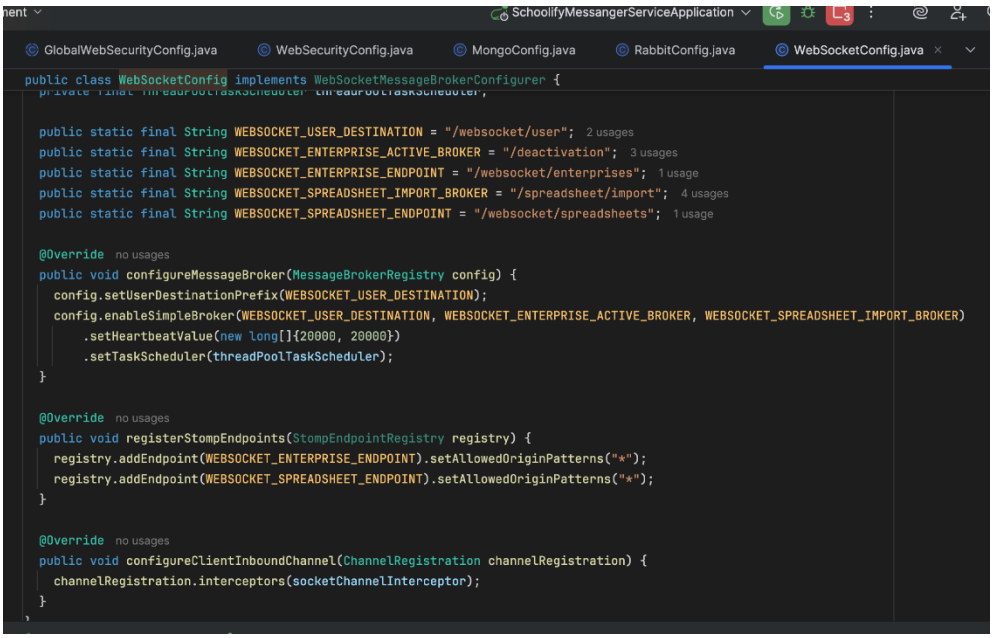


Рисунок 4.8 - Конфігурація Websocket протоколу

Вище на рисунку 4.8 зконфігуровано Websocket протокол для імпорту Excel файлів для дуплексного сполучення, що дає необмежений час і можливість асинхронно обробляти дані. Всі вхідні повідомлення будуть провірятися на наявність коректного JWT токена.

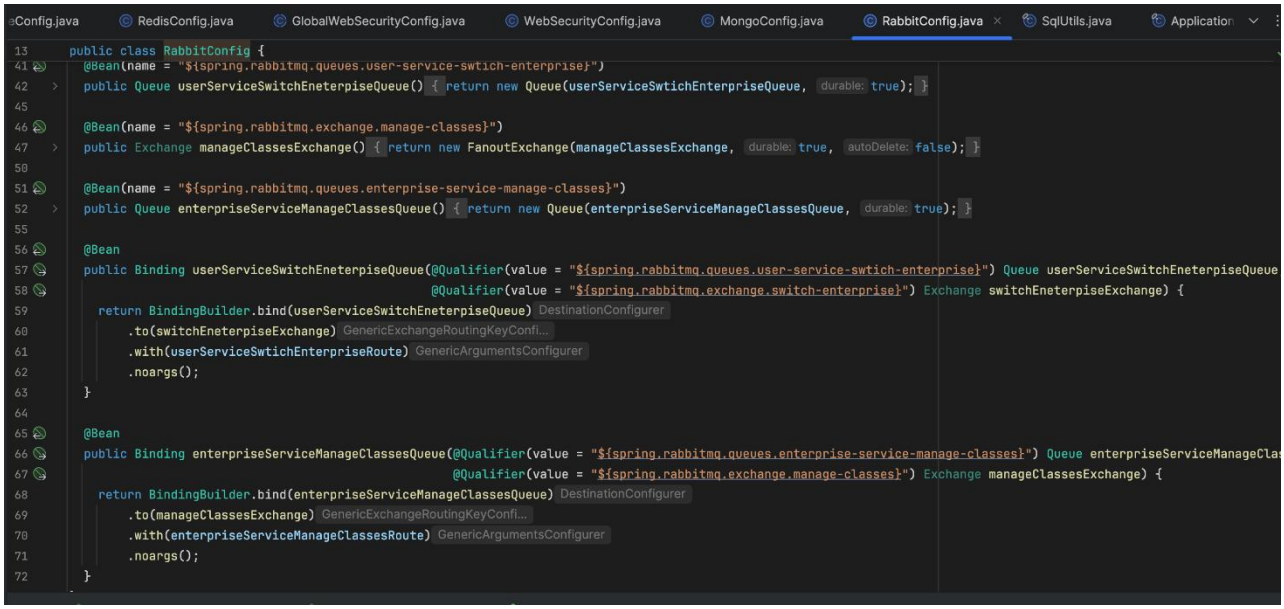


Рисунок 4.9 - Конфігурація Rabbit MQ

Конфігурація Rabbit шаблонів (рисунок 4.9) для асинхронного спілкування між сервісами проєкту у режимі Publisher/Subscriber у форматі черги через паттерн: перший зайшов - перший вийшов. Дана технологія буде необхідна для надсилання повідомлень між серверами та відправлення емейлів

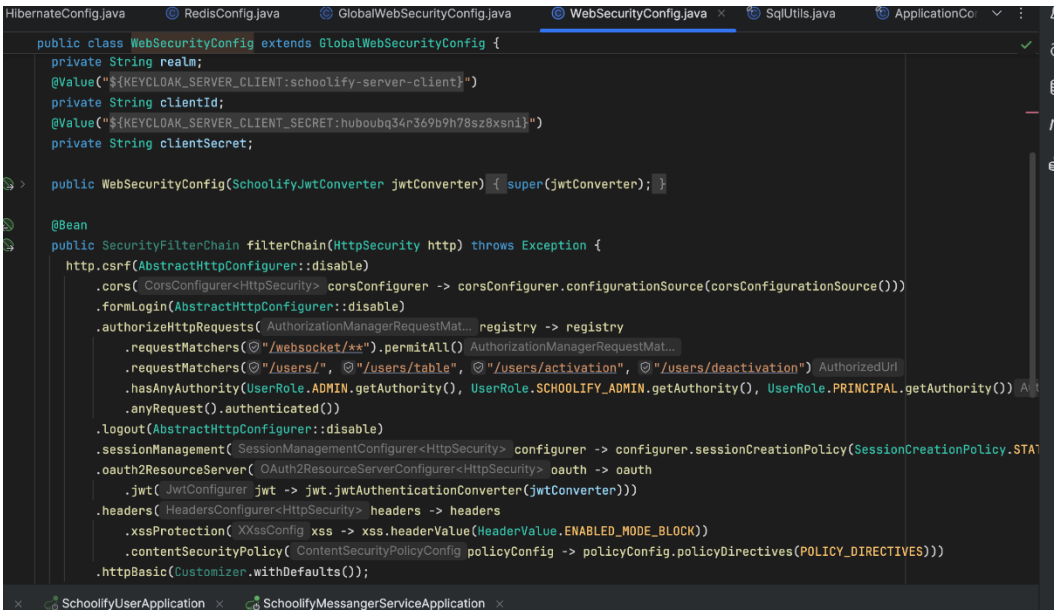


Рисунок 4.10 - Конфігурація безпеки серверу

Налаштування модуля (рисунок 4.10) Spring Security, де прописану які ендпойнти є не тільки автентифіковані, а й авторизовані. Авторизація користувачів проходить через авторизаційний сервер - OIDC Keycloak 2.0.

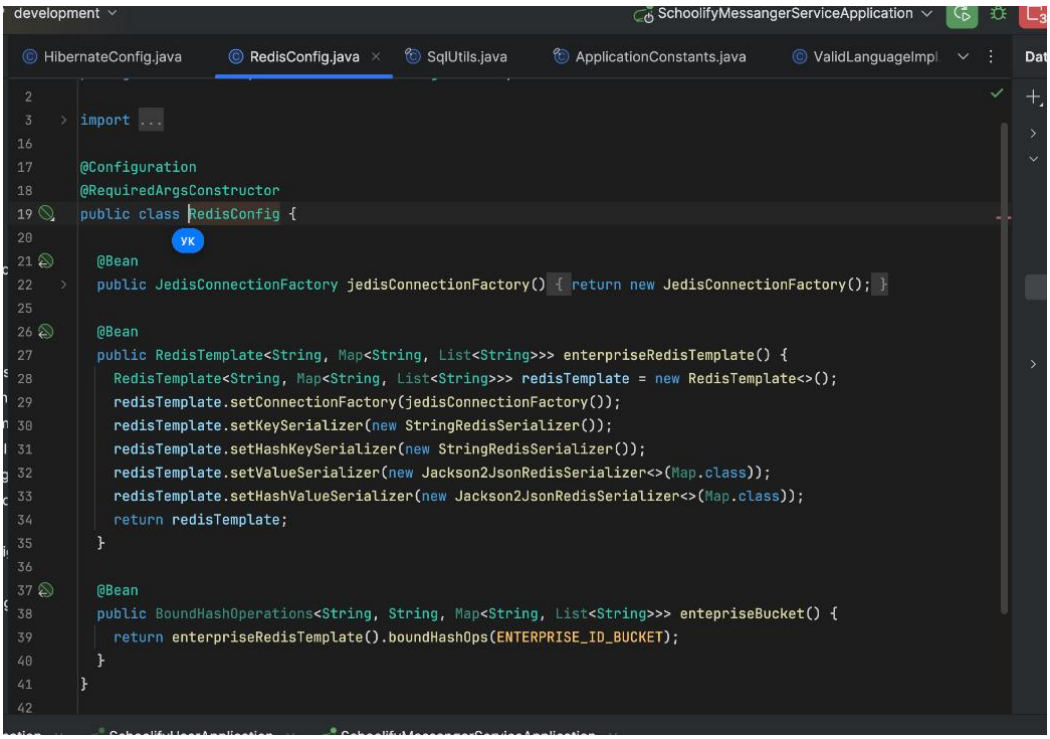
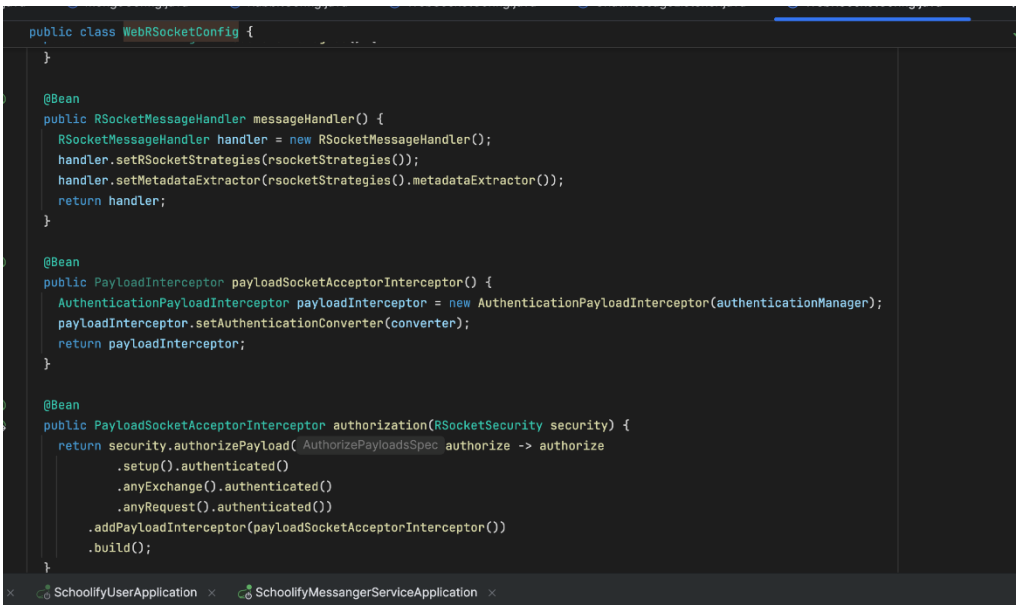


Рисунок 4.11 - Конфігурація Redis Cache

Конфігурація бази даних Redis Cache (рисунок 4.11) для кешування списків id закладів для кожного користувачана у форматі key-value, що дає більший приріст у швидкості навідміну від стандартної SQL бази даних при повторних запитах.

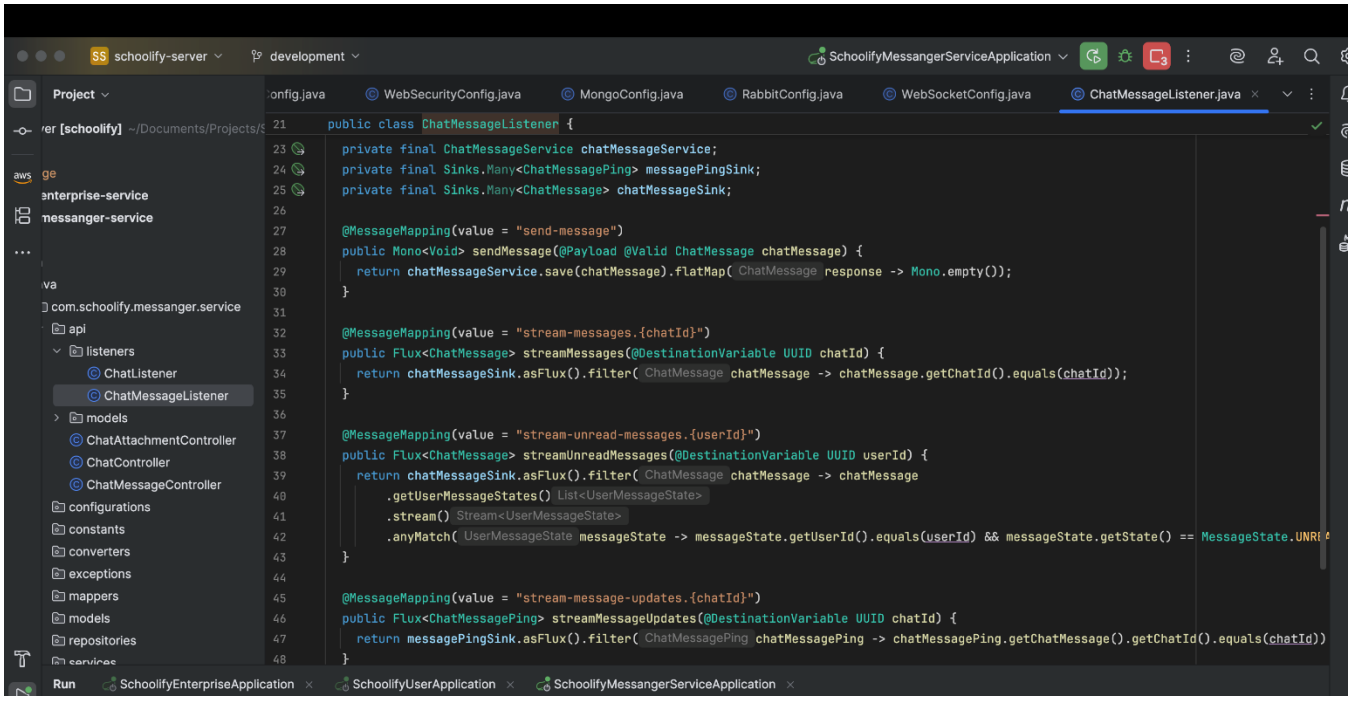


```
public class WebRSocketConfig {
}

@Bean
public RSocketMessageHandler messageHandler() {
  RSocketMessageHandler handler = new RSocketMessageHandler();
  handler.setRSocketStrategies(rsocketStrategies());
  handler.setMetadataExtractor(rsocketStrategies().metadataExtractor());
  return handler;
}

@Bean
public PayloadInterceptor payloadSocketAcceptorInterceptor() {
  AuthenticationPayloadInterceptor payloadInterceptor = new AuthenticationPayloadInterceptor(authenticationManager);
  payloadInterceptor.setAuthenticationConverter(converter);
  return payloadInterceptor;
}

@Bean
public PayloadSocketAcceptorInterceptor authorization(RSocketSecurity security) {
  return security.authorizePayload( AuthorizePayloadsSpec authorize -> authorize
          .setup().authenticated()
          .anyExchange().authenticated()
          .anyRequest().authenticated())
      .addPayloadInterceptor(payloadSocketAcceptorInterceptor())
      .build();
}
```

SchoolifyUserApplication × SchoolifyMessangerServiceApplication ×

Рисунок 4.12 - Конфігурація Spring Webflux RSocket

Конфігурація Websocket (рисунок 4.12) на реактивній парадигмі програмування, яка дає надійність при високому навантаженні запитами користувачів, задіяний для месенджера програми, всі повідомлення зберігаються у MongoDB. Данна технологія буде використовуватись для спілкування між користувачами у вбудованому мессенджері. Всі вхідні повідомлення будуть провірятися на наявність коректного JWT токена.



schoolify-server · development · SchoolifyMessangerServiceApplication

Project

config.java · WebSecurityConfig.java · MongoConfig.java · RabbitConfig.java · WebSocketConfig.java · ChatMessageListener.java

```
21  public class ChatMessageListener {
23      private final ChatMessageService chatMessageService;
24      private final Sinks.Many<ChatMessagePing> messagePingSink;
25      private final Sinks.Many<ChatMessage> chatMessageSink;
26
27      @MessageMapping(value = "send-message")
28      public Mono<Void> sendMessage(@Payload @Valid ChatMessage chatMessage) {
29          return chatMessageService.save(chatMessage).flatMap( ChatMessage response -> Mono.empty());
30      }
31
32      @MessageMapping(value = "stream-messages.{chatId}")
33      public Flux<ChatMessage> streamMessages(@DestinationVariable UUID chatId) {
34          return chatMessageSink.asFlux().filter( ChatMessage chatMessage -> chatMessage.getChatId().equals(chatId));
35      }
36
37      @MessageMapping(value = "stream-unread-messages.{userId}")
38      public Flux<ChatMessage> streamUnreadMessages(@DestinationVariable UUID userId) {
39          return chatMessageSink.asFlux().filter( ChatMessage chatMessage -> chatMessage
40                  .getUserMessageStates() List<UserMessageState>
41                  .stream() Stream<UserMessageState>
42                  .anyMatch( UserMessageState messageState -> messageState.getUserId().equals(userId) && messageState.getState() == MessageState.UNRE
43      }
44
45      @MessageMapping(value = "stream-message-updates.{chatId}")
46      public Flux<ChatMessagePing> streamMessageUpdates(@DestinationVariable UUID chatId) {
47          return messagePingSink.asFlux().filter( ChatMessagePing chatMessagePing -> chatMessagePing.getChatMessage().getChatId().equals(chatId))
48      }
```

com.schoolify.messanger.service
api
listeners
ChatListener
ChatMessageListener
models
ChatAttachmentController
ChatController
ChatMessageController
configurations
constants
converters
exceptions
mappers
models
repositories

Run · SchoolifyEnterpriseApplication · SchoolifyUserApplication · SchoolifyMessangerServiceApplication

Рисунок 4.13 - API для листування між користувачами

Вище на рисунку 4.13 представлено декілька каналів для листування між користувачами, транслювання сповіщень, та оновлень у чатах.

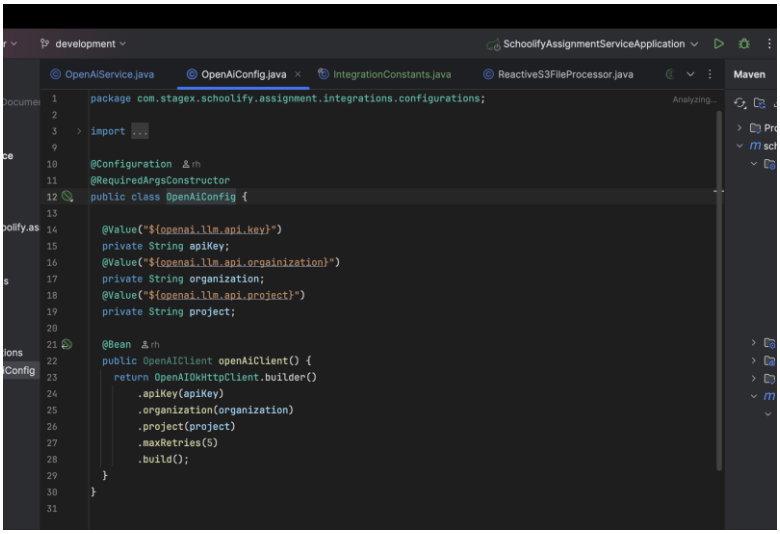


Рисунок 4.14 - Конфігурація клієнту для взаємодії серверу з інтерфейсом Open AI

Ця конфігурація (рисунок 4.14) призначений для взаємодії з інтерфейсом OPEN AI API – використовутиметься для автоматичної перевірки завдань з типом – розгорнутої відповіді або якшо відповідь містить документ чи фотографію. Автоматизація цього процессу позбавить необхідності вручну перевіряти завдання учнів чи студентів, але залишиться можливість перевірити вручну в разі потреби, оскільки ШІ може припускатися помилок. Для конфігурації клієнта необхідно придбати API ключ для доступу до ресурсів та створити аккаунт на OPEN API і визначити організацію та проєкт. Після цього вставляємо API ключ і айді організації та проєкту в конфігурацію.

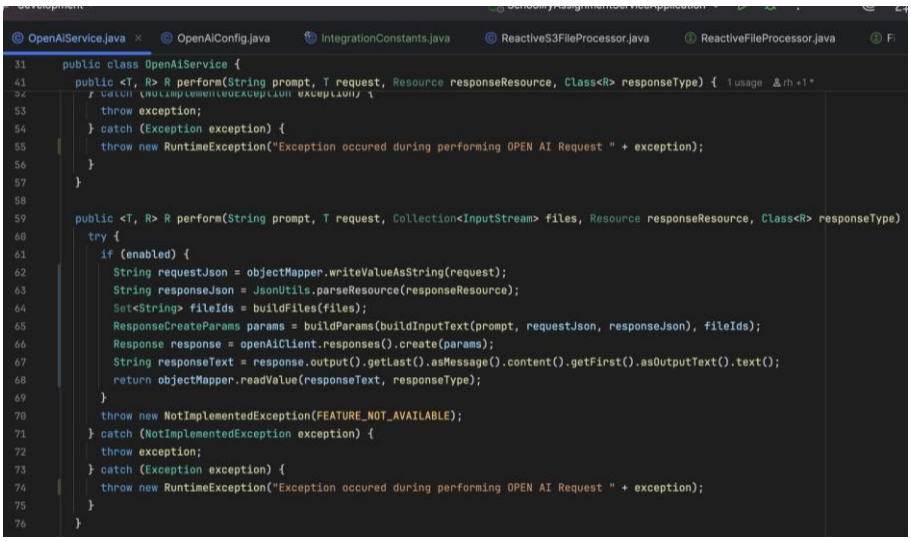


Рисунок 4.15 - Імплементація сервісу, який виконує запити на Open AI

Даний сервсіс (рисунок 4.15) приймає на вхід: промпт, тип даних які потрібно надіслати, формат відповіді який необхідно отримати від чату GPT (в даному випадку це буде JSON формат). І тип відповіді в яку потрібно буде десеріалізувати в зрозумілий для сервера тип обєкту. Дану функцію також можна відключити.

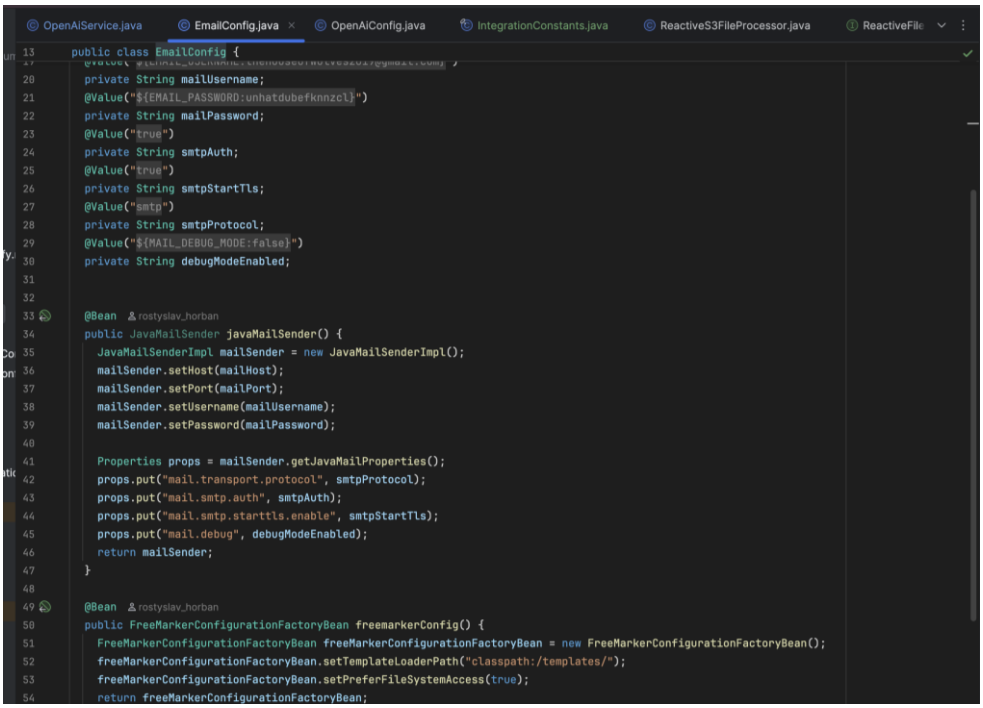


Рисунок 4.16 - Конфігурація розсилки емейл-сповіщень

Дана конфігурація (рисунок 4.16) буде використовуватись для емейл сповіщень таких як: сповіщення про реєстрацію, добавлення або видалення з класу, нагадування про початок завданнь. В якості шаблонів було використання free-marker

шаблони, з яких повідомлення в трансформується в HTML вставку для відображення відформатованих емейлів (рисунок 4.17).

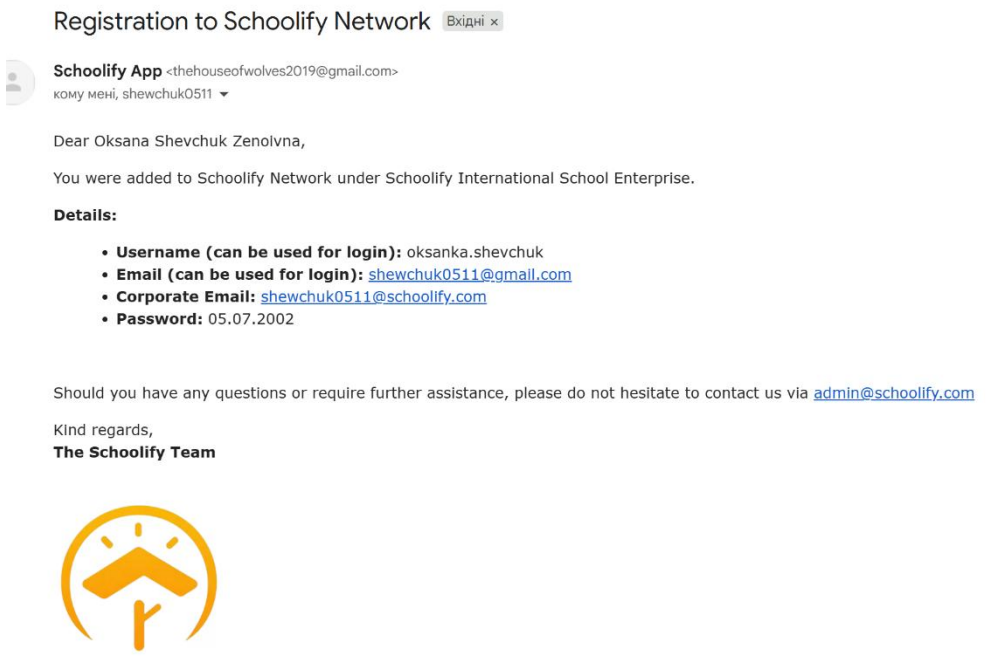


Рисунок 4.17 – Приклад емейл сповіщення про реєстрацію

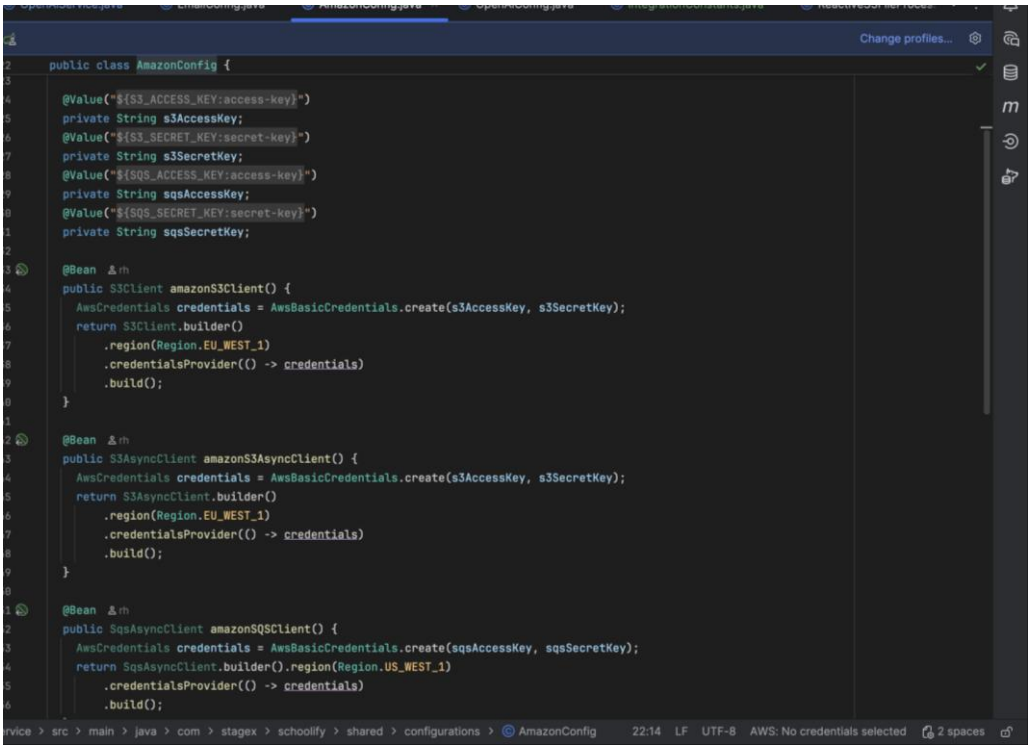


Рисунок 4.18 – Конфігурація AWS S3 клієнтів

AWS S3 Java Client — це бібліотека, яка дозволяє Java-застосункам програмно взаємодіяти з сервісом Amazon S3 для зберігання та обробки об’єктів у хмарі. За її допомогою можна створювати та видаляти бакети, завантажувати, завантажувати та

видаляти файли (об’єкти), управляти версіями та метаданими, налаштовувати права доступу та шифрування. Клієнт автоматично обробляє аутентифікацію через AWS Credentials, повтори при помилках, підключення до регіонів і інтеграцію з іншими сервісами AWS, що спрощує побудову надійних та масштабованих систем зберігання даних без необхідності працювати з HTTP-запитами вручну (рисунки 4.18 і 4.19).

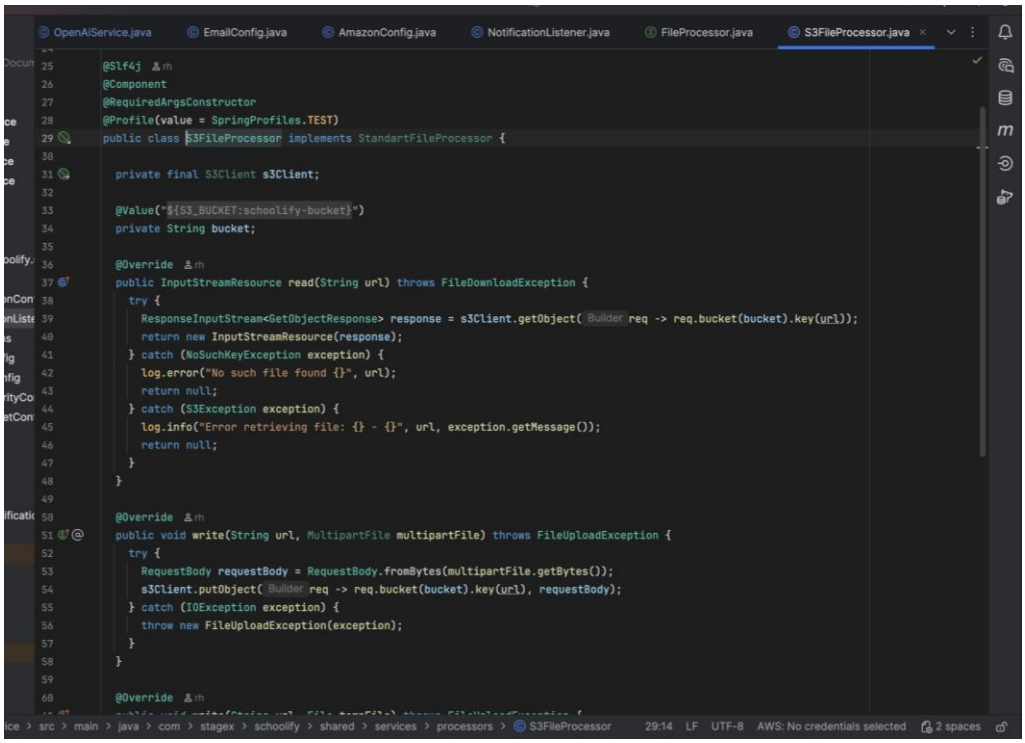


Рисунок 4.19 – Імплементація загрузки і вигрузки файлів на AWS S3

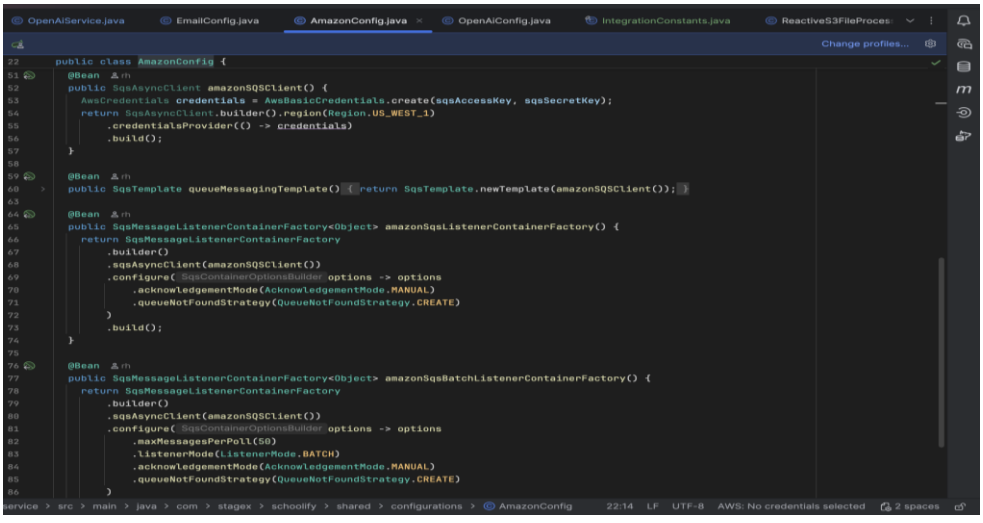


Рисунок 4.20 – Конфігурація AWS SQS клієнта

AWS SQS Java Client — це бібліотека, яка дозволяє Java-застосункам програмно взаємодіяти з сервісом Amazon SQS для асинхронного обміну

повідомленнями між компонентами системи. За її допомогою можна створювати черги, надсилати та отримувати повідомлення, видаляти їх після обробки, налаштовувати таймаути видимості та управляти атрибутами черг і повідомлень. Клієнт автоматично обробляє аутентифікацію через AWS Credentials, повтори при помилках та інтеграцію з іншими сервісами AWS, що дозволяє будувати надійні, масштабовані та розподілені системи без необхідності вручну працювати з HTTP-запитами до SQS (рисунки 4.20 і 4.21).

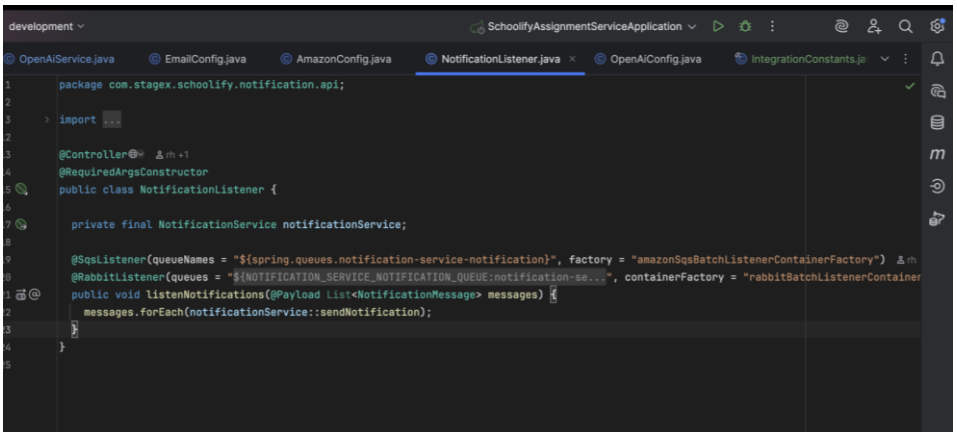


Рисунок 4.21 – Приклад приймання сповіщенн за допомогою RabbitMQ і SQS

4.5 Реалізація клієнтської частини проєкту

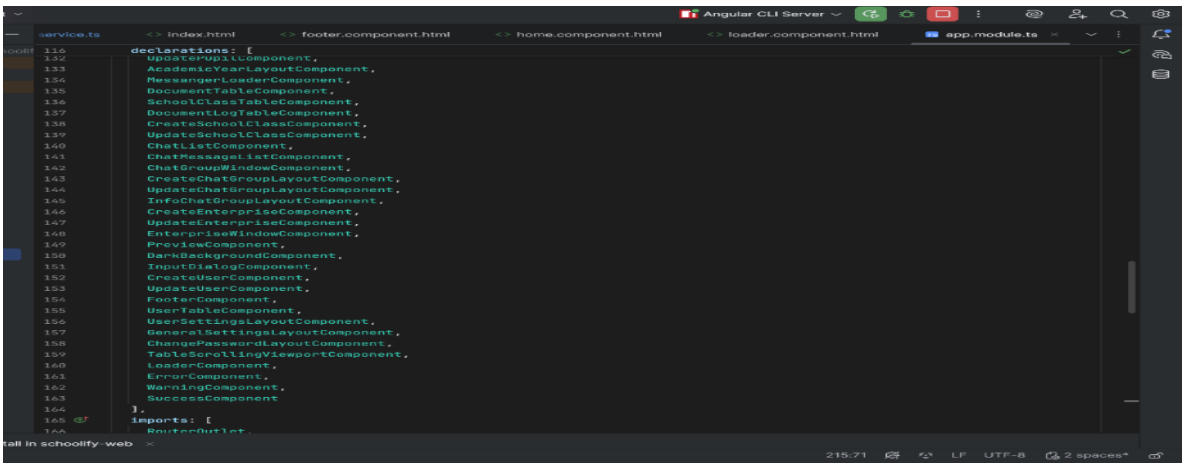


Рисунок 4.22 - Структура проєкту клієнтської частини

На рисунку 4.22 наведено структуру проєкту де:

- components відповідає за інтерфейс з html шаблонами;
- constants: константи програми;
- directives дає абстрактний функціонал для схожих за типом компонентів;

- interceptors: відопвідає за скрипти перехоплень при певних процесах;
    - Models: надає моделі об’єктів які приходять з сервера;
    - Pipes: функціонал для перетворень значень;
    - Security: забезпечує безпеку клієнтської частини;
    - Services: сервіси для обробки даних які надає API;
    - Utils: Допоміжні файли для конвертації даних.

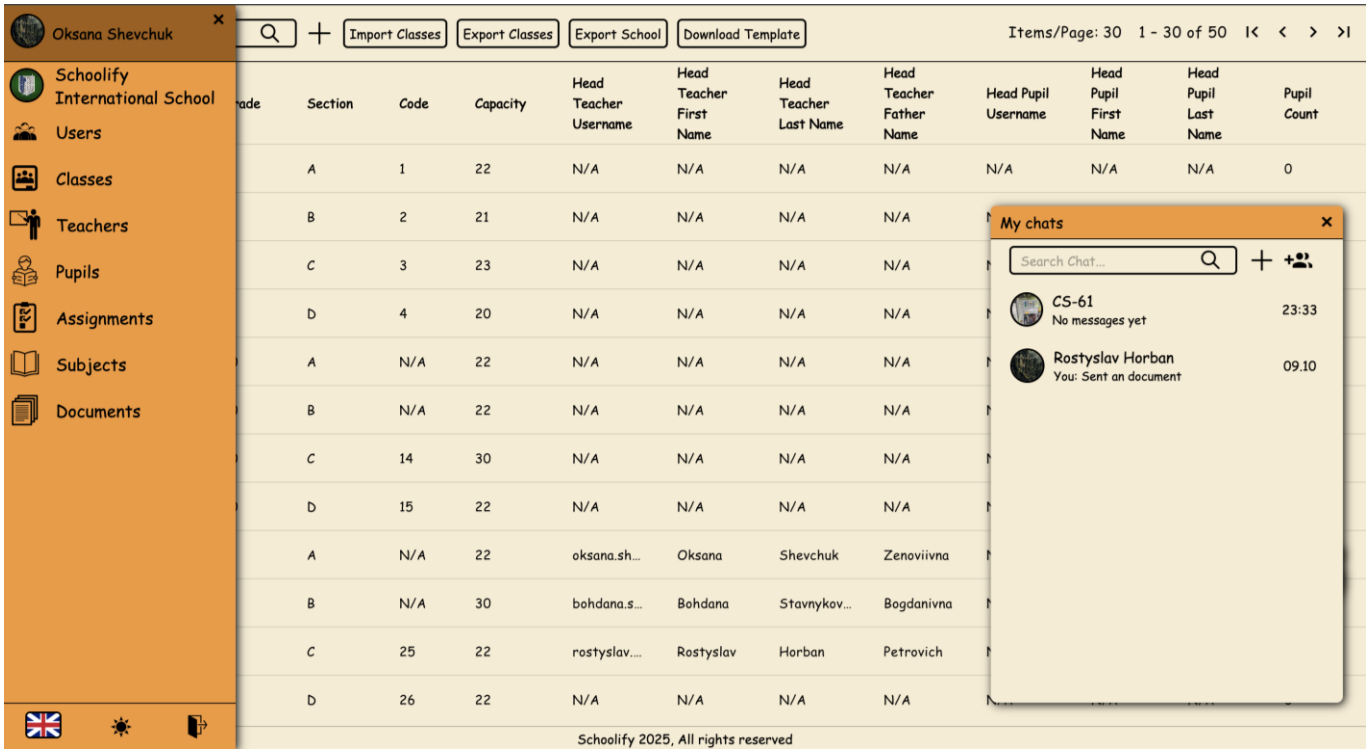


Рисунок 4.23 - Графічний інтерфейс класів у школі

Вище на рисунку 4.23 показано таблицю, де відображено шкільні класи, хто в класі є класним керівником, старостою та загальну кількість учнів. Також зображено месенджер, де можна листуватись як в режимі приватних повідомлень, так і в режимі груп (додаток А).

Рисунок 4.24 - Таблиця навчальних закладів

Вище на рисунку 4.24 зображено таблицю навчальних закладів, також зображено кнопку Users де можна відкрити таблицю користувачі та Documents, де можна завантажити всі необхідні файли які стосуються навчального закладу. Всі таблиці програми реалізовано через CQRS паттерн та пагінації за OFFSET-LIMIT стандартним патерном (додаток Б).

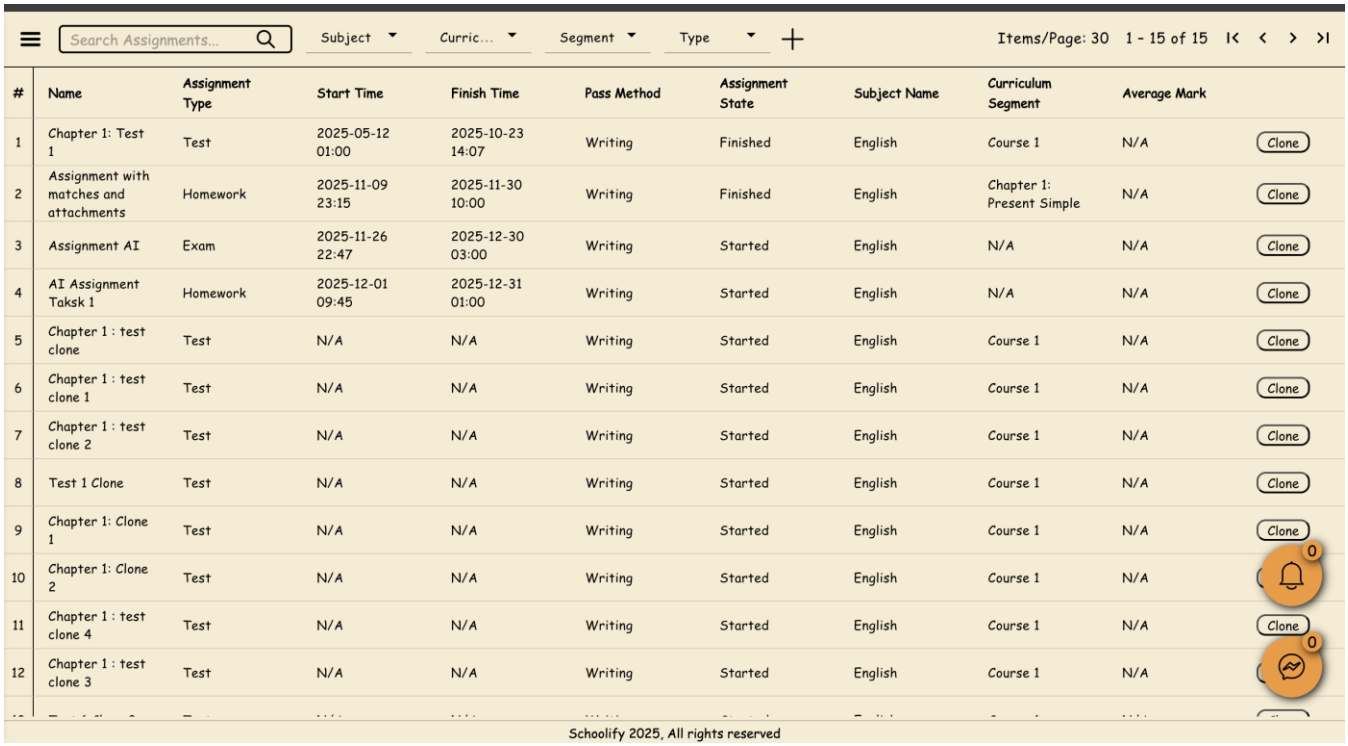
Рисунок 4.25- Таблиця предметів

На рисунку 4.25 показано вебінтерфейс з таблицею предметів у системі, для управління навчальними предметами. Вгорі є панель пошуку, поле для пошуку предметів, а також імпортування/експортування. Також присутня кнопка для додавання нового предмета. Таблиця містить колонки:

- Name — назва завдання;

- Code — код або скорочена назва завдання;
- Required — чи предмет є обов’язковим;
- Assigned Class Count — кількість класів записані на цей предмет;
- Assigned Inspector Count – кількість перевіряючих задіяні на цей педмет.

Це панель адміністратора або викладача для перегляду, фільтрування та керування завданнями в навчальному середовищі.



#	Name	Assignment Type	Start Time	Finish Time	Pass Method	Assignment State	Subject Name	Curriculum Segment	Average Mark	
1	Chapter 1: Test 1	Test	2025-05-12 01:00	2025-10-23 14:07	Writing	Finished	English	Course 1	N/A	Clone
2	Assignment with matches and attachments	Homework	2025-11-09 23:15	2025-11-30 10:00	Writing	Finished	English	Chapter 1: Present Simple	N/A	Clone
3	Assignment AI	Exam	2025-11-26 22:47	2025-12-30 03:00	Writing	Started	English	N/A	N/A	Clone
4	AI Assignment Taksk 1	Homework	2025-12-01 09:45	2025-12-31 01:00	Writing	Started	English	N/A	N/A	Clone
5	Chapter 1 : test clone	Test	N/A	N/A	Writing	Started	English	Course 1	N/A	Clone
6	Chapter 1 : test clone 1	Test	N/A	N/A	Writing	Started	English	Course 1	N/A	Clone
7	Chapter 1 : test clone 2	Test	N/A	N/A	Writing	Started	English	Course 1	N/A	Clone
8	Test 1 Clone	Test	N/A	N/A	Writing	Started	English	Course 1	N/A	Clone
9	Chapter 1: Clone 1	Test	N/A	N/A	Writing	Started	English	Course 1	N/A	Clone
10	Chapter 1: Clone 2	Test	N/A	N/A	Writing	Started	English	Course 1	N/A	Clone
11	Chapter 1 : test clone 4	Test	N/A	N/A	Writing	Started	English	Course 1	N/A	Clone
12	Chapter 1 : test clone 3	Test	N/A	N/A	Writing	Started	English	Course 1	N/A	Clone

Рисунок 4.26 - Таблиця завдань

На рисунку 4.26 показано вебінтерфейс з таблицею завдань у системі, ймовірно для управління навчальними матеріалами або курсами. Є панель пошуку та фільтри, поле для пошуку завдань, а також випадаючі списки для вибору предмета, навчальної програми, сегменту та типу завдання. Також присутня кнопка для додавання нового завдання. Таблиця містить такі колонки: **Name** — назва завдання; **Assignment Type** — тип завдання (Test, Homework, Exam); **Start Time** — дата і час початку завдання; **Finish Time** — дата і час завершення завдання; **Pass Method** — спосіб проходження завдання; **Assignment State** — стан завдання; **Subject Name** — предмет; **Curriculum**

Segment — сегмент навчальної програми; **Average Mark** — середній бал. Остання колонка містить кнопку клонування для кожного завдання.

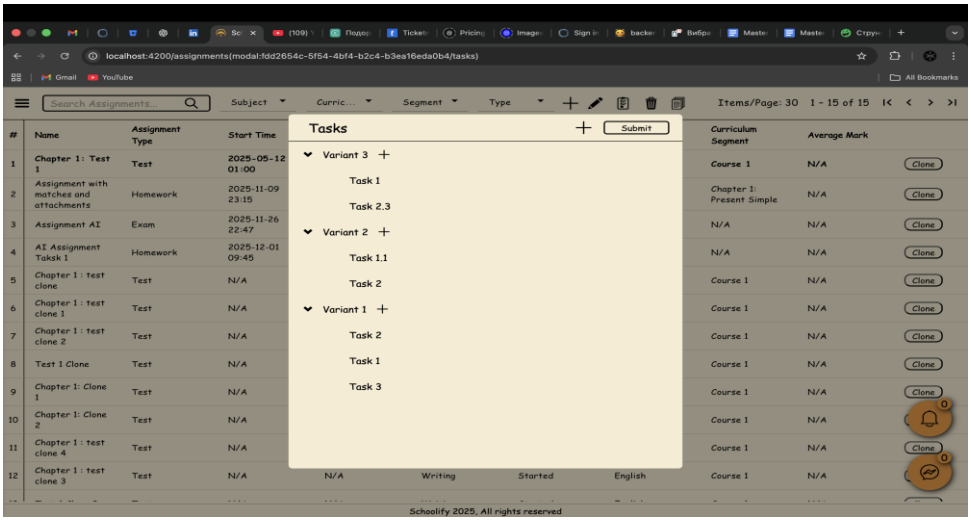


Рисунок 4.27 – Вікно з варіантами і задачами

На рисунку 4.27 зображено модальне вікно для керування варіантів і задачами. Основні елементи інтерфейсу: у центрі екрана розташоване модальне вікно *“Tasks”*, яке відображає деталі завдань для вибраного запису та містить заголовок *Tasks* і кнопку *Submit*. Перелік завдань згруповано за варіантами, під кожним з яких наведено окремі завдання, а поруч із кожним варіантом знаходиться кнопка + для додавання нового завдання. Крім того, передбачена можливість редагування вже створених завдань.

Рисунок 4.28 – Приклад вікна з задачами зі сторони вчителя

На даному рисунку 4.28 зображено завдання і відповідь учня з обов’язковими вкладеннями. Вчитель чи викладач може завантажити вкладення у форматі ZIP та поставити відповідну оцінку за відповідь. На модальному вікні також зображено назву задачі, опис та навігацію по задачах.

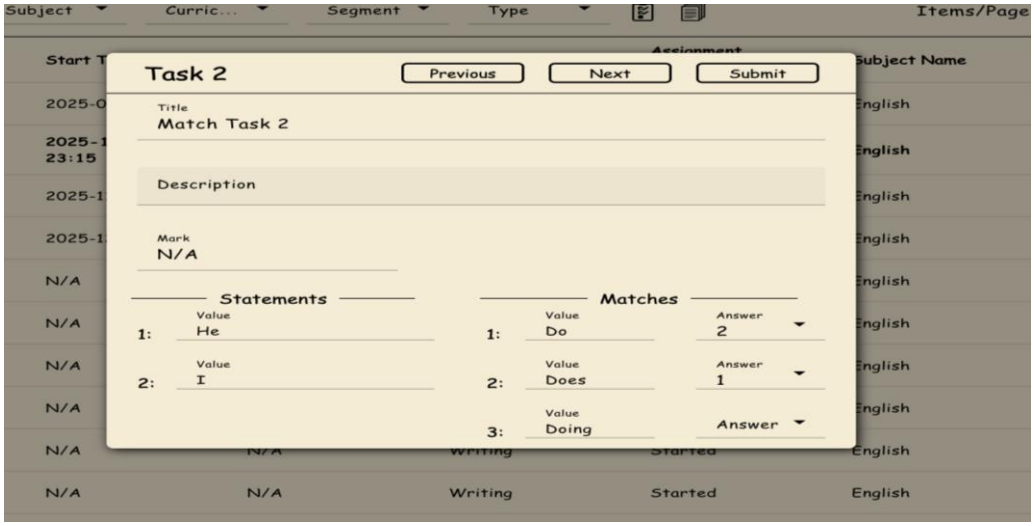


Рисунок 4.29 – Приклад вікна із задачами зі сторони виконавця

На даному риснку 4.29 зображено завдання і відповідь учня у форматі з’єднати відповідь. У першій колонці находиться перша частина визначення. У другій колонці необхідно вибрати правильне до визначення відповіть Вчитель чи викладач може завантажити вкладення у форматі ZIP та поставити відповідну оцінку за відповідь. На модальному вікні також зображено назву задачі, опис та навігацію по задачах.

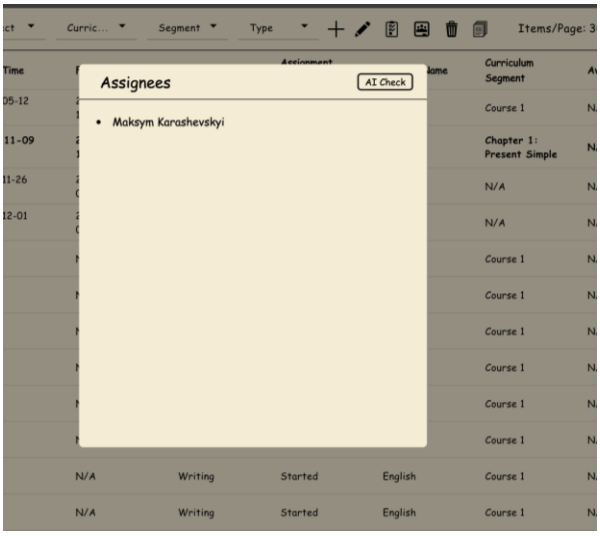


Рисунок 4.30 – Приклад вікна для списку тих хто виконав завдання

У даному модальному вікні (рисунок 4.30) вчитель або викладач отримує доступ до відповідей учнів чи студентів списком. Модальне вікно має можливість відкрити відповіді учня і можливсті перевірити відповіді за допомогою штучного інтелекту.

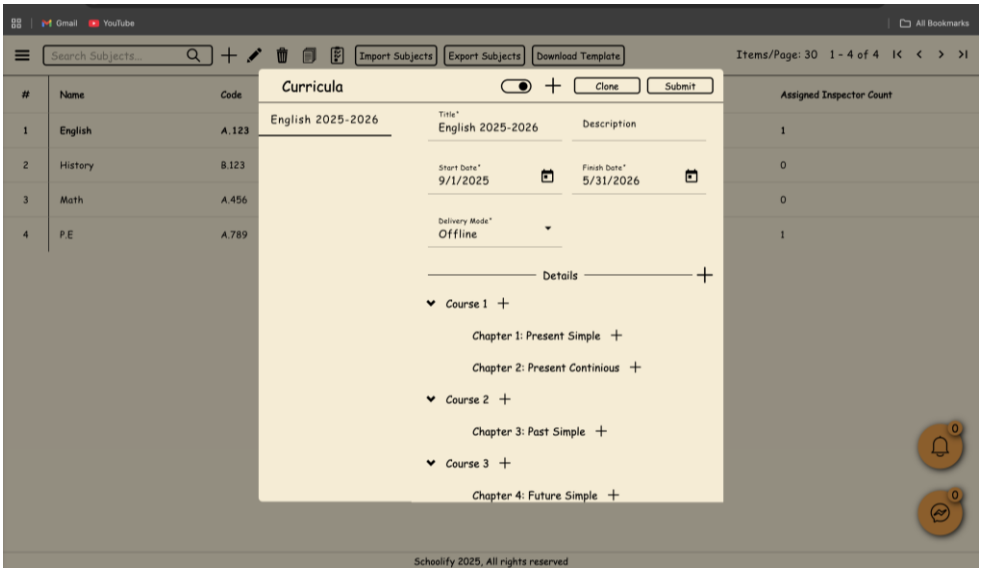


Рисунок 4.31 – Приклад вікна з структурою навчальної програми

На даному рисунку 4.31 зображено вебінтерфейс модального вікна **“Curricula”**, яке показує структуру навчальної програми для предмету **English 2025-2026**.

Загальна структура інтерфейсу

Ліворуч на сторінці знаходиться таблиця предметів з колонками: #, Name, Code та Assigned Inspector Count. Вона відображає чотири предмети — English, History, Math та P.E. — з відповідними кодами та кількістю призначених інспекторів. В центрі екрану відкрито модальне вікно **“Curricula”**, яке дозволяє переглядати та редагувати навчальну програму для вибраного предмету.

У верхній частині модального вікна розташовані поля для введення назви програми (Curricula / Title / Description), дати початку та завершення курсу (Start Date: 9/1/2025, Finish Date: 5/31/2026), а також вибір способу навчання (Delivery Mode: Offline). Ці елементи дозволяють швидко редагувати базові параметри навчальної програми.

Розділ **Details** демонструє структуру програми за курсами та главами. Наприклад, у Course 1 є Chapter 1: Present Simple, Chapter 2: Present Continuous,

Chapter 3: Past Simple та Chapter 4: Future Simple. Поруч з кожною главою та курсом є кнопка +, яка дозволяє додавати нові елементи та гнучко розширювати навчальну програму.

Вгорі вікна також присутній перемикач стану та кнопки +, **Clone** і **Submit**, які відповідають за додавання нових програм, копіювання існуючих та збереження змін. У нижньому правому куті сторінки видно дві круглі кнопки з іконками сповіщень. Весь інтерфейс демонструє детальний ієрархічний спосіб управління навчальними програмами безпосередньо через вебзастосунок.

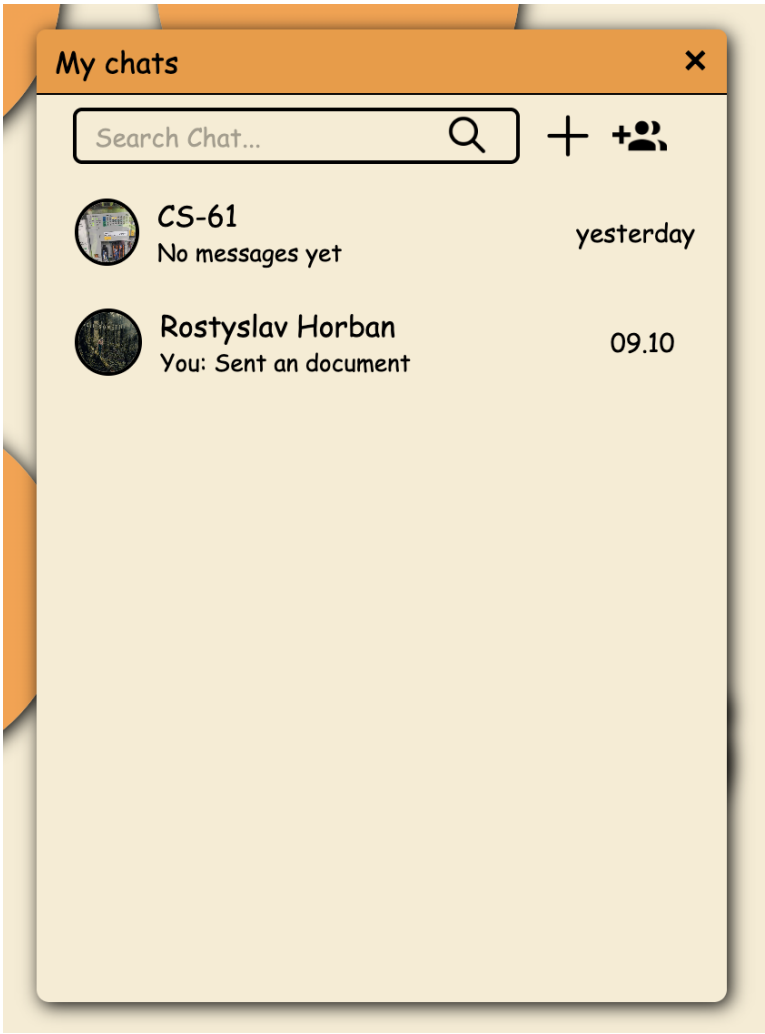


Рисунок 4.32 – Зображення списку чатів з користувачами

На рисунку 4.32 показано вікно застосунку зі списком чатів. У верхній частині є заголовок поле пошуку та кнопки для створення нового групового чату й додавання діалогу. Також можна подивитись останнє повідомлення та дату відправки останнього повідомлення.

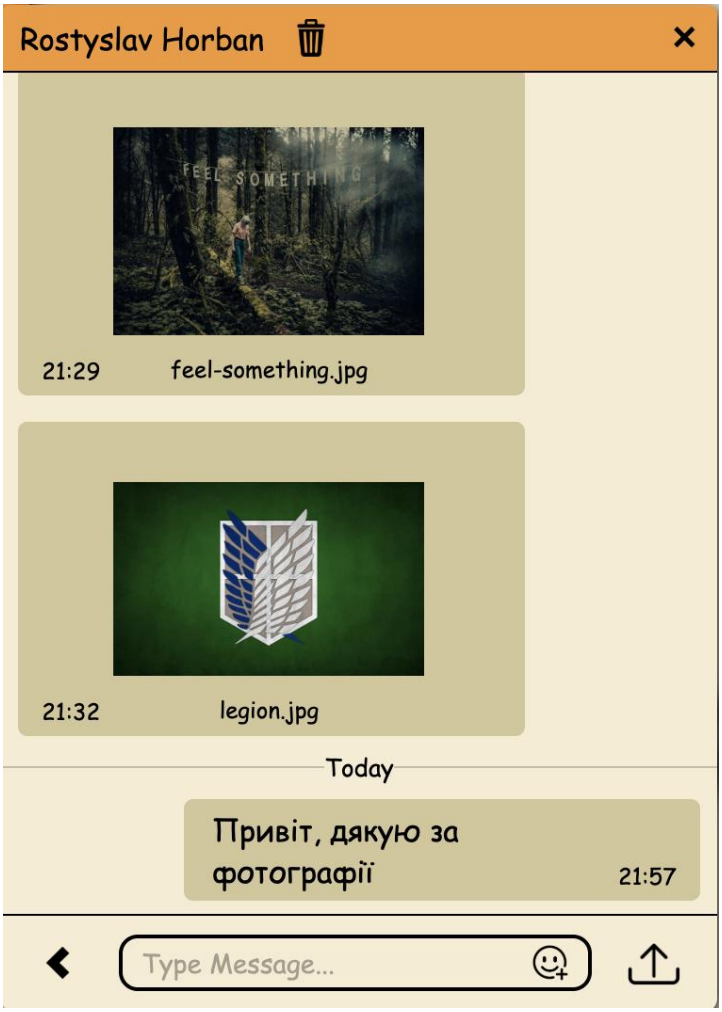


Рисунок 4.33 – Зображення списку повідомленнь з користувачем

На рисунку 4.33 показано вікно чату з користувачем **Rostyslav Horban**. У верхній частині є ім’я контакту, іконка видалення та кнопка закриття. Нижче видно надіслані зображення та відповідь на повідомлення У нижній частині інтерфейсу розташоване поле для введення повідомлень та кнопки надсилання й смайлів.

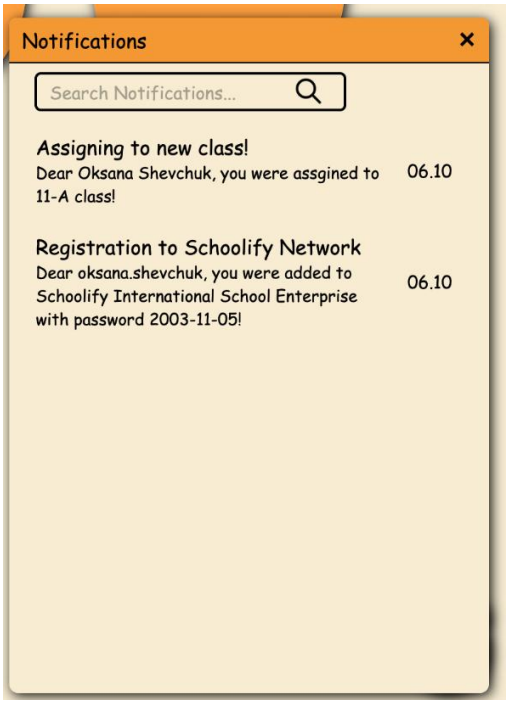


Рисунок 4.34 – Зображення списку повідомленнь з користувачем

На рисунку 4.34 показано вікно із переліком сповіщень. Угорі розташоване поле пошуку з іконкою лупи. Нижче відображено два сповіщення.

Висновки до розділу

У цьому розділі було реалізовано ключові компоненти програмного забезпечення інтелектуальної цифрової платформи для управління освітнім процесом, зокрема базу даних та клієнтську частину вебзастосунку. Створена структура бази даних забезпечує ефективне зберігання, обробку та взаємозв'язок даних про учнів, викладачів, навчальні заклади, документи та інші сутності системи. Було реалізовано механізми захисту даних, що гарантують цілісність і конфіденційність інформації.

Клієнтська частина розроблена з урахуванням принципів адаптивного дизайну та орієнтована на зручність користувача. Вона забезпечує швидкий і інтуїтивно зрозумілий доступ до основного функціоналу платформи, включаючи перегляд інформації, управління записами та інтерактивну взаємодію з системою. Застосовані технології дозволяють масштабувати рішення та легко розширювати його функціональність у майбутньому.

РОЗДІЛ 5. РОЗРОБЛЕННЯ СТАРТАП-ПРОЄКТУ

5.1 Опис ідеї проєкту

Ідея полягає у створенні багатофункціональної аплікації для комплексного менеджменту шкільного та освітнього процесу. Система дозволяє адміністрації, вчителям, учням та батькам ефективно взаємодіяти в межах єдиної платформи. Основні можливості включають:

- управління навчальними предметами, класами, розкладом та навчальними матеріалами;
- створення, видача та перевірка завдань — як вручну, так і за допомогою інструментів штучного інтелекту;
- автоматичний розрахунок оцінок, статистики успішності, середнього балу за предметами та семестром;

надання користувачам зручного інтерфейсу для комунікації й перегляду результатів.

Можливі напрямки застосування даного технологічного проєкту:

- **Шкільні заклади:** повний цифровий супровід навчального процесу, електронні журнали, розклад, контроль виконання домашніх завдань.
- **Приватні школи та освітні центри:** оптимізація управління групами, курсами, викладачами та навчальними програмами.
- **Дистанційне навчання:** можливість перевірки робіт та спілкування без фізичної присутності.
- **Батьківський контроль:** доступ до статистики успішності, домашніх завдань та комунікації з учителями.
- **Самоосвітні платформи:** індивідуальні навчальні плани, відстеження прогресу, аналіз успішності.
- **Освітні установи (ліцеї, гімназії, коледжі):** уніфіковане адміністрування освітнього процесу та оцінювання.

Основні вигоди для користувача будь якої ролі:

- **Для вчителів:**

- економія часу завдяки автоматичній перевірці завдань ШІ;
- зручне управління класами та предметами;
- автоматичне формування журналів і підрахунок середніх балів.

- **Для учнів:**
  - чітке розуміння завдань, термінів виконання та результатів;
  - швидке отримання зворотного зв’язку;
  - можливість навчатись у власному темпі.
- **Для батьків:**
  - доступ до актуальної інформації про успішність;
  - сповіщення про важливі події та оцінки;
  - зниження ризику пропуску завдань чи погіршення успішності.
- **Для адміністрації школи:**
  - оптимізація документообігу;
  - централізований контроль над навчальним процесом;

Відмінність проєкту від існуючих аналогів:

- **Інтегрована перевірка завдань за допомогою ШІ**, що значно скорочує час оцінювання та підвищує об’єктивність.
- **Автоматичне аналітичне опрацювання результатів**: система сама вираховує середні бали, формує звіти та динаміку успішності.
- **Комбінація ручної та автоматизованої перевірки**, яка дає вчителю гнучкість та контроль.
- **Модульна структура**, що дозволяє адаптувати додаток під різні типи навчальних закладів.
- **Уніфіковане середовище взаємодії**, що замінює одразу кілька платформ (журнали, месенджери, електронний щоденник, тести).

5.2 Аналіз технологічних можливостей реалізації ідей проєкту

Найперше, що потрібно зробити в межах цього етапу реалізації стартап-проєкту це провести технологічний аудит, завдяки якому можливо реалізувати всі основні ідеї проєкту. Для цього необхідно проаналізувати наступні складові:

- технологія, через яку буде реалізовуватись проєкт;
- наявність необхідних технологій, чи необхідність їх творити/доробити;
- доступність технологій для розробників проєкту.

Таблиця 5.1 - Технологічна здійсненність ідеї проєкту

№ п/п	Ідея проєкту	Технології її реалізації	Наявність технологій	Доступність технологій
1.	Розробка інформаційної системи «Schoolify»	Java	Є у наявності	Безкоштовна. Відкрита для всіх операційних систем
		Spring Boot	Є у наявності	Безкоштовна. Розроблена для Java
		Open API	Є у наявності	Платна, проте доступна. Використовується платформах
		Postgres SQL	Є у наявності	Безкоштовна. Використовується на усіх платформах
		Mongo DB	Є у наявності	Безкоштовна. Використовується на усіх платформах
		Amazon S3	Є у наявності	Безкоштовна. Використовується на усіх платформах
		RabbitMQ / Amazon SQS	Є у наявності	Безплатна. Використовується на усіх платформах
		Angular	Є у наявності	Безкоштовна. Використовується на усіх платформах

Продовження таблиці 5.1

Обрана технологія реалізації проєкту: Java + Spring Boot + PostgreSQL + MongoDB + AWS + Angular + Open API

Отже, якшо зробити висновки з таблиці 5.1, реалізація ідеї продукту має право на успіх, адже технології, які є необхідними для реалізації проєкти. Майже всі технології є безкоштовними для розробників, а платні технології є доступними та не потребують суттєвих вкладень.

5.3 Розроблення ринкової стратегії

На даному етапі необхідно визначити стратегію охоплення ринку. Це включає: опис цільових груп потенційних споживачів продукту у таблиці 5.2, визначення стратегій конкурентної поведінки у таблиці 5.3, а також формування базових стратегій розвитку проєкту у таблиці 5.4.

Таблиця 5.2 - Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи	Простота входу в сегмент	Інтенсивність конкуренції в сегменті	Орієнтований попит в межах сегменту	Готовність споживачів сприйняти продукт
1.	Учні, студенти	Висока	Середня	Високий	Висока готовність
2.	Вчителі, викладачі, адміністратори	Висока	Середня	Високий	Висока готовність

Таблиця 5.3 - Визначення базової стратегії конкурентної поведінки

Чи є проєкт «першопрохідцем» на ринку?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Стратегія конкурентної поведінки
Представлений проєкт є першопрохідцем на ринку послуг, оскільки включає комплексну систему та інтеграцію з ШІ	Створення та реалізація проєкту побудований на власних рішеннях і характеристиках	Конкуретні позиції прогнозуються бути високими, оскільки конкуретних систем не є багато.	Стратегія орієнтації на клієнта

Таблиця 5.4 - Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проєкту	Стратегія охоплення ринку	Базова стратегія розвитку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи
1.	Створення інтелектуальної цифрової платформи для управління освітнім процесом	Стратегія концентрованого маркетингу	Стратегія широкої диференціації	Підвищення якості продукту

5.4 Розроблення маркетингової програми

Для ефективної та оперативної розробки маркетингової програми стартап-проєкту насамперед потрібно визначити ключові переваги концепції майбутнього

продукту для споживачів, а також узагальнити результати попереднього аналізу його конкурентоспроможності.

Таблиця 5.5 - Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Ключові переваги перед конкурентами	Вигода, яку пропонує товар
1.	Автоматизація та введення завдань, задач, шкільного графіку та розкладу	Комплексне рішення в одній системі, що спростовує комунікацію між користувачами та пришвидшує підготовку до нового навчального року	Можливість створювати задачі, щоденники, чати, предмети, розклади в уніфікованій зручній системі
2.	Автоматизація перевірки завдань за допомогою системи та штучного інтелекту для об’єктивного та швидшого оцінювання	Відсутність перелічених функцій у конкурентів	Зменшення витрат часу та ресурсів персоналу закладу для ключових освітніх процесів

Після окреслення основних переваг концепції майбутнього продукту важливо перейти до розроблення трирівневої моделі товару. Така модель дає змогу глибше конкретизувати та деталізувати ідею проєкту, визначити його сутність, функціональні можливості та додаткову цінність для користувача.

Таблиця 5.6 - Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові
1. Продукт за задумом	Розроблення інформаційної системи “Schoolify” з комплексним рішенням для автоматизації освітніх процесів та комунікації.

Продовження таблиці 5.6

Рівні товару	Сутність та складові
2. Продукт у реальних умовах	Наділений властивостями: Можливість створення та адміністрування шкільних процесів, можливість створення та автоматичної перевірки завдань та задач. Автоматизація підрахунку балів системою. Можливість комунікацій між користувачами та створення діалогій, груп. Надходження сповіщень.
3. Товар із підкріпленням	Постійна підтримка проєкту, модернізація та додання нового функціоналу у майбутньому

На наступному етапі необхідно визначити оптимальну систему збуту продукту. Це передбачає ухвалення низки важливих рішень, зокрема: вибір посередників та обґрунтування доцільності їх використання, визначення найефективнішої для проєкту глибини каналу збуту, а також аналіз доцільності здійснення продажів власними силами чи із залученням зовнішніх партнерів. Такий підхід дозволить сформувати цілісну модель просування продукту на ринку та забезпечити найбільш ефективне охоплення цільової аудиторії.

Таблиця 5.7 - Формування системи збуту

Специфіка закупівельної поведінки цільових клієнтів	Глибина каналу збуту	Оптимальна система збуту	Функції збуту, які має виконувати постачальник товару
Клієнти та споживачі мають на меті отримувати товар за нижчою ціною та за рахунок провадження конкуренції	Канали збуту нульового рівня	Прямі канали розподілу (власноруч) та за допомогою посередників	Продаж товару потенційним клієнтам (надання доступу, можливість користування), а також надання безплатної пробної версії товару

Таблиця 5.8 - Концепція маркетингових комунікацій

Специфіка поведінки цільових клієнтів	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Канали комунікацій	Концепція рекламного звернення
Пошук пропонованих товарів через мережу та рекламу	Забезпечення конкурентного середовища, спостерігання та прогнозування розвитку інформаційної системи, підвищення якості	Представлення переваг продукту та ефективність його роботи з чітким акцентом на цінове та якісне співвідношення	Інтернет, комунікаційні системи	Доступність, зручність, комплексне вирішення проблем

Висновки до розділу

За підсумками проведених розрахунків можна зробити висновок, що комплексна інформаційна система **“Schoolify”** є економічно обґрунтованою та має високий потенціал для успішного виходу на ринок. Було здійснено всебічний аналіз продукту, а також оцінено можливості його впровадження та подальшого функціонування як стартап-проєкту. На початковому етапі було досліджено попит і визначено потреби потенційних користувачів.

Створено ринкову стратегію із чітким визначенням цільових сегментів та оцінкою конкурентного середовища. Окрім цього, визначено ключові стратегії конкурентної поведінки та розвитку. Розроблена маркетингова стратегія дала змогу окреслити основні переваги системи у порівнянні з існуючими аналогами.

Отже, базуючись на результатах економічного аналізу, можна впевнено стверджувати, що вихід цього стартап-проєкту на ринок є доцільним і перспективним. Система має всі передумови для швидкого знаходження своєї цільової аудиторії та отримання провідних позицій у відповідному сегменті ринку.

ВИСНОВКИ

У процесі виконання магістерської роботи на тему **«Інтелектуальна цифрова платформа для управління освітнім процесом»** було проведено всебічний аналіз проблемної області, що дозволило чітко окреслити актуальні потреби сучасних закладів освіти та визначити функціональні, технологічні й організаційні вимоги до створюваної системи. На основі результатів цього аналізу сформовано ґрунтовний технічний проєкт, який охоплював розробку архітектури платформи, обґрунтування вибору інструментів та технологій, а також планування етапів реалізації та інтеграції рішення.

Відповідно до розробленого проєкту створено вебзастосунок, здатний забезпечувати комплексну підтримку освітнього процесу, включаючи ведення даних про учнів, викладачів, навчальні групи, освітні матеріали та інші компоненти інформаційної інфраструктури закладу. Під час розробки особливу увагу приділено питанням інформаційної безпеки, надійності зберігання та передачі даних, а також розробленню інтерфейсу, орієнтованого на зручність і доступність для різних категорій користувачів.

Проведене тестування підтвердило відповідність функціоналу платформи поставленим вимогам, а також продемонструвало її ефективність у контексті реалізації основних задач цифровізації освітнього середовища. Отримані результати засвідчують доцільність обраних технічних рішень та підходів, а також підтверджують успішність досягнення мети та виконання завдань магістерської роботи.

Розроблена інтелектуальна цифрова платформа може бути використана як основа для подальшого розширення функціональності, інтеграції модулів аналітики, адаптивного навчання та технологій штучного інтелекту, що відкриває перспективи для подальших досліджень і практичного впровадження в закладах освіти.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бородкіна І. Л., Бородкін Г. О. Web-технології та Web-дизайн: застосування мови HTML для створення електронних ресурсів, 2020. – 212 с.;
2. Копитко М.Ф., Іванків К.С. Основи програмування мовою Java: Тексти лекцій. – Львів: Видавничий центр ЛНУ ім. Івана Франка, 2002. – 83 с.;
3. Free Learning Platform For Better Future [Електронний ресурс] – Режим доступу до ресурсу: https://www.javatpoint.com/disadvantage-of-multithreading-in-java (дата звернення: 02.04.2025);
4. Eric A. Meyer, Estelle Weyl, CSS: The Definitive Guide: Web Layout and Presentation [5th Edition]; 2023. – 1126 c.;
5. Jon Duckett, JavaScript and jQuery: Interactive Front-End Web Development [1st Edition]; 2018. – 640 c.;
6. Herbert Schildt , Danny Coward, Java: The Complete Reference [13th Edition]; 2024. - 1280c.;
7. Bauer, Christian, Gavin King, and Gary Gregory. Java Persistence with Hibernate. 2nd ed. Shelter Island: Manning Publications, 2015;
8. Рейтинг мов програмування 2024 [Електронний ресурс] – Режим доступу до ресурсу: https://dou.ua/lenta/articles/language-rating-2024/ (дата звернення: 26.06.2025);
9. Nick Morgan, JavaScript Crash Course: A Hands-On, Project-Based Introduction to Programming, 2024. – 376 c.;
10. IntelliJ IDEA: The Leading Java and Kotlin IDE [Електронний ресурс] – Режим доступу до ресурсу: https://www.jetbrains.com/idea (дата звернення: 16.05.2025).
11. Васильєв О. М. Програмування мовою Java: Видавництво Навчальна книга – Богдан, 2020. – 696 с.;
12. Spring Boot [Електронний ресурс] – Режим доступу до ресурсу: https://docs.spring.io/spring-boot/index.html (дата звернення: 25.08.2025).

ДОДАТКИ

ДОДАТОК А

```
<div class="content">
  <div id="input-container" class="input-container">
    <input id="class-search-input" class="search-input" placeholder="Search Class..."
           #classInput (keydown.enter)="searchTable()"
(input)="refetch(classInput.value)">
    <i (click)="searchTable()" class="search-icon"></i>
    <i *ngIf="hasWritePermission()" (click)="showCreate()" class="create-icon"></i>
    <i *ngIf="hasWritePermission() && clickedRecord" (click)="showUpdate()"
class="edit-icon"></i>
    <div *ngIf="hasWritePermission()" class="import-classes"
(click)="fileImport.click()">
      <input #fileImport (change)="upload()" style="height: 100%; width: 100%;
display: none" type="file"/>
      Import Classes
    </div>
    <div class="import-classes" (click)="download('SCHOOL_CLASS')">
      Export Classes
    </div>
    <div class="import-classes" (click)="download('SCHOOL_ENTERPRISE')">
      Export School
    </div>
    <button *ngIf="hasWritePermission()" class="download-import-template">
      <a class="download-link" href="/templates/import-classes-
template.xlsx">Download Template</a>
    </button>
    <schoolify-create-school-class #createSchoolClassComponent
*ngIf="showCreateWindow$ | async"
                                   [visible$]="showCreateWindow$">
    </schoolify-create-school-class>
    <schoolify-update-school-class *ngIf="showUpdateWindow$ | async"
                                   #updateSchoolClassComponent
                                   [visible$]="showUpdateWindow$"
                                   [schoolClassId]="clickedRecord.id">
    </schoolify-update-school-class>
    <mat-paginator class="paginator"
                   [pageSize]="pageSize"
                   [length]="totalRecords"
                   [pageIndex]="pageIndex"
                   (page)="onPageChange($event)"
                   showFirstLastButtons
                   aria-label="Select page of periodic elements">
    </mat-paginator>
  </div>
  <table-scrolling-viewport [itemSize]="pageSize" [totalItems]="totalRecords">
    <table mat-table [dataSource]="page" matSort class="mat-elevation-z8 table">
      <ng-container matColumnDef="identifier">
        <th mat-header-cell *matHeaderCellDef>#</th>
        <td mat-cell *matCellDef="let i = index">{{ (i + 1) + (pageIndex * pageSize)
}}</td>
```

```
      </ng-container>
      <ng-container matColumnDef="name">
        <th mat-header-cell *matHeaderCellDef mat-sort-header>Name</th>
        <td mat-cell *matCellDef="let schoolClass">{{ schoolClass.name }}</td>
      </ng-container>
      <ng-container matColumnDef="grade">
        <th mat-header-cell *matHeaderCellDef mat-sort-header>Grade</th>
        <td mat-cell *matCellDef="let schoolClass">{{ schoolClass.grade }}</td>
      </ng-container>
      <ng-container matColumnDef="section">
        <th mat-header-cell *matHeaderCellDef mat-sort-header>Section</th>
        <td mat-cell *matCellDef="let schoolClass">{{ schoolClass.section }}</td>
      </ng-container>
      <ng-container matColumnDef="schoolClassType">
        <th mat-header-cell *matHeaderCellDef mat-sort-header>Class Type</th>
        <td mat-cell *matCellDef="let schoolClass">{{
StringUtils.humanizeEnum(schoolClass.schoolClassType) }}</td>
      </ng-container>
      <ng-container matColumnDef="code">
        <th mat-header-cell *matHeaderCellDef mat-sort-header>Code</th>
        <td mat-cell *matCellDef="let schoolClass">{{ schoolClass.code || 'N/A'
}}</td>
      </ng-container>
      <ng-container matColumnDef="capacity">
        <th mat-header-cell *matHeaderCellDef mat-sort-header>Capacity</th>
        <td mat-cell *matCellDef="let schoolClass">{{ schoolClass.capacity || 'N/A'
}}</td>
      </ng-container>
      <ng-container matColumnDef="headTeacherUsername">
        <th mat-header-cell *matHeaderCellDef mat-sort-header>Head Teacher
Username</th>
        <td mat-cell *matCellDef="let schoolClass">{{ schoolClass.headTeacherUsername
|| 'N/A' }}</td>
      </ng-container>
      <ng-container matColumnDef="headTeacherFirstname">
        <th mat-header-cell *matHeaderCellDef mat-sort-header>Head Teacher First
Name</th>
        <td mat-cell *matCellDef="let schoolClass">{{
schoolClass.headTeacherFirstname || 'N/A' }}</td>
      </ng-container>
      <ng-container matColumnDef="headTeacherLastname">
        <th mat-header-cell *matHeaderCellDef mat-sort-header>Head Teacher Last
Name</th>
        <td mat-cell *matCellDef="let schoolClass">{{ schoolClass.headTeacherLastname
|| 'N/A' }}</td>
      </ng-container>
      <ng-container matColumnDef="headTeacherFatherName">
        <th mat-header-cell *matHeaderCellDef mat-sort-header>Head Teacher Father
Name</th>
        <td mat-cell *matCellDef="let schoolClass">{{
schoolClass.headTeacherFatherName || 'N/A' }}</td>
      </ng-container>
      <ng-container matColumnDef="headPupilUsername">
        <th mat-header-cell *matHeaderCellDef mat-sort-header>Head Pupil
Username</th>
```

```
        <td mat-cell *matCellDef="let schoolClass">{{ schoolClass.headPupilUsername
|| 'N/A' }}</td>
      </ng-container>
      <ng-container matColumnDef="headPupilFirstname">
        <th mat-header-cell *matHeaderCellDef mat-sort-header>Head Pupil First
Name</th>
        <td mat-cell *matCellDef="let schoolClass">{{ schoolClass.headPupilFirstname
|| 'N/A' }}</td>
      </ng-container>
      <ng-container matColumnDef="headPupilLastname">
        <th mat-header-cell *matHeaderCellDef mat-sort-header>Head Pupil Last
Name</th>
        <td mat-cell *matCellDef="let schoolClass">{{ schoolClass.headPupilLastname
|| 'N/A' }}</td>
      </ng-container>
      <ng-container matColumnDef="headPupilFatherName">
        <th mat-header-cell *matHeaderCellDef mat-sort-header>Head Pupil Father
Name</th>
        <td mat-cell *matCellDef="let schoolClass">{{ schoolClass.headPupilFatherName
|| 'N/A' }}</td>
      </ng-container>
      <ng-container matColumnDef="pupilCount">
        <th mat-header-cell *matHeaderCellDef mat-sort-header>Pupil Count</th>
        <td mat-cell *matCellDef="let schoolClass">{{ schoolClass.pupilCount }}</td>
      </ng-container>
      <tr mat-header-row *matHeaderRowDef="schoolClassDisplayedColumns"></tr>
      <tr mat-row (click)="handleRowClick(schoolClass)" (dblclick)="showUpdate()"
          [class.row-is-clicked]="isRowClicked(schoolClass)"
          *matRowDef="let schoolClass; columns: schoolClassDisplayedColumns;">
      </tr>
    </table>
  </table-scrolling-viewport>
</div>
```

ДОДАТОК Б

```
<div class="content">
  <div id="input-container" class="input-container">
    <input id="enterprise-input" class="search-input"
[placeholder]="'type_enterprise_name' | translate"
           #enterpriseInput (keydown.enter)="searchTable()"
(input)="refetch(enterpriseInput.value)">
    <i (click)="searchTable()" class="search-icon"></i>
    <i *ngIf="hasWritePermission()" (click)="showCreate()" class="create-icon"></i>
    <i *ngIf="clickedRecord && hasWritePermission()" (click)="showUpdate()"
class="edit-icon"></i>
    <label *ngIf="clickedRecord && hasWritePermission()" class="switcher">
      <input type="checkbox" (click)="switch()"
[checked]="instantinateRecord().active" #enterpriseActiveSwitcher>
      <span class="slider"></span>
    </label>
    <button class="enterprise-checkout-button" *ngIf="clickedRecord &&
instantinateRecord().active" (click)="checkout()">
      {{ 'checkout' | translate }}
    </button>
```

```
    <ng-container *ngIf="(currentEnterprise$ | async) as currentEnterprise">
      <schoolify-create-enterprise #createEnterpriseComponent
*ngIf="showCreateWindow$ | async"
                                   [visible$]="showCreateWindow$"
                                   [currentEnterprise]="currentEnterprise">
      </schoolify-create-enterprise>
      <schoolify-update-enterprise *ngIf="showUpdateWindow$ | async"
                                   #updateEnterpriseComponent
                                   [visible$]="showUpdateWindow$"
                                   [enterpriseId]="clickedRecord.id"
                                   [currentEnterprise]="currentEnterprise">
      </schoolify-update-enterprise>
    </ng-container>
    <mat-paginator class="paginator"
                   [pageSize]="pageSize"
                   [length]="totalRecords"
                   [pageIndex]="pageIndex"
                   (page)="onPageChange($event)"
                   showFirstLastButtons
                   aria-label="Select page of periodic elements">
    </mat-paginator>
  </div>
  <table-scrolling-viewport [itemSize]="pageSize" [totalItems]="totalRecords">
      <table mat-table style="overflow-y: hidden;" [dataSource]="page" matSort
class="mat-elevation-z8 table">
          <ng-container matColumnDef="identifier">
              <th mat-header-cell *matHeaderCellDef>#</th>
              <td mat-cell *matCellDef="let i = index">{{ (i + 1) + (pageIndex *
pageSize) }}</td>
          </ng-container>
        <ng-container matColumnDef="enterpriseIdentifier">
          <th mat-header-cell *matHeaderCellDef mat-sort-header>{{'identifier' |
translate}}</th>
          <td mat-cell *matCellDef="let enterprise">{{
enterprise.enterpriseIdentifier }}</td>
        </ng-container>
          <ng-container matColumnDef="name">
              <th mat-header-cell *matHeaderCellDef mat-sort-header>{{'name' |
translate}}</th>
              <td mat-cell *matCellDef="let enterprise">{{ enterprise.name }}</td>
          </ng-container>
          <ng-container matColumnDef="location">
              <th mat-header-cell *matHeaderCellDef mat-sort-header>{{'location' |
translate}}</th>
              <td mat-cell *matCellDef="let enterprise">{{ enterprise.location
}}</td>
          </ng-container>
        <ng-container matColumnDef="country">
          <th mat-header-cell *matHeaderCellDef mat-sort-header>{{'country' |
translate}}</th>
          <td mat-cell *matCellDef="let enterprise">{{ enterprise.country | translate
}}</td>
        </ng-container>
          <ng-container matColumnDef="enterpriseType">
              <th mat-header-cell *matHeaderCellDef mat-sort-header>{{'type' |
```

```
translate}}</th>
              <td mat-cell *matCellDef="let enterprise">{{ enterprise.enterpriseType
| translate }}</td>
          </ng-container>
          <ng-container matColumnDef="corporateEmail">
              <th mat-header-cell *matHeaderCellDef mat-sort-
header>{{'corporate_email' | translate}}</th>
              <td mat-cell *matCellDef="let enterprise">{{ enterprise.corporateEmail
}}</td>
          </ng-container>
          <ng-container matColumnDef="supportEmail">
              <th mat-header-cell *matHeaderCellDef mat-sort-header>{{'support_email'
| translate}}</th>
              <td mat-cell *matCellDef="let enterprise">{{ enterprise.supportEmail
}}</td>
          </ng-container>
          <ng-container matColumnDef="active">
              <th mat-header-cell *matHeaderCellDef>{{'active_column' |
translate}}</th>
              <td mat-cell *matCellDef="let enterprise">{{ (enterprise.active ?
'active' : 'inactive') | translate }}</td>
          </ng-container>
          <tr mat-header-row *matHeaderRowDef="enterpriseDisplayedColumns"></tr>
          <tr mat-row (click)="handleRowClick(enterprise)" (dblclick)="showUpdate()"
              [class.row-is-clicked]="isRowClicked(enterprise)"
              *matRowDef="let enterprise; columns: enterpriseDisplayedColumns;">
          </tr>
      </table>
  </table-scrolling-viewport>
</div>
```

ДОДАТОК В

```
package com.schoolify.enterprise.service.services;

import com.schoolify.enterprise.service.api.models.CreateEnterpriseRequest;
import com.schoolify.enterprise.service.api.models.UpdateEnterpriseRequest;
import com.schoolify.enterprise.service.exceptions.EnterpriseDeactivationException;
import com.schoolify.enterprise.service.exceptions.EnterpriseExistsException;
import com.schoolify.enterprise.service.mappers.EnterpriseMapper;
import com.schoolify.enterprise.service.models.enterprise.Enterprise;
import com.schoolify.enterprise.service.models.enterprise.EnterpriseType;
import com.schoolify.enterprise.service.repositories.EnterpriseRepository;
import com.schoolify.shared.service.Dropdown;
import com.schoolify.shared.service.SwitchEnterpriseMessage;
import com.schoolify.shared.service.constants.EnterpriseScope;
import com.schoolify.shared.service.exceptions.FileUploadException;
import com.schoolify.shared.service.managers.CachedEnterpriseManager;
import com.schoolify.shared.service.models.EnterpriseSnapshot;
import com.schoolify.shared.service.models.attachments.AttachmentType;
import com.schoolify.shared.service.services.BasePersistenceService;
import com.schoolify.shared.service.utils.AuthenticationUtils;
import com.schoolify.shared.service.utils.TransactionUtils;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
```

```
import org.apache.commons.lang3.StringUtils;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.messaging.simp.SimpMessagingTemplate;
import org.springframework.security.access.AccessDeniedException;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import org.springframework.util.CollectionUtils;
import org.springframework.web.multipart.MultipartFile;

import java.util.List;
import java.util.Objects;
import java.util.UUID;

import static
com.schoolify.enterprise.service.configuration.WebSocketConfig.WEBSOCKET_ENTERPRISE_A
CTIVE_BROKER;
import static com.schoolify.enterprise.service.constants.ExceptionMessages.*;

@Slf4j
@Service
@RequiredArgsConstructor
public class EnterpriseService extends BasePersistenceService<Enterprise> {

  private final EnterpriseRepository enterpriseRepository;
  private final EnterpriseAttachmentService enterpriseAttachmentService;
  private final CachedEnterpriseManager cachedEnterpriseManager;
  private final KeycloakGroupService keycloakGroupService;
  private final SimpMessagingTemplate messagingTemplate;
  private final EnterpriseMapper enterpriseMapper;

  @Value("${spring.rabbitmq.exchange.switch-enterprise}")
  private String switchEnterpriseExchange;

  @Transactional(noRollbackFor = FileUploadException.class)
  public Enterprise save(CreateEnterpriseRequest request, MultipartFile file) throws
FileUploadException {
    String currentEnterpriseType =
cachedEnterpriseManagerFacade.currentEnterprise().getEnterpriseType();
    if (StringUtils.equalsAny(currentEnterpriseType, EnterpriseType.SCHOOL.name(),
EnterpriseType.DEPARTMENT.name())) {
      throw new AccessDeniedException(SCHOOL_CREATE_ACCESS_DENIED);
    }
    UUID userEnterpriseId = AuthenticationUtils.userRootEnterprise();
    boolean exists = enterpriseRepository.existsInScope(userEnterpriseId,
request.getName(), request.getCorporateEmail());
    if (exists) {
      throw new EnterpriseExistsException(String.format(ENTERPRISE_ALREADY_EXISTS,
request.getName()));
    }
    Enterprise enterprise = enterpriseMapper.asEnterprise(request);
    Enterprise savedEnterprise = save(enterprise);
    if (Objects.nonNull(file)) {
      enterpriseAttachmentService.save(savedEnterprise, AttachmentType.AVATAR, file);
    }
    keycloakGroupService.create(savedEnterprise);
```

```
    cachedEnterpriseManager.clearEnterpriseScopeCache();
    log.info("Created Enterprise with ID: {}", savedEnterprise.getId());
    return savedEnterprise;
  }

 @Transactional(noRollbackFor = FileUploadException.class)
  public Enterprise update(UUID enterpriseId, UpdateEnterpriseRequest request,
MultipartFile file) throws FileUploadException {
    boolean exists = enterpriseRepository.existsInScope(request.getName(),
request.getCorporateEmail(), enterpriseId);
    if (exists) {
      throw new EnterpriseExistsException(String.format(ENTERPRISE_ALREADY_EXISTS,
request.getName()));
    }
    Enterprise existent = getExistent(enterpriseId);
    Enterprise updatedEnterprise = enterpriseMapper.asEnterprise(existent, request);
    Enterprise savedEnterprise = save(updatedEnterprise);
    if (Objects.nonNull(file)) {
      enterpriseAttachmentService.updateAvatar(savedEnterprise, file);
    }
    keycloakGroupService.update(savedEnterprise);
    cachedEnterpriseManager.clearEnterpriseScopeCache();
    log.info("Updated Enterprise with ID: {}", savedEnterprise.getId());
    return savedEnterprise;
  }

  @Transactional
  public void activate(UUID enterpriseId) {
    SwitchEnterpriseMessage message = new
SwitchEnterpriseMessage(List.of(enterpriseId), true);
    enterpriseRepository.activateEnterprise(enterpriseId);
    rabbitService.send(switchEnterpriseExchange, message);
    log.info("Activated Enterprise with ID: {}", enterpriseId);
  }

  @Transactional
  public void deactivate(UUID enterpriseId) {
    if
(cachedEnterpriseManagerFacade.currentEnterprise().getEnterpriseId().equals(enterpris
eId)) {
      throw new EnterpriseDeactivationException();
    }
    List<UUID> enterpriseIds = getEnterprisesByScope(enterpriseId,
EnterpriseScope.CURRENT_DEEP_CHILDREN);
    SwitchEnterpriseMessage message = new SwitchEnterpriseMessage(enterpriseIds,
false);
    enterpriseRepository.deactivateEnterprises(enterpriseIds);
    rabbitService.send(switchEnterpriseExchange, message);
    log.info("Deactivated Enterprises with ID: {}", enterpriseIds);
    TransactionUtils.afterCommit(() ->
messagingTemplate.convertAndSend(WEBSOCKET_ENTERPRISE_ACTIVE_BROKER, enterpriseIds));
  }

  public EnterpriseSnapshot currentEnterprise() {
    return cachedEnterpriseManagerFacade.currentEnterprise();
```

```
  }

  public Enterprise currentEnterpriseEntity() {
    EnterpriseSnapshot enterpriseSnapshot =
cachedEnterpriseManagerFacade.currentEnterprise();
    return getExistent(enterpriseSnapshot.getEnterpriseId());
  }

  public EnterpriseSnapshot cacheRootEnterprise() {
    UUID currentEnterpriseId = AuthenticationUtils.userRootEnterprise();
    Enterprise enterprise = getExistent(currentEnterpriseId);
    EnterpriseSnapshot currentSnapshot =
enterpriseMapper.asEnterpriseSnapshot(enterprise, AuthenticationUtils.currentUser());
    return cachedEnterpriseManager.saveCurrentEnterprise(currentSnapshot);
  }

  public EnterpriseSnapshot checkout(UUID enterpriseId) {
    Enterprise enterprise = getExistent(enterpriseId);
    if (!enterprise.isActive()) {
      throw new IllegalStateException(ENTERPRISE_CHECKOUT_INACTIVE);
    }
    EnterpriseSnapshot currentSnapshot =
enterpriseMapper.asEnterpriseSnapshot(enterprise, AuthenticationUtils.currentUser());
    cachedEnterpriseManager.clearEnterpriseScopeCache();
    return cachedEnterpriseManager.saveCurrentEnterprise(currentSnapshot);
  }

  public List<Dropdown> getDropdownList() {
    UUID currentEnterprise = currentEnterprise().getEnterpriseId();
    return enterpriseRepository.findAllEnterpriseDropdowns(currentEnterprise);
  }

  public List<UUID> getCurrentEnterprisesByScope(EnterpriseScope enterpriseScope) {
    UUID currentEnterprise = currentEnterprise().getEnterpriseId();
    List<UUID> enterpriseIds = getEnterprisesByScope(currentEnterprise,
enterpriseScope);
    return cachedEnterpriseManager.putEnterprises(enterpriseIds, enterpriseScope);
  }

  public List<UUID> getEnterprisesByScope(UUID enterpriseId, EnterpriseScope
enterpriseScope) {
    return getEnterpriseIdsByScope(enterpriseId, enterpriseScope);
  }

  public boolean hasEnterpriseAccess(UUID enterpriseId) {
    UUID currentEnterprise = currentEnterprise().getEnterpriseId();
    List<UUID> userEnterpriseScopeAccess =
enterpriseRepository.findAllDeepTreeEnterprises(currentEnterprise);
    boolean hasAccess =
CollectionUtils.contains(userEnterpriseScopeAccess.iterator(), enterpriseId);
    if (hasAccess) {
      cachedEnterpriseManager.putEnterprises(userEnterpriseScopeAccess,
EnterpriseScope.DEEP_TREE);
    }
    return hasAccess;
```

```
  }

  public boolean hasEnterpriseCheckoutAccess(UUID enterpriseId) {
    UUID currentEnterprise = AuthenticationUtils.userRootEnterprise();
    List<UUID> userEnterpriseScopeAccess =
enterpriseRepository.findAllDeepChildrenEnterprises(currentEnterprise);
    return CollectionUtils.contains(userEnterpriseScopeAccess.iterator(),
enterpriseId);
  }

  private List<UUID> getEnterpriseIdsByScope(UUID enterpriseId, EnterpriseScope
enterpriseScope) {
    return switch (enterpriseScope) {
      case CURRENT -> List.of(currentEnterprise().getEnterpriseId());
      case CURRENT_CHILDREN ->
enterpriseRepository.findAllChildrenEnterprises(enterpriseId);
      case CURRENT_DEEP_CHILDREN ->
enterpriseRepository.findAllDeepChildrenEnterprises(enterpriseId);
      case PARENT_CURRENT ->
enterpriseRepository.findAllParentEnterprises(enterpriseId);
      case DEEP_PARENT_CURRENT ->
enterpriseRepository.findAllDeepParentEnterprises(enterpriseId);
      case TREE -> enterpriseRepository.findAllTreeEnterprises(enterpriseId);
      case DEEP_TREE ->
enterpriseRepository.findAllDeepTreeEnterprises(enterpriseId);
    };
  }

  @Override
  protected EnterpriseRepository getRepository() {
    return enterpriseRepository;
  }
}
```

ДОДАТОК Г

```
public void calculateAssignmentResult(AssigneeAnswer assigneeAnswer) {
    if (assigneeAnswer == null) {
      return;
    }
    Assignment assignment = Optional.ofNullable(assigneeAnswer.getAssignmentTask())
        .map(AssignmentTask::getAssignment)
        .orElse(null);
    if (ObjectUtils.anyNull(assignment, assigneeAnswer.getMark())) {
      return;
    }
    Result result =
resultRepository.findByAuthorIdAndAssignmentId(assigneeAnswer.getCreatedBy(),
assignment.getId());
    if (result == null) {
```

```
      String calculatedMark = calculateAssignmentMark(EMPTY_STRING, assigneeAnswer);
      Result newResult = resultMapper.asResult(assigneeAnswer, calculatedMark);
      save(newResult);
    } else {
      String currentMarkValue = calculateAssignmentMark(result.getAssignmentMark(),
assigneeAnswer);
      result.setAssignmentMark(currentMarkValue);
      result.getInspectors().add(assigneeAnswer.getMark().getInspector());
      save(result);
    }
  }

  private String calculateAssignmentMark(String currentMark, AssigneeAnswer
assigneeAnswer) {
    String markValue = assigneeAnswer.getMark().getMarkValue().getValue();
    float koef = assigneeAnswer.getAssignmentTask().getRateKoeficient();
    if (StringUtils.isEmpty(currentMark)) {
      if (NumberUtils.isCreatable(markValue)) {
        float numberMark = NumberUtils.createFloat(markValue);
        return String.valueOf(Math.round(numberMark * koef));
      }
      String americanMark = AMERICAN_EDUCATIONAL_GRADE_SYSTEM.stream()
          .filter(mark -> StringUtils.equalsIgnoreCase(markValue, mark))
          .findFirst()
          .orElseThrow();

      return ResultUtils.calculateAmericanMark(americanMark, koef);
    }
    if (NumberUtils.isCreatable(markValue)) {
      float numberMark = NumberUtils.createFloat(markValue);
      float taskMark = numberMark * koef;
      float calculatedMark = Float.parseFloat(currentMark) + taskMark;
      return String.valueOf(Math.round(calculatedMark));
    }
    String americanMark = AMERICAN_EDUCATIONAL_GRADE_SYSTEM.stream()
        .filter(mark -> StringUtils.equalsIgnoreCase(markValue, mark))
        .findFirst()
        .orElseThrow();

    String interpolatedMark = ResultUtils.calculateAmericanMark(americanMark, koef);
    return ResultUtils.addAmericanMark(currentMark, interpolatedMark);
   }
```