МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

15.03 — КМР. 1939 –“С” 2022.12.30. 18 ПЗ

ПОНЗЕЛЯ ДМИТРА ЮРІЙОВИЧА

2023 р.

НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

УДК 004.9:37.018.43

«ПОГОДЖЕНО»	**«ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ»**
Декан факультету	Завідувач кафедри
інформаційних технологій	комп'ютерних наук
Глазунова О.Г., д.п.н., професор	Голуб Б.Л., к.т.н., доцент

р. **____ _____ 202_ р.**

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Система для дистанційного навчання»

Спеціальність: 121 – «Інженерія програмного забезпечення»

Освітня програма – «Програмне забезпечення інформаційних систем»

Орієнтація освітньої програми: Освітньо-професійна

Гарант освітньої програми

ст. викладач _____ Міловідов Ю. О.
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник магістерської кваліфікаційної роботи

ст. викладач _____ Міловідов Ю. О.
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ Понзель Д. Ю.
(підпис) (ПІБ студента)

КИЇВ - 2023

НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук
доцент, к. т. н _____ _____ Голуб Б. Л.
(науковий ступінь, вчене звання) (підпис) (ПІБ)
"____" _____20_____року

ЗАВДАННЯ
ДО ВИКОНАННЯ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
СТУДЕНТУ ПОНЗЕЛЮ ДМИТРУ ЮРІЙОВИЧУ

Спеціальність: «121 – Інженерія програмного забезпечення»

Освітня програма: Програмне забезпечення інформаційних систем

Орієнтація освітньої програми: освітньо-професійна

Тема магістерської кваліфікаційної роботи: «Система для дистанційного навчання» затверджена наказом ректора НУБіП України від "____"_____20___р. №_____

Термін подання завершеної роботи на кафедру: _____
(рік, місяць, число)

Вихідні дані до магістерської кваліфікаційної роботи: інформація про оцінки, відвідуваність, структуру університету, тощо.

Перелік питань, що підлягають дослідженню:

- изначення рівня успішності студентів залежно від року вступу.
- изначення відвідуваності студентів в залежності від року навчання та предмету.
- роектування БД.
- ослідження способів оптимізації високонавантаженої системи.
- ослідження роботи обраних технологій.

Дата видачі завдання "_____"_____20___ р.

Керівник магістерської кваліфікаційної роботи _____ Міловідов Ю. О.
(підпис) (прізвище та ініціали)

Завдання прийняв до виконання _____ Понзель Д. Ю.
(підпис) (прізвище та ініціали студента

ЗМІСТ

ВСТУП

З розвитком інтернету та в цілому інформаційних технологій все простішим стає навчання з дому. В 2020 році, коли світ поглинула епідемія, можливість навчання онлайн переросла в безальтернативний варіант здобуття освіти. На щастя, на даний момент ця проблема не стоїть так гостро, і відвідування занять очно не є небезпечним для життя та здоров’я в більшості країнах світу. Але це не стосується України. Через повномасштабне вторгнення росії, постійні обстріли території України в більшості регіонів держави навчання очно або неможливе, або обмежено можливе. Але це не є і не може бути причиною зупинки навчального процесу. Тому сервіси, які спрощують навчання онлайн є актуальними і повинні розвиватися, щоб створювати конкуренцію і стати рушієм прогресу.

Зараз викладачі проводять лекції у форматі відеоконференцій, використовуючи такі засоби, як Google Meet, Microsoft Teams, Discord. Зазвичай університет жорстко не регулює сервіс для проведення занять, тому в студентів часто виникають труднощі із пошуком посилання на відповідну конференцію. Також може виникнути проблема із поширенням ресурсів, таких як конспекти, методичні матеріали, завдання до лабораторних. Для того, щоб не використовувати різні сервіси (від Google Drive до Telegram) потрібно використовувати єдиний сервіс для доступу до всіх матеріалів.

Однією з проблем дистанційного навчання є те, що студенти можуть пропустити якусь важливу інформацію, для прикладу, про дедлайн здачі лабораторної роботи, або про те, що завтра самостійна робота. Також різні організаційні питання можуть вийти за межі уваги студента. В більшості випадків вся важлива інформація доноситься від старости до студентів, але бувають випадки, коли ця інформація просто губиться у величезній кількості повідомлень із різних месенджерів.

Якщо говорити про переваги навчання онлайн, то до них можна віднести можливість отримувати статистичні дані в реальному часі (як здобувачу освіти, так і викладачу). Якщо говорити про студента, то це в першу чергу доступ до оцінок, бо при навчанні офлайн немає доступу до журналу. Якщо говорити про викладача, то це можливість отримувати статистику про успішність учнів в залежності від різних критеріїв (рік вступу, місце народження, стать). Ці дані можуть бути корисними для керівництва ВНЗ для розуміння тенденцій в сфері навчання.

Підсумувавши, можна сказати, що об’єктом дослідження є процес навчання онлайн, а предметом – організація навчання онлайн.

Якщо говорити про мету, то вона полягає в тому, щоб створити веб застосунок, який спростить організацію онлайн навчання, використовуючи сучасні технології. Цей веб застосунок повинен працювати швидко та надійно і надавати всі необхідні інструменти для дистанційного навчання, а саме централізованість отримання даних та взаємодії із викладачами, можливість отримання сповіщень через популярні соціальні мережі, отримання статистичних даних для подальшого аналізу.

Завданням дипломного проекту є вивчення нових технологій, формулювання вимог, побудова моделі предметної області, розробка архітектури системи, побудова діаграм, проектування БД.

Під час виконання дипломної роботи будуть використовуватися мова програмування SQL, система управління базами даних MS SQL Server, середовище BI MS SQL Server. Для написання самого сайту (front-частини) буде використовуватися React, для backend сервісів буде використовуватися ASP .NET. Для пришвидшення роботи програми буде застосований Redis. Системою контролю версій буде Git, а місцем зберігання коду – GitHub. Як сервіс для хостингу застосунку будуть використовуватися хмарні сервіси Azure.

Зараз існує багато систем для допомоги організації дистанційного навчання, тому важко зробити продукт із великою кількістю наукової новизни.

Тим не менш, до новизни можна віднести можливість отримання сповіщень в Telegram. Також туди можна віднести можливість витягнути статистику в залежності від певних критеріїв.

Проект має наступну структуру розділів: вступ, системний аналіз предметної області, моделювання системи, розробка системи, результат дослідження, висновків та переліку використаних джерел.

ИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

ороткий опис предметної області

В наш час існує безліч систем для дистанційного навчання. Варіантів дуже багато під різні потреби. Є системи з обширним функціоналом, які дають доступ до великої кількості функцій. Такі системи зазвичай платні і деякі заклади не можуть їх собі дозволити. Тим більше в таких системах часто буває надлишковий функціонал. Бувають і застосунки з вузьким напрямленням.

Основним призначенням систем для дистанційного навчання є допомога налаштувати взаємозв’язок між студентами та викладачами, спростити комунікацію та дати єдине джерело інформації про заходи, документи, оцінювання.

Можна виділити основні типи систем для дистанційного навчання: онлайн-системи доступом через мережу, веб застосунки, додатки для мобільних пристроїв, цифрові системи управління навчанням, платформи з офлайн-функціоналом, онлайн-навчальний контент та інструменти для створення цифрового навчального контенту.

Цифрові системи управління навчанням, також системи дистанційного навчання (англ. Learning management system) – це система управління навчальною діяльністю, яка використовується для розробки, управління та поширення навчальних матеріалів із забезпеченням спільного доступу [1].

Особливістю LMS є те, що вони в більшості випадків добре інтегруються із SCORM-модулями. Sharable Content Object Reference Model (SCORM) – набір стандартів та специфікацій, розроблений для систем дистанційного навчання. Цей стандарт містить вимоги до організації навчального матеріалу та всієї системи дистанційного навчання. SCORM дозволяє забезпечити сумісність компонентів та можливість їх багаторазового використання: навчальний матеріал представлений окремими невеликими блоками, котрі можуть включатись у різні навчальні курси [2].

Даний набір стандартів зазвичай використовується не для навчання в ВНЗ, а і для підняття рівня знань співробітників в деяких компаніях, бо формат таких модулів не дуже підходить для студентів. Тому, на маю думку, така функціональність є надлишковою для системи навчання саме для ВНЗ.

В даному застосунку я не буду використовувати специфікацію LMS, бо цей стандарт передбачає покриття об’ємного функціоналу, що є надлишковим для даного типу роботи. Натомість я зосереджуся на помірній кількості функцій, але і додам такі, яких нема в невеликих системах, але які є корисні, наприклад, можливість відправити сповіщення через соцмережу.

1.2 Аналіз наявних рішень

На даний момент існує безліч рішень для організації дистанційного навчання. Деякі є потужними і багатофункціональними, деякі є невеликі і спрямовані конкретно на певну задачу.

Можна виділити основні системи для навчання, які користуються популярністю, і виділити основні їх відмінності.

– Moodle – це відкрите програмне забезпечення для створення електронних курсів і платформа для навчання. Вона надає різні інструменти для створення і курування курсами, обміну матеріалами та оцінки студентів.

  – це потужна система для навчання, яку використовують багато університетів і шкіл. Вона пропонує різноманітні функції для створення та керування курсами.

  є іншою популярною LMS, яка допомагає вчителям і студентам взаємодіяти та навчатися онлайн. Вона включає в себе інструменти для спільної роботи та оцінки.

  як частина Google Workspace for Education, Google Classroom дозволяє

вчителям створювати курси, додавати завдання та взаємодіяти зі студентами через Google Apps [3].

– edX – це платформа для масових відкритих онлайн-курсів (MOOC), яка пропонує курси від багатьох провідних університетів та інститутів.
– ще одна популярна MOOC-платформа, Coursera надає доступ до курсів великої кількості предметів, розроблених університетами та компаніями.
– Udemy - це платформа для навчання онлайн, де ви можете знайти курси на різноманітні теми, створені інструкторами зі всього світу.

Хоч кількість систем для навчання онлайн є дуже великою, але все одно можна виділити кількох фаворитів, які є надзвичайно популярними на території України, і детально пройтися по їх перевагах та недоліках, щоб зрозуміти, куди потрібно рухатися при виконанні дипломної роботи. Це будуть Moodle та Google Classroom – найпопулярніші системи для навчання в даний час.

1.2.1 Moodle

Moodle - це відкрите програмне забезпечення для системи управління навчанням (LMS), яке дозволяє створювати онлайн-курси і сприяє організації навчання та співпраці між вчителями і студентами.

В системі Moodle студент може бачити список своїх предметів і має змогу відкрити кожен із них. В предметі (або курсі) в студента є можливість переглянути прикріплені вчителем файли, завантажити їх, є можливість

відвантажити готові роботи, переглянути оцінки. Інтерфейс застосунку можна побачити на рисунку 1.1.

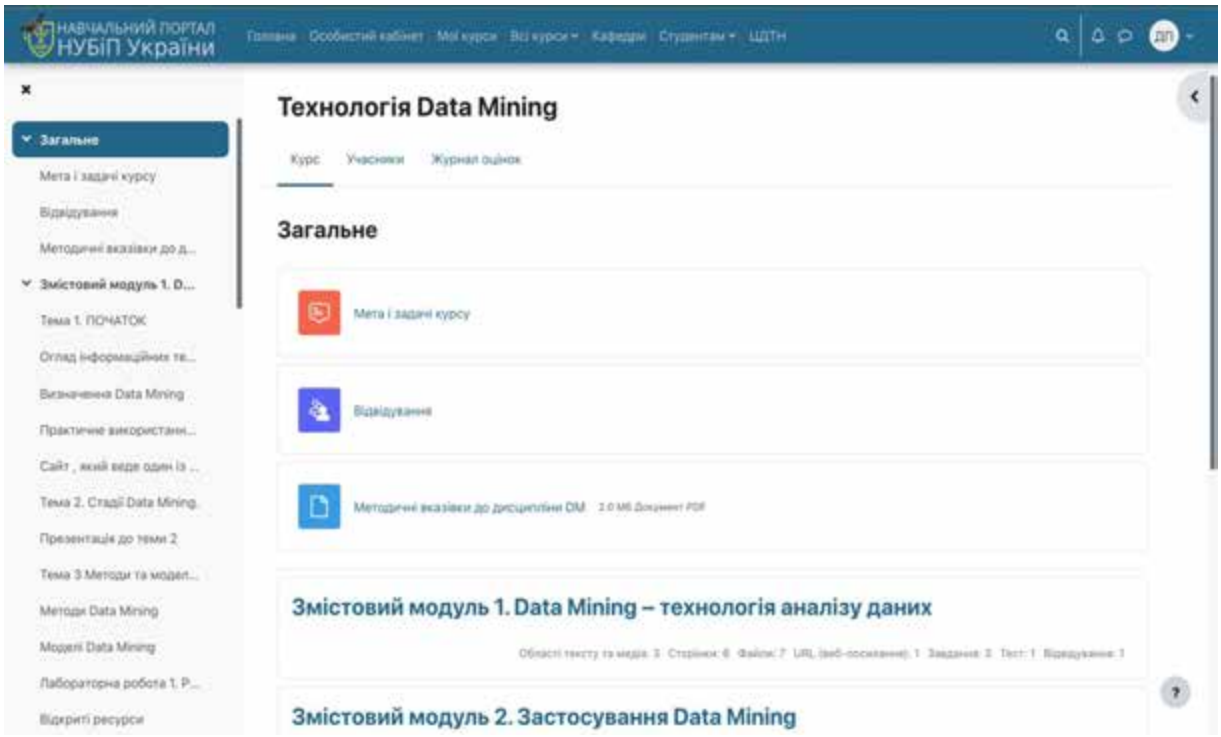

Рисунок 1.1 – Інтерфейс додатку Moodle для студента

Ось деякі переваги та недоліки платформи Moodle:

Преваги Moodle:

– Відкритий код: Moodle є відкритим програмним забезпеченням, що означає, що ви можете безкоштовно встановлювати, налаштовувати і редагувати його вихідний код. Це дає велику гнучкість і можливість адаптації до вашого навчального середовища.
– Захищеність і конфіденційність: Moodle надає можливість контролювати дані і доступ до них, що важливо для навчальних закладів, які ставлять насамперед конфіденційність.
– Розширені можливості навчання: Платформа підтримує багато типів контенту, включаючи тексти, відео, аудіо, тести, завдання та форуми для обговорення.

– Контроль та оцінка: Moodle надає інструменти для створення та оцінювання завдань, тестів і інших методів контролю знань.
– Спільна робота: Можливість взаємодії між вчителями та студентами через форуми, чати та інші інструменти для спільної роботи.
– Гнучкість в розгортанні: Ви можете встановити Moodle на власному сервері або використовувати хмарні хостингові рішення.
– Активна спільнота користувачів: Маючи велику глобальну спільноту користувачів, ви можете знайти підтримку, навчальний матеріал та розширення для Moodle.

Недоліки Moodle:

– Складність в налаштуванні: Налаштування Moodle може вимагати технічних навичок і ресурсів, що може бути складним для деяких користувачів.
– Брак інтеграції з іншими системами: Інтеграція з іншими програмами і системами може бути менш зручною, але це можливо завдяки розширенням.
– Великий обсяг функцій: Для новачків велика кількість можливостей і налаштувань може бути заплутаною.
– Не найкраща інтерфейс користувача: Деякі користувачі можуть вважати інтерфейс Moodle менш інтуїтивним і не таким сучасним, як у деяких інших LMS.
– Велика кількість спаму: В форумах може бути проблема зі спамом, особливо на публічних сайтах.

Загалом, Moodle є потужною платформою для навчання з великою кількістю можливостей, особливо для навчальних закладів та організацій, які можуть витратити час на налаштування і налагодження системи. Однак вона може бути складною для новачків та вимагати деякого рівня технічних знань.

1.2.2 Google Classroom

- це безкоштовна платформа для навчання онлайн, розроблена Google. Вона спрощує взаємодію між вчителями і студентами, надаючи інструменти для створення курсів, надсилання завдань, спільної роботи та оцінки студентів. На рисунку 1.2 можна побачити інтерфейс застосунку для учня.

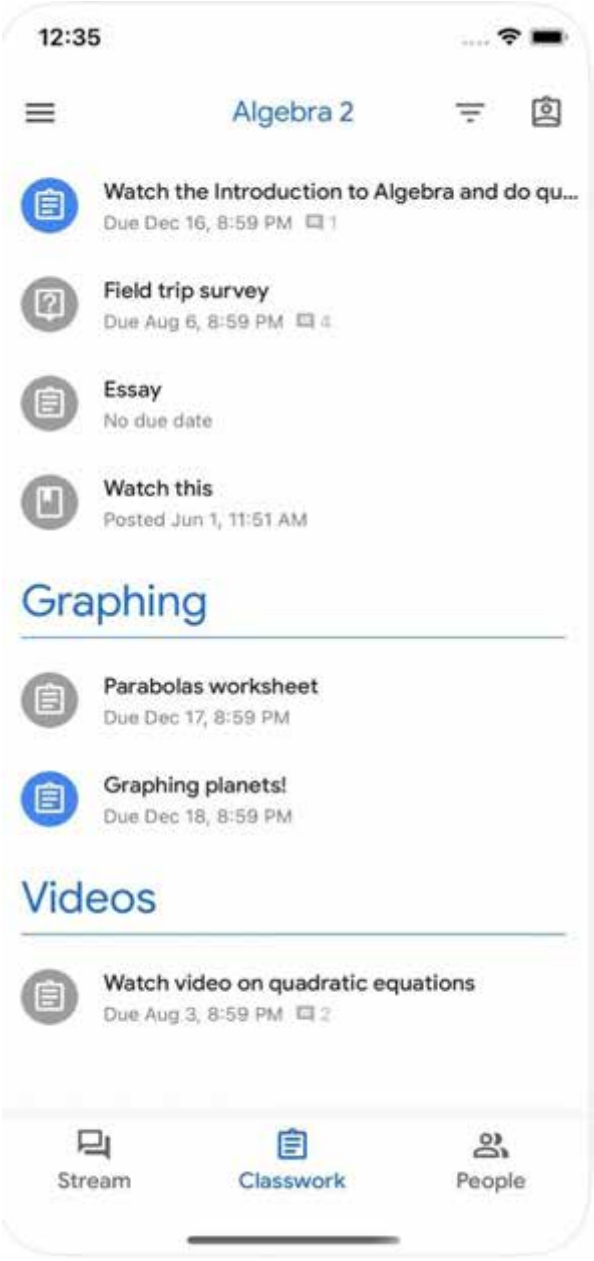

Рисунок 1.2 – Інтерфейс Google Classroom для учня

За останні роки ця платформа набула популярності, адже є зручною та інтуїтивною, не перевантаженою різним функціоналом. Для того, щоб почати навчання за допомогою цього інструменту достатньо мати Gmail електронну

пошту. Отримавши код доступу до курсу від викладача, студент може легко доєднатися до курсу. Розроблений мобільний застосунок, який спрощує взаємодію здобувача освіти із системою.

Ось деякі з переваг і недоліків Google Classroom:

Переваги Google Classroom:

- Безкоштовність: Google Classroom є безкоштовною для використання і доступною всім користувачам Google.
- Інтеграція з Google Apps: Платформа пов'язана з іншими продуктами Google, такими як Google Drive, Google Docs і Gmail, що дозволяє зручно спільно працювати з документами та матеріалами.
- Простота використання: Інтерфейс Google Classroom інтуїтивно зрозумілий і легко доступний навіть новачкам.
- Структуровані курси: Ви можете створювати курси, додавати уроки, завдання та ресурси для структурованого навчання.
- Зручність для вчителів: Вчителі можуть створювати завдання, надсилати їх студентам, встановлювати терміни виконання і оцінювати роботу студентів прямо в середовищі Classroom.
- Сповіщення та нагадування: Платформа надсилає сповіщення про оновлення курсів і завдань, що сприяє активній взаємодії.
- Можливість дистанційного навчання: Google Classroom ідеально підходить для дистанційного навчання і використовується багатьма учбовими закладами під час епідемічних обставин.

Недоліки Google Classroom:

- Обмежені можливості порівняно з іншими LMS: В порівнянні з іншими системами управління навчанням (LMS), Google Classroom може мати обмежені функціональні можливості.
- Залежність від Інтернету: Для користування Google Classroom потрібний постійний доступ до Інтернету, що може бути проблематичним для студентів і вчителів з обмеженим з'єднанням.

– Відсутність деяких функцій для корпоративного навчання: Для корпоративного навчання інші платформи можуть мати більше розширених функцій, які відсутні в Google Classroom.
– Орієнтована переважно на освітні установи: Google Classroom більше спрямована на школи і навчальні заклади, і вона може не задовольняти потреби корпоративного сектору.

Загалом, Google Classroom - це зручна та безкоштовна платформа для навчання, особливо для викладачів і студентів, які користуються Google Apps. Вона проста у використанні та сприяє ефективному навчанню, але вона може бути обмеженою для великих учбових закладів або корпорацій зі специфічними потребами.

ормування вимог до системи

Оскільки предметом розробки даної дипломної роботи є веб застосунок, який допомагає навчанню онлайн, то можна виділити основні категорії вимог, які має задовольняти даний продукт. Це вимоги до надійності, швидкодії, доступності та функціоналу.

Вимоги до надійності продукту:

– Захист від витоку даних: Для дистанційного навчання важливо забезпечити конфіденційність інформації студентів та вчителів. Сайт повинен мати надійні механізми захисту даних і уникати витоків особистої інформації.
– Резервне копіювання: Регулярне резервне копіювання даних допомагає відновити інформацію у випадку втрати або пошкодження даних.
– Захист від кібератак: Важливо мати механізми виявлення та запобігання кібератакам, включаючи DDoS-атаки і спроби вторгнення.

Вимоги до доступності сайту:

– Висока доступність: Сайт повинен бути доступним для користувачів у будь-який час, навіть при великому навантаженні або випадкових збоях.

Це може досягатися за допомогою резервування серверів та використанням CDN (мережі доставки контенту).

- Підтримка багатьох пристроїв: Сайт повинен коректно відображатися на різних типах пристроїв, включаючи комп'ютери, планшети та смартфони.
- Підтримка різних браузерів: Веб-сайт повинен бути сумісним з різними веб-браузерами, щоб забезпечити доступність для різних користувачів.
- Подання альтернативного контенту: Для зображень і мультимедіа-вмісту мають надаватися альтернативні текстові описи або інші засоби для розуміння контенту користувачами, які не можуть бачити або слухати.
- Підтримка екранних читачів: Сайт повинен бути сумісним з екранними читачами, що дозволяють людям з вадами зору отримувати інформацію з веб сайту.

Вимоги до швидкодії сайту:

- Швидка завантаження сторінок: Веб-сайт повинен завантажуватися швидко, навіть на повільних з'єднаннях, щоб користувачі могли миттєво отримувати доступ до вмісту.
- Кешування контенту: Використання кешування допомагає зменшити навантаження на сервер та прискорити завантаження сторінок для повторних відвідувачів.

Вимоги до функціоналу:

- Повинна бути система авторизації, для того, шоб люди могли зареєструватися чи ввійти в систему під своїм обліковим записом.
- Для учня має бути можливість переглянути доступні йому курси (предмети), подивитися розклад, свої оцінки, завантажити файли і відвантажити готові роботи.
- Для викладача має бути можливість додавати файли, бачити розклад, проставляти оцінки, редагувати інформацію про предмет, проставляти відвідуваність.

– Щодо адміністратора, то в нього повинні бути функції по менеджменту предметів, створенню груп, отриманні статистики за певними параметрами, додаванню користувачів.

остановка завдання

Проаналізувавши попередні підрозділи цього розділу, можна зробити висновок щодо завдання, яке потрібно виконати під час роботи над дипломом. Оскільки йдеться про систему для дистанційного навчання, то проаналізувавши подібні існуючі рішення і усвідомивши вимоги до такого роду системи, треба сформувати завдання.

Дивлячись на інші рішення, які описані в підрозділі 1.2, можна прийти до висновку, що не потрібно реалізовувати щось дуже масштабне по двох причинах. Перша причина – можна зробити систему важкою та із великою кількістю непотрібних функцій. Друга причина – моделювання та реалізація такої системи буде дуже довгим та ресурсоємним процесом, що виходить за межі дипломної магістерської роботи.

Тому потрібно взяти лише найкращі функції із розглянутих застосунків, описати та реалізувати їх, а також додати зверху свої додаткові функції, які будуть якісно відрізняти даний продукт від інших, та забезпечать неповторний досвід для користувачів. До таких функцій можна віднести можливість отримання сповіщень в якусь із популярних соціальних мереж, а також можливість отримати вичерпної статистики адміністратором.

Так як продукт маж бути надійним, швидким то потрібно використовувати популярні технології для розробки даного продукту. Серед таких технологій є React для створення сайту, який буде швидко працювати. Для серверної частини буде використовуватися ASP .NET. Ця технологій популярна та надійна. Для зберігання даних буде використовуватися БД сервер MS SQL, а для підвищення швидкодії – кеш Redis. Для контролю версій продукту буде використано Git та

Для такого продукту важливо також розробити правильну архітектуру, побудувати діаграми класів, сутностей БД, станів, прецедентів.

Тому підсумовуючи можна поставити такі основні завдання:

- Вивчити і розібратися із новими технологіями, які дозволять реалізувати дану систему.
- Правильно змоделювати систему, дотримуючись загальних принципів по побудові і використовуючи різного роду діаграми.
- Спроектувати БД для зберігання даних.
- Реалізувати задуману систему, користуючись вибраними раніше технологіями, та перевірити її на якість.
- Дати можливість адміністратору отримувати статистику в різних вимірах використовуючи сховище даних та OLAP-куб.

ОДЕЛЮВАННЯ СИСТЕМИ

азові принципи моделювання системи

Перед початком реалізації якогось продукту необхідно здійснити моделювання системи. В цей процес входить функціональне моделювання, моделювання архітектури, вибір технологій, моделювання компонентів.

Функціональне моделювання відбувається за допомогою різного роду UML-діаграм. Основною задачею є опис основних функцій системи, ролей, розкривання детальної інформації про деякі послідовності дій системи, опис стану системи та процесу переходу із одного стану в інший.

Побудова вимог до архітектури полягає в тому, щоб описати як має бути побудована програма, яких принципів дотримуватися, наприклад, SOLID. Також важливо описати компоненти системи, тобто складові частини, а також добре подумати над вибором технологій, за допомогою яких буде реалізована система.

оделювання архітектури

При розробці системи найважливіше є правильно сформулювати архітектурні вимоги, та вимоги до написання коду, адже від цього буде залежати розширюваність, швидкодія, захищеність системи. Архітектура програмного забезпечення системи представляє проектні рішення, пов’язані із загальною структурою та поведінкою системи. Архітектура допомагає зацікавленим сторонам зрозуміти й проаналізувати, як система досягне таких основних якостей, як можливість модифікації, доступність і безпека [4].

Можна виділити основні критерії хорошої архітектури: розширюваність, висока продуктивність, супровідність.

Розширюваність (Extensibility) є одним із ключових критеріїв хорошої архітектури програмного забезпечення. Цей принцип означає, що архітектура повинна бути спроектована так, щоб легко додавати новий функціонал або

змінювати існуючий без необхідності ревізії всієї системи. Це допомагає зберегти гнучкість та пристосовуваність програмного продукту до змінних вимог і умов.

Існує кілька архітектурних підходів та паттернів, які сприяють розширюваності програмних систем: модульна архітектура, паттерн розширення, плагінна архітектура, шарова архітектура. Для цього застосунку буде використовуватися шарова архітектура, яка відображена на рисунку 2.1.

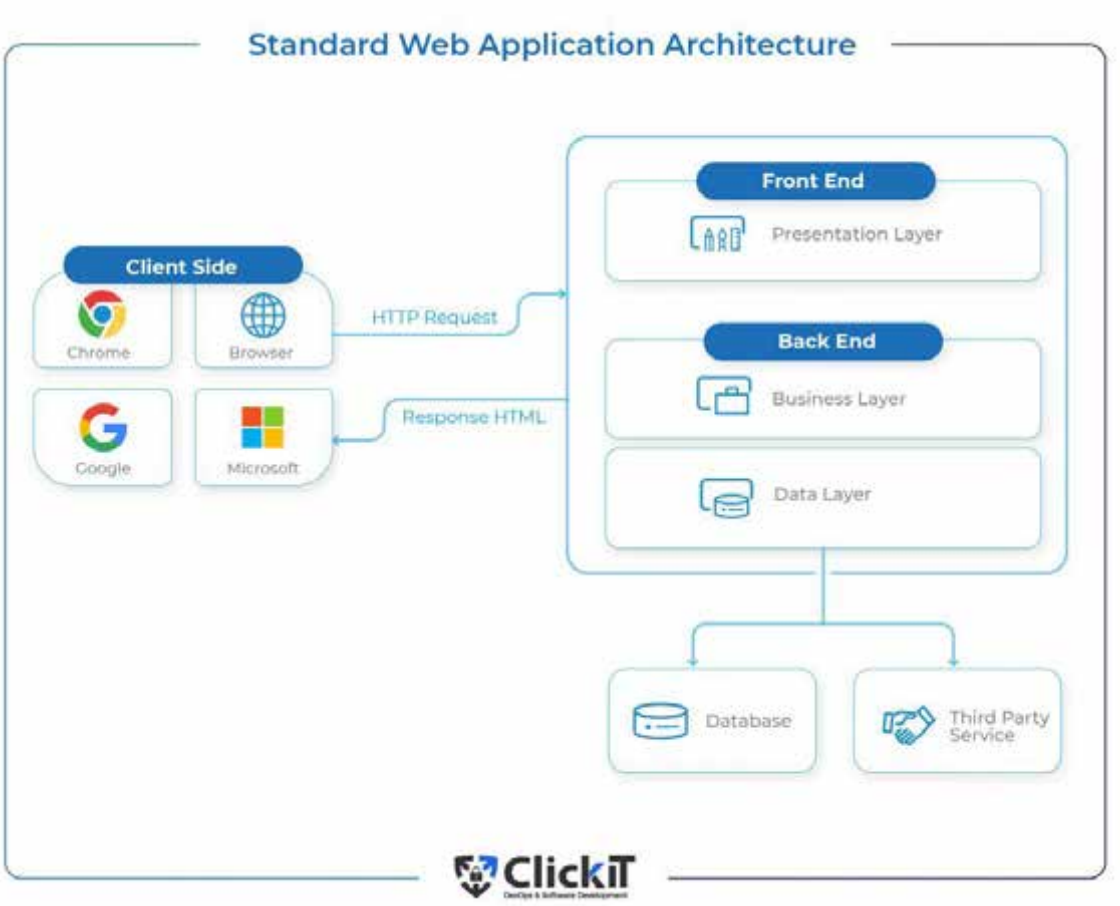

Рисунок 2.1 – Шарова архітектура [5]

Шарова архітектура (Layered Architecture), також відома як "архітектура у рівнях" або "архітектура великої розсіяної програми", є однією з загальноприйнятих архітектурних моделей для побудови програмного забезпечення. У цій архітектурі програма розділяється на окремі логічні "шари" або рівні, де кожен рівень відповідає за певний аспект функціоналу. Кожен рівень може взаємодіяти з рівнями нижче і вище, але інші рівні не повинні взаємодіяти

напряму один з одним. Ця модель сприяє структурованості, модульності і підтримці програмного продукту.

В даному випадку архітектура буде складатися із трьох рівнів:

- Клієнтський рівень (рівень презентації) – відповідає за відображення контенту для юзера, та є єдиним рівнем із яким користувач співпрацює напряму. Зазвичай це фронтова частина застосунку.
- Рівень бізнес логіки – відповідає за логіку обробки даних, які отримує від рівня доступу до даних, та перетворення цих даних в необхідну форму для передачі на рівень презентації.
- Рівень доступу до даних – відповідає за збереження даних. Туди відноситься БД та кеш.

Досягнення високої продуктивності (Performance) в програмних системах може бути досягнуто за допомогою вибору та оптимізації відповідної архітектури. В даному випадку для цього буде застосовано кешування. Кешування – це використання кешів для зберігання та швидкого доступу до часто використовуваних даних. Кешування може бути реалізоване на різних рівнях архітектури, включаючи рівень додатку, рівень сервера та рівень бази даних.

Супровідність (Maintainability) в архітектурі програмного забезпечення означає, наскільки легко систему можна підтримувати та розвивати з часом. Для досягнення високої супровідності важливо вибирати та розробляти архітектуру, яка полегшує такі аспекти: Читабельність і структурованість коду, модульність, документація, розділення відповідальностей.

Для дотримання цих вимог ввели такі поняття як зв’язність та зв’язаність

Зв'язність відноситься до того, наскільки функції або об'єкти в межах одного модулю (класу, функції тощо) пов'язані та спрямовані на виконання спільного завдання. Висока зв'язність вказує на те, що всі частини коду в модулі служать одному конкретному завданню, і вони добре узгоджені між собою. Це сприяє читабельності та обслуговуваності коду. Висока зв'язність зазвичай є

бажаною, оскільки вона допомагає підтримувати код, який є легким для розуміння, тестування та модифікації.

Зв'язаність відноситься до того, наскільки один модуль або компонент програми залежить від іншого модулю або компонента. Висока зв'язаність вказує на те, що модулі дуже сильно взаємозалежні, і зміни в одному модулі можуть вплинути на інші модулі. Низька зв'язаність означає, що модулі слабо залежать один від одного, і зміни в одному модулі мають обмежений вплив на інші. Низька зв'язаність зазвичай є бажаною, оскільки вона дозволяє більше незалежності та полегшує модифікацію та розширення програмного забезпечення [6].

На рисунку 2.2 зображено низька зв’язність та висока зв’язаність.

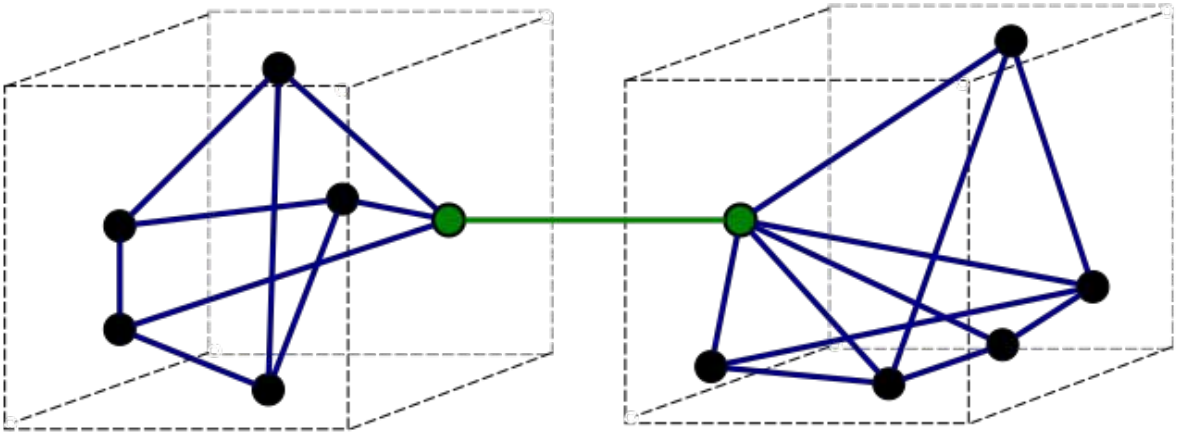

Рисунок 2.2 - Низька зв’язність та висока зв’язаність

На рисунку 2.3 зображено низьку зв’язаність та високу зв’язність, що не є хорошим показником.

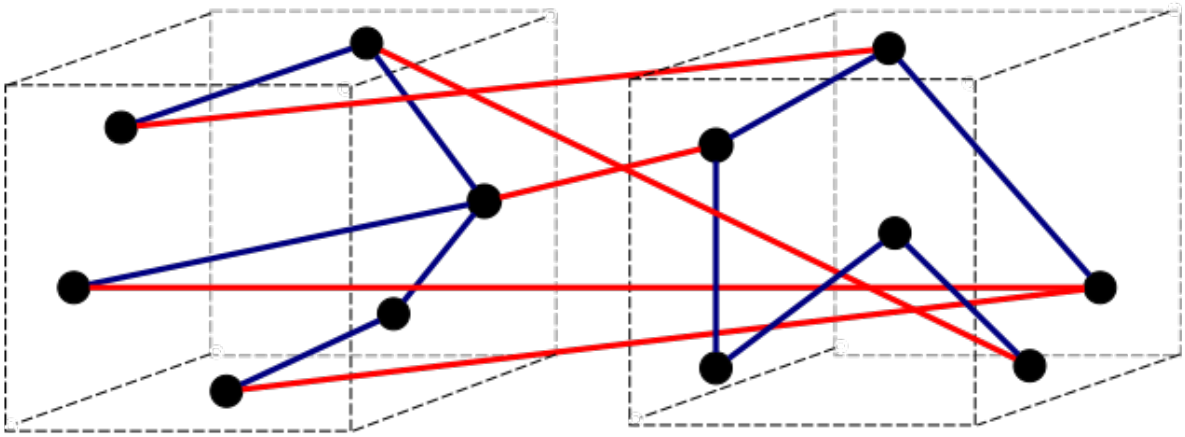

Рисунок 2.3 – Низька зв’язаність та висока зв’язність

Не можна не згадати про чудовий набір принципів програмування, які мають назву SOLID. Принципи SOLID - це набір п'яти основних принципів, які допомагають створювати якісне та легко підтримуване програмне забезпечення. Кожна літера в слові "SOLID" представляє окремий принцип. Ось короткий опис деяких із них:

– Принцип єдиної відповідальності (Single Responsibility Principle, SRP). Цей принцип стверджує, що кожен клас або модуль повинен мати лише одну причину для зміни. Іншими словами, клас повинен виконувати лише одну конкретну функцію або завдання. Якщо клас має більше однієї причини для зміни, це може призвести до складності та проблем при підтримці.

– Принцип відкритості для розширення, закритості для змін (Open/Closed відкритим для розширення новим функціоналом, але закритим для змін. Це досягається за допомогою використання абстракцій та інтерфейсів, що дозволяють додавати новий функціонал без зміни вже існуючого коду.

Використання принципів SOLID допомагає підвищити якість програмного забезпечення, зменшити складність та полегшити його підтримку та розширення. Вони є важливими в розробці програмного забезпечення, особливо у великих проектах та в командній розробці [7].

ибір платформи застосунку

При виборі платформи для розробки застосунку, слід врахувати різні фактори, включаючи цільову аудиторію, вид застосунку, ресурси та технічні вимоги. Ось декілька варіантів платформ для розгляду: мобільна платформа, десктопна платформа, веб платформа.

Основні мобільні платформи для розробки додатків – це Android і iOS. Якщо вибрати платформу суто для Android, то для цього можна використовувати мову програмування Java або Kotlin. Розробка для iOS зазвичай використовує мову програмування Swift або Objective-C. Але написання двох окремих

застосунків дуже ресурсоємне, тому є можливість написати крос-платформенний мобільний застосунок, який буде компілюватися в застосунки і для Android, і для iOS. Для цього підійде один із крос-платформенних фреймворків React Native, Flutter або Xamarin.

Щодо десктопних платформ, то тут є три чотири варіанти: Windows, macOS, Linux або крос-платформенний застосунок. Оскільки писати 3 окремі застосунки є нераціонально, то можна розглянути крос-платформенний застосунок. Його можна реалізувати за допомогою фреймворку Electron.

І останній варіант, це розробка веб застосунку. Тут можна використати чимало різних фреймворків: React, Angular, NextJS. Плюсом є те, що додаток можна буде запустити з будь якого девайсу.

В таблиці 2.1 показано порівняння трьох платформ.

Таблиця 2.1 – Порівняння платформ.

	Mobile	Desktop	Web
Підтримується всіма популярними пристроями			
Платформа популярна, актуальна, буде ще довго на ринку			
Платформа має обмеження, які не дозволять реалізувати певний функціонал			

Із таблиці видно, що більше плюсів має саме веб застосунок, тому, що може бути запущений на будь-якому пристрої. Оскільки проект має великий функціонал, і повинен бути зручним то доводиться відмовитися від мобільної платформи, оскільки для адміністратора буде не зручно керувати таким застосунком із малого екрану телефону. Так само відпадає десктоп платформа,

оскільки студентам часто зручніше скористатися телефоном, щоб взяти якусь інформацію, а не тягати із собою всюди ноутбук.

сновні принципи при створенні дизайну

При створенні дизайну веб-сайту важливо орієнтуватися на кілька ключових аспектів, які допоможуть зробити ваш сайт привабливим та ефективним. Ось деякі важливі фактори, на які варто звернути увагу:

Цільова аудиторія. Розуміння аудиторії сайту – це ключ до успіху. Потрібно визначити, хто буде використовувати сайт і які їхні потреби та очікування. Дизайн має відповідати потребам цільової аудиторії. Оскільки цільовою аудиторією є здобувачі освіти та викладачі, то дизайн має бути лаконічним, без лишнього пафосу.

Інтуїтивність та навігація. Потрібно забезпечити легкий інтуїтивний спосіб навігації по сайту. Меню, кнопки та посилання мають бути легкими для розуміння та використання. Для мобільної версії треба використовувати бургер меню для навігації, а для десктопа – хедер.

Мінімалізм. Мінімалістичний дизайн дозволяє зосередити увагу на головних елементах, уникаючи перевантаженого дизайну та надмірного використання кольорів та графіки.

Респонсивність. Треба забезпечити, щоб сайт був адаптивним до різних пристроїв та екранних розмірів. Це важливо для зручності користувачів, які відвідують сайт на мобільних пристроях.

Кольорова палітра. Було вибрано кольорову палітру, яка відповідає бренду та атмосфері сайту для організації дистанційного. Кольори можуть впливати на емоційний стан користувачів, тому були підібрані спокійні кольори. На рисунку 2.4 можна побачити палітру для сайту.

Рисунок 2.4 – Палітра для сайту

Завантаження сторінки. При завантаженні сторінки потрібно додати анімації завантаження, для того, щоб користувач розумів, що сайт не завис.

Типографіка. Вибір правильного шрифту і розміру шрифту важливий для читабельності та естетичного вигляду. Тому треба використовувати шрифти, які легко читаються на екрані. Було вибрано сімейство шрифтів Volkorn. На рисунку 2.5 можна побачити підібрані шрифти для сайту.

Рисунок 2.5 – Шрифти для сайту

Контент. Потрібно розміщувати контент легко для розуміння і доступу. Важливу інформацію розміщують заздалегідь, а надлишковий текст доречно приховати за допомогою випадаючих вікон.

Взаємодія. Треба розглянути, як користувачі будуть взаємодіяти з сайтом і додати елементи взаємодії, такі як форми, коментарі, кнопки тощо.

Загалом, ефективний дизайн сайту має поєднувати естетику з функціональністю та враховувати потреби користувачів та бізнес-цілей [8].

ункціональне моделювання

Функціональне моделювання – це процес створення моделі програмного забезпечення, яка відображає функції та функціональні вимоги до системи. Цей вид моделювання допомагає зрозуміти, як програма буде виконувати певні завдання, які функції вона повинна виконувати та як вони будуть взаємодіяти.

Основні аспекти функціонального моделювання включають наступне:

- Опис функцій - визначення і опис функцій, які повинні бути реалізовані в програмному забезпеченні. Це може включати усі можливі функції, які система повинна виконувати, від основних до додаткових.
- Взаємодія між функціями – визначення, як функції взаємодіють одна з одною та як вони обмінюються даними. Це допомагає зрозуміти порядок виконання функцій та послідовність подій.
- Функціональні вимоги – формалізація функціональних вимог до системи. Це включає у себе створення специфікацій, які описують, як має виконуватися кожна функція та які вимоги до результатів роботи функцій.
- Діаграми та графи – використання графічних інструментів, таких як блок-схеми, діаграми потоку даних, діаграми класів тощо, для візуалізації функціональної моделі.
- Тестування та верифікація – функціональна модель може бути використана для розробки тестових випробувань та перевірки, як добре програма відповідає функціональним вимогам.

Функціональне моделювання допомагає командам розробників та аналітикам розібратися в тому, як система повинна працювати, і забезпечує основу для подальшого проектування та реалізації програмного забезпечення. Це особливо важливо на початкових етапах розробки для уникнення недорозумінь і помилок при реалізації функцій системи [9].

іаграма прецедентів

Діаграма прецедентів – це графічний інструмент у рамках моделювання вимог в системному аналізі та проектуванні програмних систем (UML). Ця діаграма допомагає визначити та візуалізувати функціональні вимоги до системи з точки зору їх взаємодії з різними акторами (користувачами чи іншими системами).

Основні компоненти діаграми прецедентів включають наступне: актори, прецеденти та відношення між акторами та прецедентами.

Актори – це суб'єкти або користувачі, які взаємодіють з системою. Актори можуть бути людьми, іншими системами або навіть іншими компонентами системи. Кожен актор зображується у вигляді піктограми, такої як іконка людини.

Прецеденти – це конкретні дії або функції, які виконує система відповідно до запитів акторів. Кожен прецедент описується коротким ім'ям або описом. Вони зображуються у вигляді овалів. Одна із послідовностей дій системи спрямована на виконання головної цільової функції даного прецеденту і визначається, як основний потік подій (сценарій). Альтернативні потоки описують поведінку системи у виняткових ситуаціях, наприклад, при помилках [10].

Відношення між акторами та прецедентами: Стрілки, що з'єднують акторів і прецеденти, показують взаємодію. Існують два типи відношень:

– Відношення "викликає": Показує, що актор викликає певний прецедент.
– Відношення "успадковує": Використовується, коли один прецедент успадковує функціонал іншого.

Діаграма прецедентів використовується для уточнення вимог до системи, ідентифікації функціональних можливостей та потреб акторів, опису ролей акторів та визначення їх взаємодії з системою, визначення імен і описів прецедентів.

Діаграми прецедентів є потужним інструментом для аналізу та документування вимог до програмних систем і допомагають створити зрозумілу картину функціональності системи з точки зору користувачів [10].

На рисунку 2.6 можна побачити діаграму прецедентів, яка описує основні функції застосунку для трьох ролей.

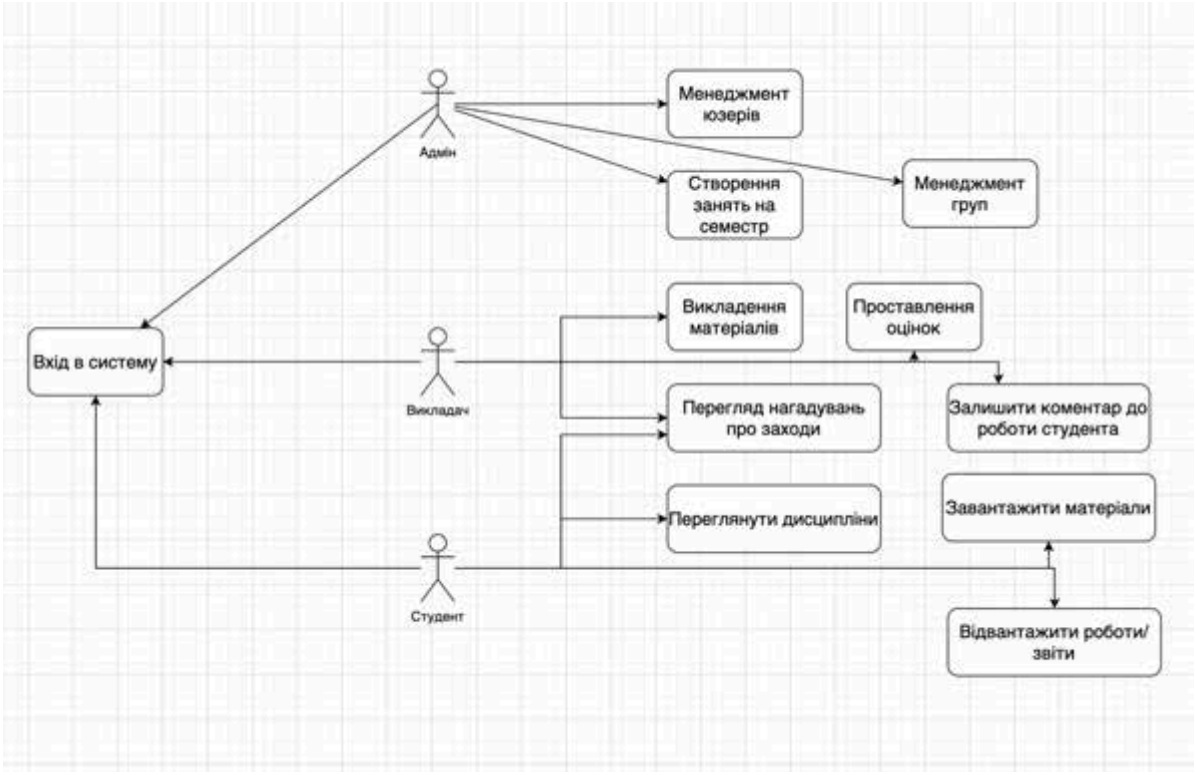

Рисунок 2.6 – Діаграма прецедентів

агальний опис ролей в системі

Дивлячись на діаграму прецедентів на рисунку 2.6, можна зробити висновок, що в системі будуть виділені три ролі для різних класів користувачів. Кожна роль матиме свій унікальний функціонал, але деякі можливості будуть спільними для всіх ролей, такі як авторизація в додатку. Нижче наведено короткий опис кожної з виділених ролей.

Адміністратор:

- Має можливість управляти користувачами і їх ролями.
- Відповідає за управління сутностями в системі.
- Має функціонал для керування всією системою.

Викладач:

- Може оцінювати учнів і редагувати заняття.

– Має доступ до функціоналу, пов'язаного з навчанням і оцінкою студентів.

Студент:

– Має можливість переглядати свої оцінки, заняття та завантажувати готові роботи.
– Отримує сповіщення та інформацію, яка стосується його навчання.

Розподілення по ролях важливе для ефективного управління доступом та безпекою системи. Адміністратор має вищий рівень доступу та може керувати іншими користувачами, тоді як викладачі та студенти мають обмежений функціонал відповідно до їхніх потреб. Ця розподілена система дозволяє кожній ролі виконувати свої функції, не втручаючись в роботу інших ролей та забезпечує відповідність правилам безпеки та конфіденційності.

пис можливостей адміністратора

Розроблена система має різні ролі користувачів, але найвищий рівень доступу має адміністратор. Він відповідає за керування користувачами, групами та дисциплінами. Для кращого розуміння, ось приклади операцій, які може виконувати адміністратор:

– Керування користувачами:
    - Додавання нових користувачів до системи.
    - Видалення користувачів, наприклад, коли викладач звільняється.
    - Редагування профілів користувачів та надання їм відповідних ролей.
    - Додавання нових користувачів до системи, таких як студенти, викладачі тощо.
    - Заповнення обов'язкових полів, таких як електронна пошта, ПІБ, дата народження та контактний номер телефону.

- Призначення початкового пароля для користувача, який потребує зміни при першому вході.

– Менеджмент груп:
  - Створення нових груп на початку навчального року.
  - Визначення назви групи та призначення її до певної кафедри та факультету.
  - Вибір року навчання для створеної групи.
  - Можливість додати студентів до певної групи.
  - Вибір студентів зі списку та їхнє додавання до обраної групи.
– Керування дисциплінами:
  - Створення та редагування дисциплін.
  - Визначення для кожної дисципліни типу занять (лекції, практики, лабораторні).
  - Призначення викладачів для кожного типу занять.
  - Створення розкладу для кожної дисципліни.

Адміністратор має повний доступ до всіх операцій та може керувати всіма аспектами системи. Це дозволяє забезпечити ефективне управління та підтримку системи з високим рівнем контролю.

пис можливостей викладача

Для викладача передбачено функції та можливості, пов'язані з оцінюванням студентів та навчанням матеріалу. Ось опис доступних можливостей для викладача:

– Перегляд предметів:
  - Викладач має доступ до списку предметів, які були призначені йому адміністратором.
– Графік занять:

- Для кожного предмета викладач може переглядати графік занять, включаючи дні та час проведення.

– Журнал оцінок та відвідуваності:
  - Викладач може відкривати журнал для кожного предмета та ставити оцінки студентам.
  - Він також може позначати відвідуваність студентів на кожному занятті.
– Контрольні та практичні роботи:
  - Викладач може призначити контрольні або практичні роботи на кожне заняття та оцінити їх.
  - У журналі підраховується сума всіх оцінок для кожного студента.
– Матеріали для предмету:
  - Викладач може додавати матеріали для кожного предмету, такі як зображення, презентації, відео, документи тощо.
  - Для завантаження файлу використовується кнопка "Завантажити матеріал".
– Статистика завантажень:
  - Викладач може переглядати статистику по скачуваннях матеріалів, включаючи інформацію про те, які студенти завантажили конкретний файл.
– Нагадування в Telegram:
  - Викладач може надсилати нагадування студентам через Telegram про контрольні роботи або перевірку завдань.
– Посилання на відеоконференції:
  - Викладач може додавати посилання на відеоконференції для проведення онлайн-занять.

Ці функції допоможуть викладачу ефективно виконувати свої обов'язки щодо оцінювання студентів і надання матеріалів для навчання.

пис можливостей студента

Для студента передбачено доступ до функціоналу, необхідного для отримання інформації та матеріалів занять. Ось опис можливостей для студента:

- Розклад занять:
  - Студент має доступ до розкладу занять, де відображені всі предмети, які він повинен відвідувати.
  - Можливість переглядати дати, час та місце проведення занять.
- Успішність та контрольні роботи:
  - Студент може перевірити свою успішність на кожному предметі, включаючи оцінки та розклад контрольних робіт.
- Відвідуваність:
  - Можливість перевірити власну відвідуваність занять.
- Матеріали для навчання:
  - Доступ до матеріалів, які надаються викладачем, таких як презентації, зображення, відео тощо.
  - Завантаження матеріалів для власного використання.
- Сповіщення в Telegram:
  - Студент може отримувати сповіщення через Telegram щодо змін у розкладі, контрольних робіт тощо, якщо він вказав свій номер телефону.
- Завантаження робіт:
  - Можливість завантажити власні роботи для перевірки, такі як лабораторні роботи або завдання на оцінку.
  - Одиночний файл або архів, який містить кілька файлів, можуть бути завантажені у форматах pdf, img, png, doc, docx, xlsx або

- Список завдань:
  - Перегляд списку завдань, які студент повинен виконати, включаючи дату здачі, максимальний бал та оцінку після перевірки.

Цей функціонал допоможе студентам ефективно взаємодіяти з системою та отримувати необхідну інформацію для навчання.

іаграма стану

Діаграма стану, також відома як діаграма автомата або state chart diagram, використовується для візуалізації різних станів та переходів між ними в системі або процесі. Вона є частиною мови моделювання UML (Unified Modeling Language) і допомагає розуміти, як система реагує на різні події та умови.

Діаграма стану складається з таких елементів:

- Стан (State): Представляє конкретний стан системи або об'єкта. Наприклад, стан "Включено", "Вимкнено", "Очікування", тощо. Позначається прямокутником.
- Перехід (Transition): Відображає переміщення системи або об'єкта із одного стану в інший відповідно до певних подій чи умов. Переходи позначаються стрілками.
- Подія (Event): Події, які спричиняють переходи між станами. Наприклад, "Натиснути кнопку", "Отримати сигнал", "Закінчити операцію".
- Умова (Condition): Умови, що пов'язані з переходами. Вони вказуються поруч із стрілкою переходу та регулюють, за яких умов перехід відбувається. Позначається ромбом.
- Початковий стан (Initial State): Вказує початковий стан системи або об'єкта. Позначається кругом.
- Завершальний стан (Final State): Позначає закінчення системи або об'єкта. Позначається колом і кругом всередині нього.

Діаграма стану дозволяє моделювати різні сценарії поведінки системи і визначити, як вона реагує на різні події. Вона особливо корисна для моделювання складних процесів та систем, де стани та їх переходи грають важливу роль [11].

В цій проектній роботі є функціонал по зміні паролю для юзерів. Цей процес описаний за допомогою діаграми станів на рисунку 2.6.

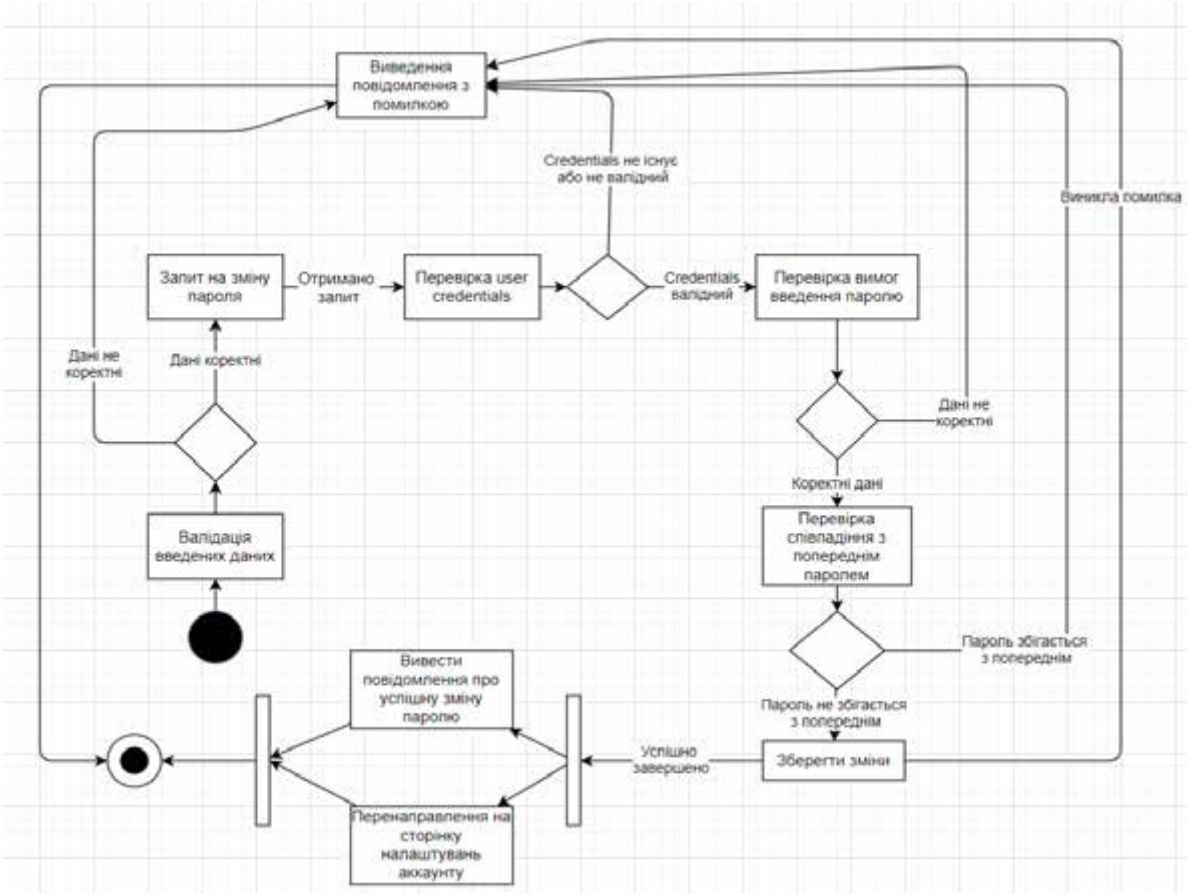

Рисунок 2.7 – Процес зміни паролю, описаний за допомогою діаграми станів

Зміна паролю - це важлива функція для системи. Перш за все, користувач отримує свій початковий пароль від адміністратора, який може бути згенерований автоматично або встановлений вручну. Після першого входу в систему користувачеві рекомендується змінити свій пароль. Для цього він може перейти на сторінку "Особистий кабінет".

Для зміни паролю користувачу потрібно ввести свій старий пароль і встановити новий пароль, після чого повторно введення нового паролю для

підтвердження. Після введення цих даних користувач повинен натиснути кнопку "Змінити". Якщо всі дані введені правильно і пройшли валідацію, користувач отримає повідомлення про успішну зміну паролю і буде перенаправлений на сторінку авторизації. У випадку невірного введення даних, користувач отримає відповідне повідомлення про помилку.

Процес зміни паролю включає наступні стани:

- Валідація введених даних.
- Виведення повідомлення про помилку, якщо дані не валідні.
- Запит на зміну паролю, відправлений на API-сервіс.
- Перевірка даних користувача на API-сервісі.
- Виведення повідомлення про помилку перевірки даних користувача, якщо вони не відповідають умовам.

Ця функція допомагає користувачам забезпечити безпеку свого облікового запису та змінювати пароль за потреби.

іаграма послідовності

Діаграма послідовності, також відома як Sequence Diagram, це один із типів діаграм UML (Unified Modeling Language). Ця діаграма використовується для візуалізації імперативних повідомлень, які передаються між різними об'єктами або компонентами в системі в певній послідовності.

Діаграми послідовності часто використовуються для моделювання взаємодій між об'єктами або компонентами під час виконання конкретного сценарію чи операції. Вони малюють послідовність повідомлень, які передаються між об'єктами (або компонентами) у вигляді стрілок і ліній життєвого циклу. Ця діаграма допомагає візуалізувати, які дії виконуються в системі та в якому порядку.

Діаграми послідовності особливо корисні при аналізі та проектуванні систем, де важливо розуміти порядок виконання дій між різними об'єктами чи

компонентами. Вони допомагають комунікувати та розуміти логіку взаємодій в системі.

Ця діаграма може бути корисною при створенні детальних моделей послідовності конкретних операцій або сценаріїв у вашій системі [12].

Під час розробки застосунку діаграми послідовностей часто ставали в пригоді. Прикладом може слугувати діаграма процесу авторизації в систему, відображена на рисунку 2.8.

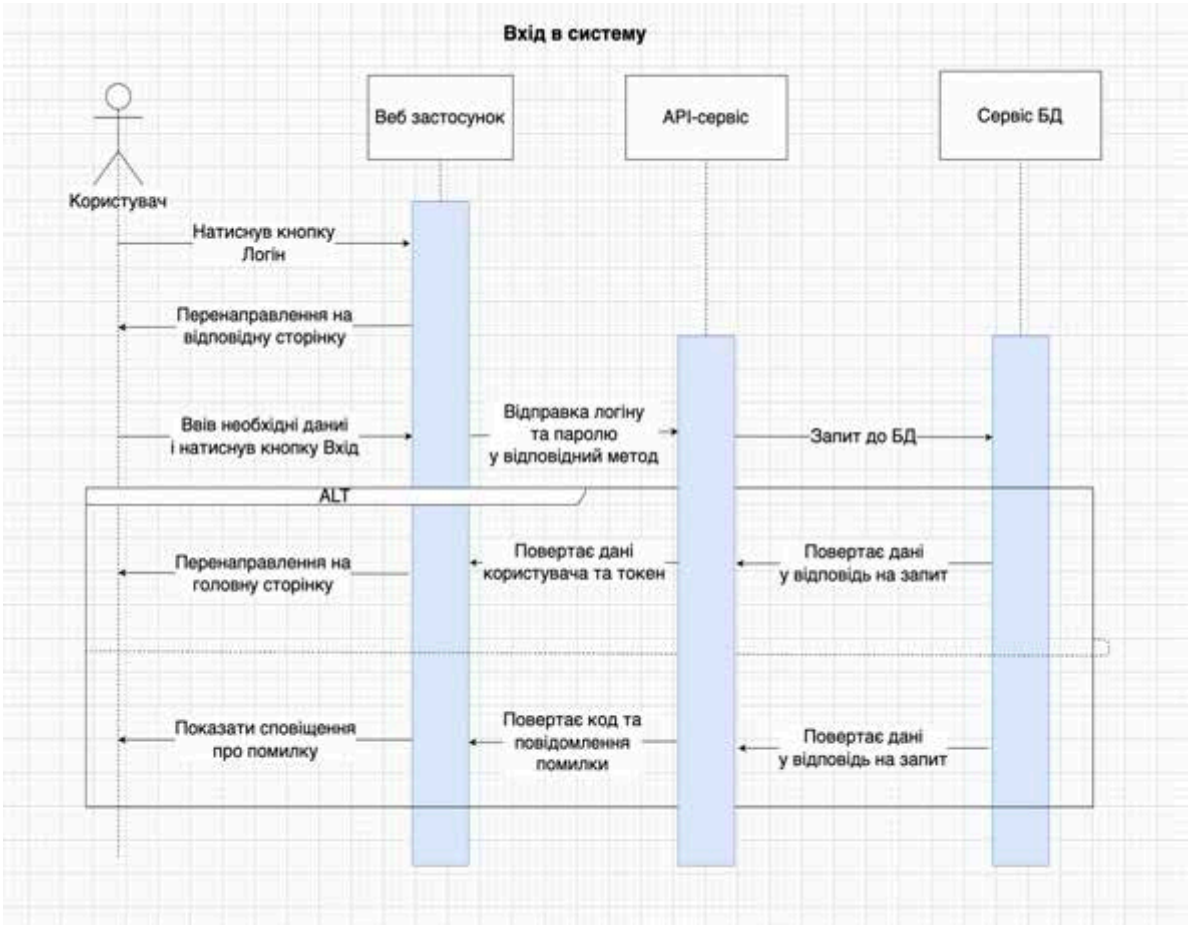

Рисунок 2.8 – Діаграма послідовності – вхід в систему

На схемі позначені чотири об'єкти: користувач, веб застосунок, API-сервіс та сервіс БД. У початковий момент, користувач активує кнопку «Логін», яка знаходиться у хедері веб застосунку. Після цього, управління переходить до веб застосунку.

Веб застосунок виконує перехід на сторінку авторизації, де розташована форма із полями «Email» та «Пароль». Користувач заповнює ці поля та натискає

кнопку «Вхід». Тоді управління знову передається веб застосунку. Веб застосунок аналізує ці дані і відправляє POST-запит до API-сервісу із відповідними даними для авторизації.

Протягом цього часу, управління переходить до API-сервісу, який приймає вхідні дані від веб застосунку та проводить перевірку на їхню вірність. З погляду безпеки, паролі в БД не зберігаються у відкритому вигляді, але замість цього, вони хешуються за допомогою спеціальної функції. Тому API-сервіс спершу хешує отриманий пароль і отримує результат у вигляді, який зберігається в БД. Далі API-сервіс формує запит до БД для вибору користувача, у якого email відповідає email, отриманому у запиті, а пароль – хешованому паролю.

Після цього управління передається серверу бази даних. Тоді можливі два розвитки подій. У першому сценарії сервер бази даних знаходить користувача в таблиці "Users" за вказаними умовами і відправляє відповідь із відповідними даними. Далі API-сервіс генерує JWT-токен, який надсилає відповідь веб застосунку, включаючи інформацію про користувача.

Потім веб застосунок зберігає отримані дані в сховище. Збережений JWT-токен буде додаватися до кожного наступного запиту до API, а ім'я користувача буде відображатися у заголовку веб застосунку. На закінчення веб застосунок перенаправляє користувача на головну сторінку.

У другому сценарії сервер бази даних не знаходить жодного запису, який відповідає вказаним умовам, і повертає порожній результат. API-сервіс приймає цю відповідь і передає веб застосунку повідомлення про помилку відповідно до запиту. Коли управління повертається веб застосунку, додаток обробляє отриманий результат і виводить повідомлення про помилку у вікні сповіщення.

Отже, Sequence Diagrams є корисним інструментом для розробки та розуміння послідовностей подій і взаємодій між об'єктами та компонентами системи. Вони надають чітку візуальну репрезентацію того, як об'єкти та компоненти системи взаємодіють один з одним в конкретний момент часу,

допомагаючи розробникам краще розуміти послідовність операцій та логіку системи.

Sequence Diagrams можуть виявити можливі конфлікти або помилки в послідовності операцій і відслідковувати їх виправлення. Вони служать ефективним інструментом комунікації між членами розробницької команди і стають частиною документації проекту, полегшуючи розуміння системи.

Sequence Diagrams також допомагають відслідковувати послідовність виконання функцій, виявляти можливі пункти оптимізації та виправлення помилок у системі. Завдяки їм розробники можуть краще аналізувати та оптимізувати взаємодії в системі, забезпечуючи її надійність та ефективність.

роектування компонентів системи

омпоненти системи

Діаграма компонентів є одним із видів діаграм UML (Unified Modeling Language), який використовується для моделювання структури системи та її компонентів. Ця діаграма надає візуальне подання компонентів, їх зв’язків та взаємодій у системі. Вона корисна для архітектурного проектування, описування розподіленої системи та розробки програмного забезпечення.

Основні складові діаграми компонентів:

- Компоненти – це основні будівельні блоки системи, такі як модулі, бібліотеки, класи, сервіси, додатки тощо. Кожен компонент має ім'я та інтерфейс, який визначає методи та властивості, доступні для інших компонентів.
- Зв’язки. Діаграма компонентів показує зв'язки між компонентами. Зв'язки вказують, як один компонент може використовувати або залежати від іншого. Зв'язки можуть бути напрямними (вказують на однобічну залежність) або двобічними (вказують на взаємну залежність).

– Інтерфейси. Вони визначають, які методи та властивості доступні для інших компонентів. Вони надають абстрактний спосіб взаємодії з компонентом і визначають контракти, які інші компоненти повинні дотримуватися.
– Профілі компонентів. Профілі дозволяють додатково описати компоненти за допомогою метаданих. Це може включати інформацію про розміщення, версію, мову програмування тощо.

Діаграма компонентів корисна для визначення архітектури системи та розподілу компонентів, аналізу залежностей між компонентами та їх взаємодій. Також її використовують для документування структури системи для розробників та стейкхолдерів і виявлення можливостей для вдосконалення та оптимізації архітектури.

На рисунку 2.9 можна побачити діаграму компонентів для даного застосунку.

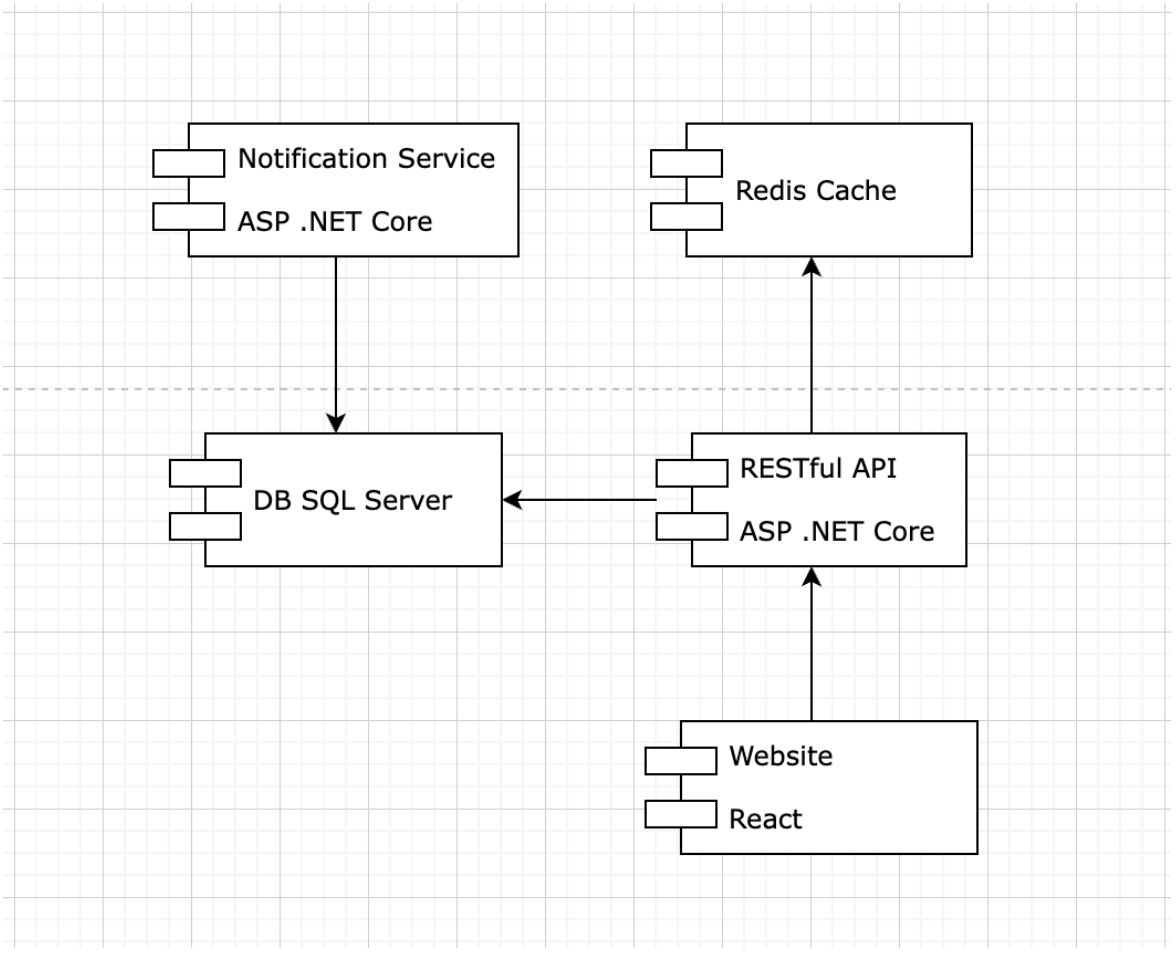

Рисунок 2.9 – Діаграма компонентів

Система складається із п’яти компонентів: Website – фронт-частина системи, написана на фреймворку ReactJS, API – серверна частина системи, яка відповідає за всю бізнес логіка та внутрішню роботи програми, DB Server – це база даних, де зберігаються всі дані по системі, такі як юзери, предмети, бали і т. д., використовується MS SQL Server, Redis Cache – сервісна частина яка відповідає за оптимізацію швидкодії та Notification Service – це сервіс, який співпрацює з БД і буде відправляти повідомлення користувачам, написаний на

Розберемо більш детально кожен компонент системи, та опишемо процес вибору технології для цього компоненту.

ибір технологій для компонентів

Вибір технологій для розробки сайту є критичним завданням, і від нього залежить успішність проекту. Про детальний вибір технологій буде розписано нижче, але є основні вимоги та критерії, які слід враховувати при виборі кожної з технологій для кожного з компонентів

- Стабільність технології. Треба вибирати технології, які мають стабільну і відому історію розвитку. Технології, які постійно оновлюються або перебувають в стані бета-версій, можуть бути менш надійними.
- Спільнота та підтримка. Важливо переконатися, що вибрана технологія має активну спільноту користувачів і розробників, а також регулярну підтримку і оновлення. Це допоможе вирішувати можливі проблеми та отримувати підтримку у випадку аварій.

- Безпека. Треба зверніть увагу на безпеку технології і вибирати технології, які мають відому історію щодо виправлення безпекових дірок і загроз.
- Масштабованість. Вибір технології повинен допускати масштабованість вашого сайту. Технологія має підтримувати горизонтальне та вертикальне масштабування.
- Тестування та якість коду. Технології з розвинутою екосистемою для тестування (наприклад, фреймворки для тестування) можуть сприяти надійності сайту.

Вибір технологій повинен бути обґрунтованим і враховувати потреби та завдання конкретного проекту. Також варто пам'ятати про можливість майбутнього розширення та підтримки сайту на обраній технології.

ибір технологій для фронт частини

Для Website компоненту був вибір між трьома фреймворками: Angular, React та Vue. Нижче буде наведено переваги та недоліки цих трьох фреймворків.

Переваги React:

- Віртуальний DOM: React використовує віртуальний DOM для оптимізації оновлення сторінки, що забезпечує високу швидкодію.
- Широкий вибір сторонніх бібліотек та компонентів: Велика кількість готових бібліотек та компонентів для React дозволяють розширювати функціонал вашого додатку.
- Спрощена структура проекту: React не нав'язує конкретної структури проекту, що дозволяє розробникам вибирати підходящий для них спосіб організації коду.

Недоліки React:

- Необхідність вибору додаткових бібліотек: React не містить всі функції "з коробки", і для певних завдань потрібно вибирати та налаштовувати сторонні бібліотеки.

- Брак вбудованого маршрутизатора та стану: React не має вбудованого роутера та стану, тому для їх реалізації може бути потрібно використовувати додаткові бібліотеки.

Переваги Angular:

- Спрощений процес розробки: Angular надає комплексний набір інструментів, що допомагає вирішувати багато задач, включаючи маршрутизацію та управління станом додатку.
- Модульність та сповіщення про помилки: Angular впроваджує концепцію модульності, яка спрощує розробку та підтримку великих додатків. Також, Angular має потужний механізм обробки помилок.

Недоліки Angular:

- Велика вага коду: Додатки, розроблені на Angular, можуть бути більшими за аналогічні на React або Vue.
- Складність навчання: Angular має досить високий поріг входження для нових розробників через свою складність та обширну документацію.

Переваги Vue:

- Легко навчитись: Vue є дуже легким для навчання, що робить його привабливим для новачків та невеликих команд.
- Висока продуктивність: Vue має дуже швидкий і мінімальний віртуальний DOM, що допомагає підтримувати високу швидкодію додатку.

Недоліки Vue:

- Менше сторонніх бібліотек та компонентів: Хоча Vue має активну спільноту, кількість готових компонентів та бібліотек менша порівняно з React або Angular.
- Менша спільнота розробників: За кількістю розробників та вакансій на ринку Vue трохи поступається React та Angular.

В даному застосунку буде використано React, зважаючи на всі його переваги та недоліки.

ибір технологій для бек частини

Для бек частини розглядалися два варіанти, які є найпопулярніші на ринку: Java та ASP .NET Core. Розглянемо обидва варіанти, та виберемо найкращий.

ASP.NET Core - це відкрите, кросплатформенне та високопродуктивне фреймворк для розробки веб-додатків. Він розроблений компанією Microsoft і є наступним поколінням ASP.NET Framework. Одна з ключових особливостей Windows, так і на платформі Linux та macOS.

Основні переваги ASP.NET Core:

- Кросплатформенність: Додатки можуть бути розгорнуті на різних операційних системах, що дозволяє зменшити витрати на інфраструктуру.
- Відкритий код: ASP.NET Core має відкритий вихідний код та активну спільноту розробників, що сприяє розвитку та підтримці фреймворку.
- Висока продуктивність: ASP.NET Core надає високу продуктивність та масштабованість завдяки асинхронному програмуванню та іншим оптимізаціям.
- Модульність: Фреймворк розроблений з урахуванням модульності, що дозволяє використовувати лише ті компоненти, які потрібні для конкретного проекту.
- Підтримка хмарних рішень: ASP.NET Core ідеально підходить для розробки хмарних додатків та мікросервісів.

Аналогом ASP.NET Core на Java є фреймворк Spring. Spring - це потужний фреймворк для розробки веб-додатків і включає в себе модулі та інструменти для створення сучасних, масштабованих та надійних додатків. Він пропонує різні

підпроекти та бібліотеки, зокрема Spring Boot, Spring MVC і Spring Security, що дозволяють створювати як веб-додатки, так і великі підприємницькі рішення.

Spring є дуже популярним в середовищі Java-розробників і має активну спільноту. Він підтримує архітектурні стилі, такі як REST, і надає інструменти для взаємодії з базами даних, створення API та керування сесіями користувачів. Spring також пропонує можливості інверсії управління та аспектно-орієнтованого програмування.

Оскільки ці дві технології дуже схожі, але вибір припав на .NET, оскільки ця система краще інтегрується із Microsoft SQL Server.

ибір технологій для кешу

Для кешу вибір був між Redis та Apache Kafka. Ці дві технології призначені для пришвидшення роботи застосунку. Порівняємо ці два варіанти.

– це розподілена база даних у вигляді ключ-значення, яка зберігає дані в оперативній пам'яті. Redis широко використовується для кешування даних та прискорення доступу до них. Він дуже швидкий завдяки тому, що дані зберігаються в оперативній пам'яті, і він підтримує багато різних структур даних, включаючи строки, набори, хеш-таблиці та списки.

Redis також має можливості для роботи з розподіленими системами, реплікацією та відмовостійкістю. Він підтримує декілька мов програмування для взаємодії, включаючи Python, Java, C# та інші.

Apache Kafka - це розподілена платформа для обробки та передачі потокових даних у реальному часі. Вона розроблена для роботи з великими обсягами даних та надає можливість надійної обробки повідомлень, розподіленої обробки даних, збереження та відтворення потоків даних. Apache Kafka широко використовується для реалізації журналів подій, аналізу даних у реальному часі, обробки великих потоків даних та інших сценаріїв, де необхідно ефективно та надійно керувати потоками інформації.

dis відомий своєю простотою та легкістю використання, в той час як Apache

Kafka є складнішим у налаштуванні та вимагає більше ресурсів для підтримки. Тому вибір впав на Redis.

ОЗРОБКА СИСТЕМИ

нструменти розробки

Для розробки програмного забезпечення в цілому і веб сайтів конкретно створено безліч застосунків, які спрощують цю роботу, прискорюють її, та навіть дають поради. Оскільки технологій для розробки є дуже багато, то і інструментів розробки є ще більше. Важливо знати існуючі рішення, щоб вибрати найзручніший інструмент для розробки певної технології. В загальному такі інструменти мають назву IDE.

Інтегроване середовище розробки, або IDE (Integrated Development Environment), є комп'ютерною програмою, яка надає розробникам засоби та середовище для створення, редагування, тестування та налагодження програмного забезпечення. IDE об'єднує різні інструменти та функціональність, необхідну для розробки програм на конкретній мові програмування або для роботи з певними технологіями [13].

Основні компоненти та функції IDE включають текстовий редактор, який дозволяє розробникам створювати та редагувати вихідний код, засоби автодоповнення, які пропонують автоматичні підказки та завершення коду для швидшого та точного програмування, відлагодження (dubugging), що надає інструменти для виявлення та усунення помилок в коді, встановлення точок зупинки, відстеження змінних тощо.

Також до сучасних IDE часто входить управління версіями, що допомагає зберігати та відстежувати зміни в коді за допомогою систем контролю версій, таких як Git.

Тому розглянемо IDE та застосунки, які були використані для створення даного проекту.

Для розробки API та Notification Service було використано Visual Studio 2022.

Visual Studio – це інтегроване середовище розробки (IDE) від Microsoft, яке призначене для розробки програмного забезпечення на різних платформах та мовах програмування. Visual Studio відомий своєю потужністю та гнучкістю, що робить його популярним інструментом серед розробників для створення різноманітних програмних продуктів. На рисунку 3.1 можна побачити інтерфейс VS 2022.

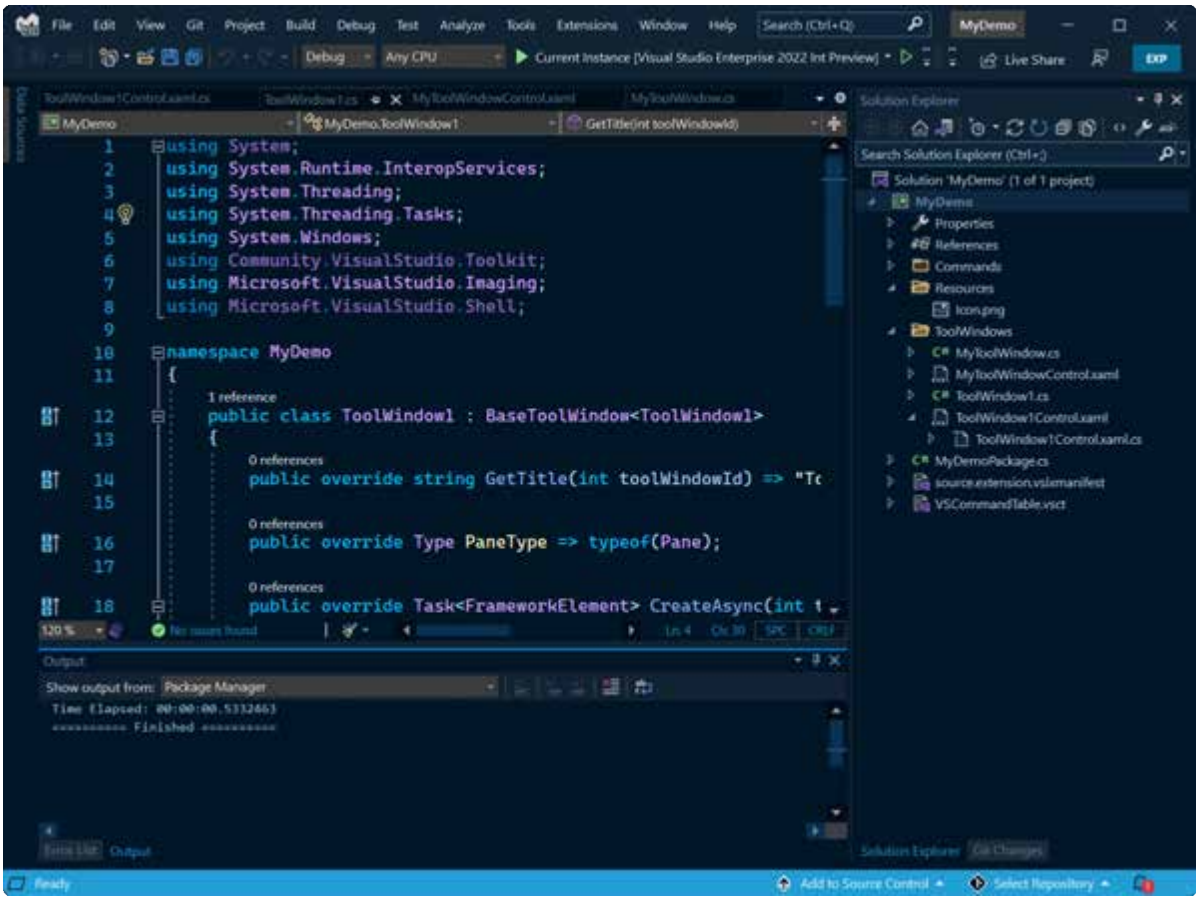

Рисунок 3.1 – Інтерфейс VS 2022 [14]

Також важливо відмітити, що цей інструмент добре підходить для розробки застосунків на мові програмування C#, на якій написано два компоненти системи. Саме тому вибір впав саме на цю IDE.

Для розробки фронтової частини сайту було використано Visual Studio програмного забезпечення, створений компанією Microsoft. Цей інструмент призначений для програмістів, які розробляють програми на різних мовах

програмування та платформах. Visual Studio Code вирізняється своєю легкістю використання та розширюваністю завдяки багатьом доступним плагінам. Він має інтуїтивний інтерфейс та надає зручні можливості для написання, редагування та відлагодження коду. Крім того, він підтримує багато мов програмування та має інтегровані інструменти для керування версіями, допомагаючи розробникам працювати більш продуктивно та ефективно. Visual Studio Code є популярним вибором для багатьох розробників завдяки своїй широкій функціональності та можливості адаптувати його до конкретних потреб кожного користувача. На рисунку 3.2 можна побачити інтерфейс VS Code.

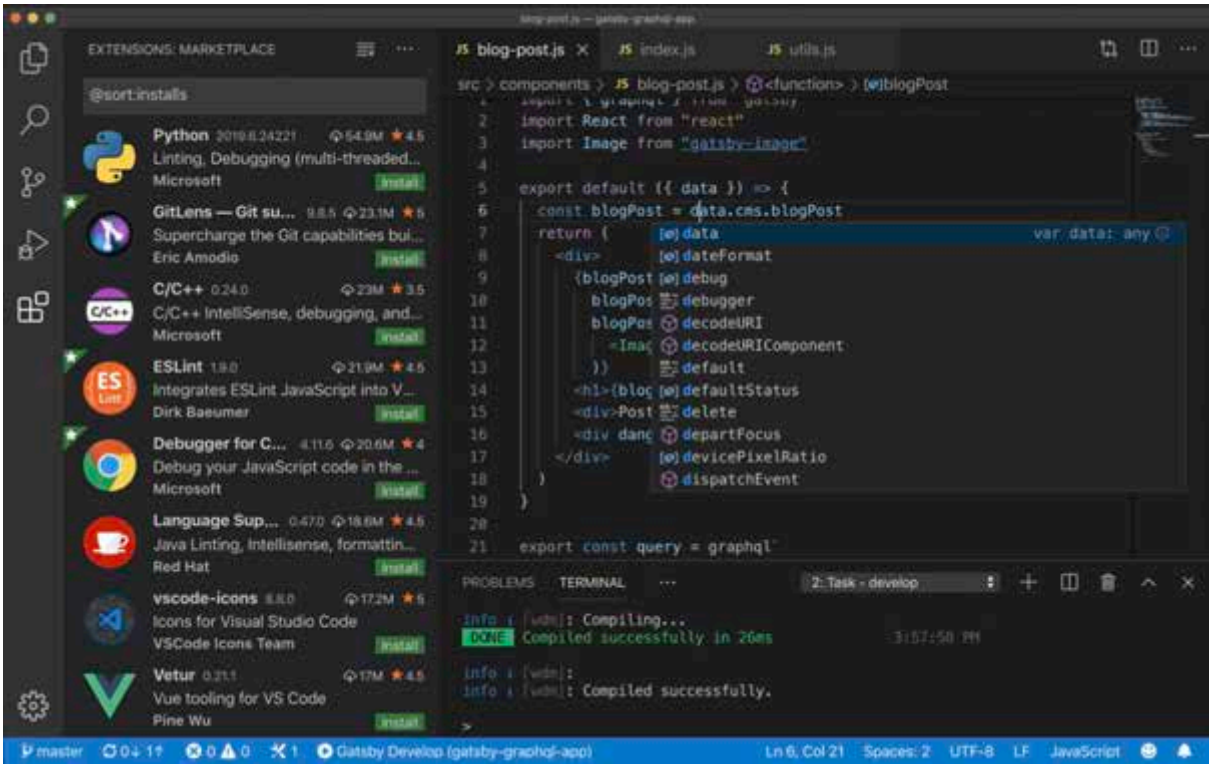

Рисунок 3.2 – Інтерфейс VS Code

Розробка React застосунків в Visual Studio Code дуже зручна завдяки ряду функціональних переваг. VS Code надає автодоповнення для React-компонентів та JavaScript, інтегрований відлагоджувальник для відлагодження коду, можливості встановлення розширень для підтримки React, а також допомогу у керуванні версіями та залежностями через вбудований термінал і інтеграцію з

системами контролю версій, що спрощує процес розробки React-застосунків в цьому середовищі.

Для проектування та керування БД було використано інструмент під назвою SQL Server Management Studio. SQL Server Management Studio, також відомий як SSMS, це програмне забезпечення, розроблене компанією Microsoft для адміністрування та управління базами даних SQL Server. Це потужний інструмент, який надає можливість створювати, редагувати, видаляти та оптимізувати бази даних, а також виконувати SQL-запити для взаємодії з даними.

SSMS також має інтегрований відлагоджувальник для зручного аналізу запитів та процедур, а також надає інструменти для моніторингу та налагодження параметрів бази даних. Він є важливим інструментом для баз даних SQL Server та допомагає адміністраторам та розробникам ефективно управляти та підтримувати дані в них. Інтерфейс – на рисунку 3.3.

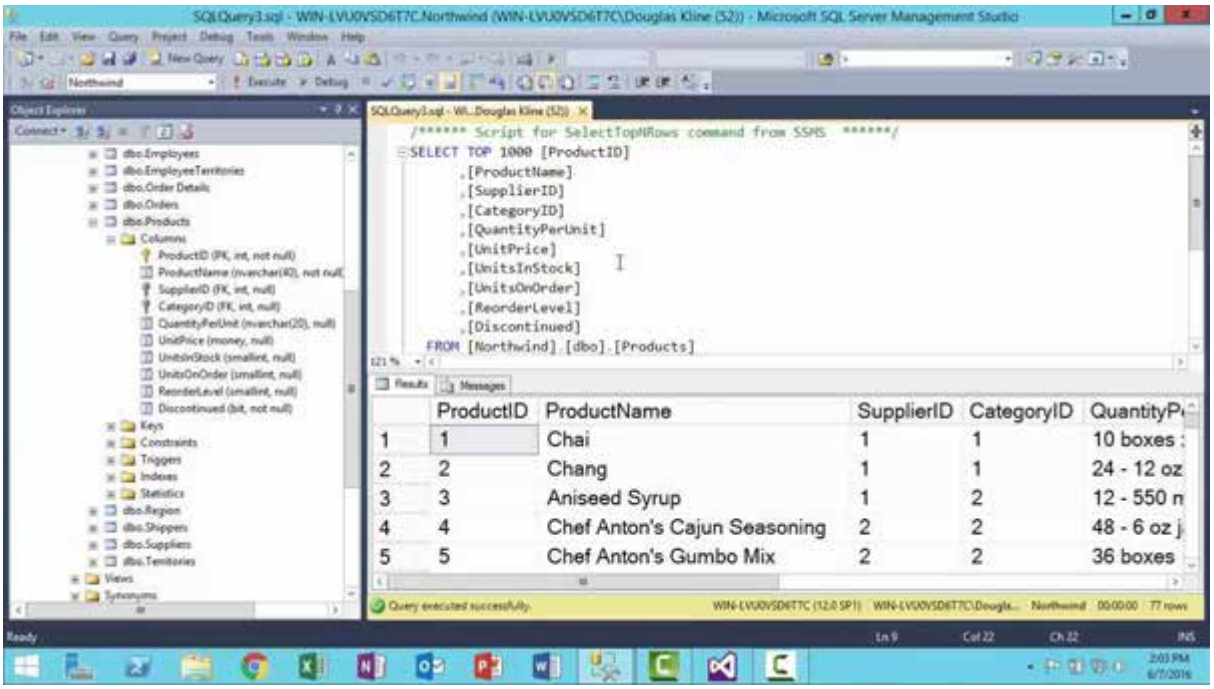

Рисунок 3.3 – Інтерфейс SSMS

Важливо згадати про процес розробки дизайну для сайту. На цей раз було не важко зробити вибір, адже найпопулярніший і найлегший для освоєння інструмент це Figma.

Figma – це онлайн-платформа для дизайну і спільної роботи, яка призначена для створення і прототипування веб-сайтів і мобільних додатків. Вона дозволяє дизайнерам та командам створювати векторні графічні елементи, макети та інтерактивні прототипи, обмінюватися даними та спільно працювати над проектами в реальному часі. Figma також підтримує відстеження змін та коментування, спрощуючи спільну роботу над дизайном і забезпечуючи швидкий процес розробки, що видно на рисунку 3.4.

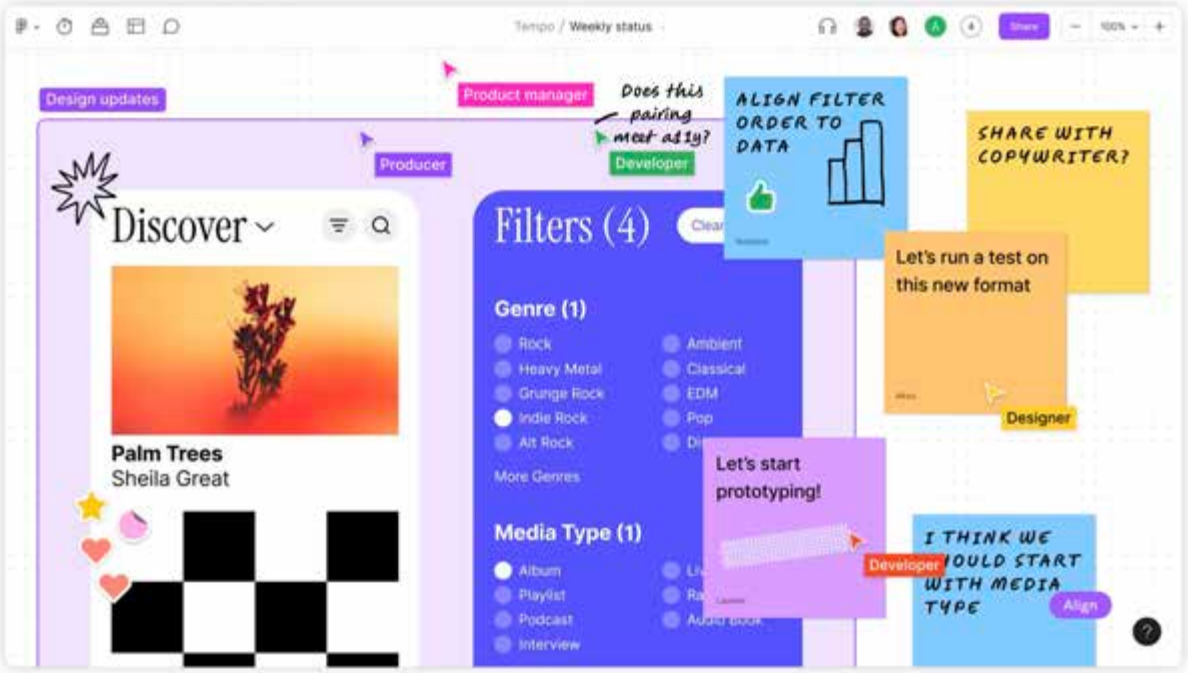

Рисунок 3.4 – Інтерфейс Figma

Отже так виглядає список інструментів, які були використані, але це не всі. Важливо згадати RedisInsight, що полегшує роботу із кешем Redis. Для контрою версій було використано Git та GitHub, а для деплою та хостингу планується використання Azure Services.

озробка різних рівнів застосунку

озробка рівня доступу до даних

Рівень доступу до даних являє собою базу даних, кеш, а також сервіси в API компоненті, які відповідають за доступ до самої БД та кешу. Для розуміння

розглянемо діаграму таблиць бази даних і коротко опишемо для чого вони потрібні. На рисунку 3.5 можна побачити цю діаграму.

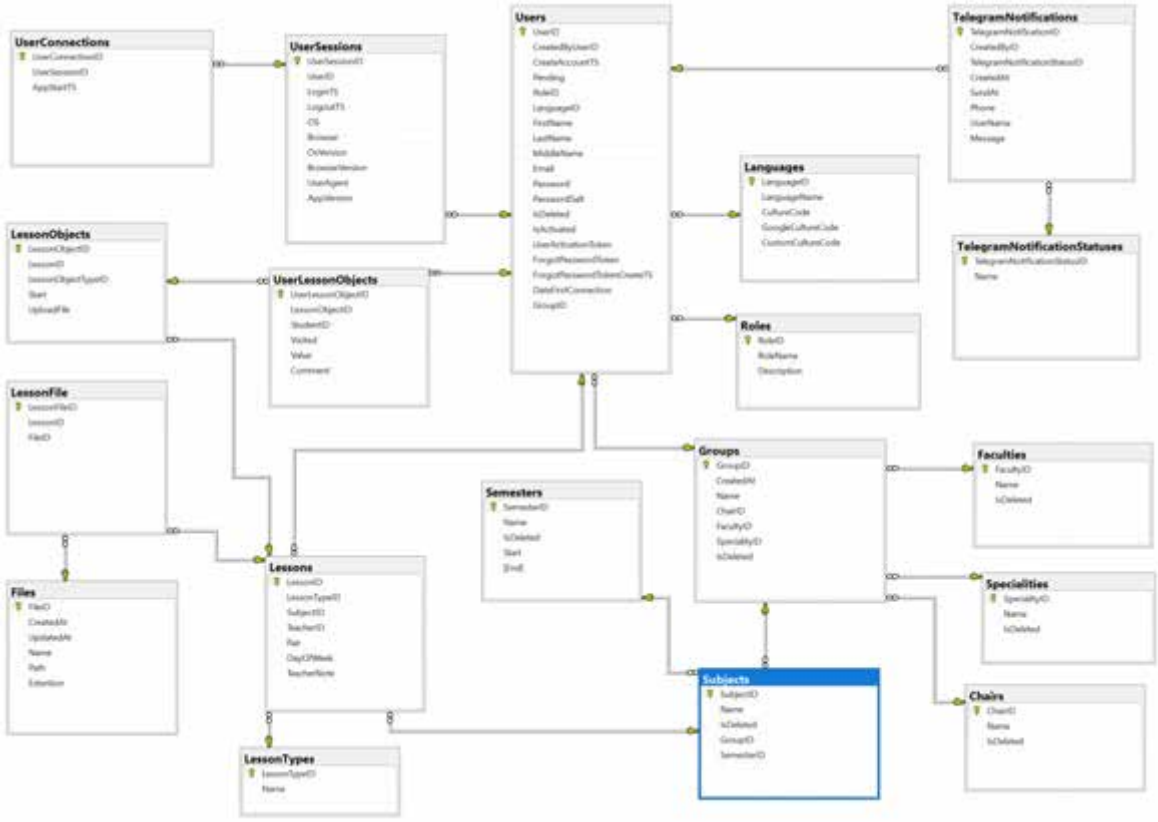

Рисунок 3.5 – діаграма таблиць БД

Почнемо із таблиці Users, ця таблиця, як не важко здогадатися відповідає за зберігання даних про користувача в системі. Нагадаю, що користувачем може бути як адмін, викладач, так і студент. В цій таблиці зберігається як і системна інформація, така як дата створення, ким створено, захешований пароль, прапорець, що свідчить про те, чи видалений юзер та чи активний. До інших даних таблиці, які юзер зможе побачити в інтерфейсі сайту належать FirstName, LastName, MiddleName, Email. Якщо цей юзер є студентом, то в нього заповнене поле GroupID, яке відповідає за групу, в якій навчається студент, якщо це не студент, то це поле буде порожнє. Зв’язок між таблицею Group і User є один або нуль до багатьох.

Також можна розуміти по діаграмі, що таблиця користувачів пов’язана із таблицею ролей зв’язком один до багатьох. В таблиці ролей зберігаються три

записи про ролі в системі. Можна більш детально розповісти про зв’язки між таблицями в реляційній БД.

Зв'язки в РБД зазвичай встановлюються за допомогою зовнішніх ключів. Зовнішні ключі - це поля в одній таблиці, які посилаються на первинний ключ іншої таблиці. Це дозволяє створити взаємозв'язок між записами цих таблиць.

Існують різні типи зв'язків, такі як один-до-одного, один-до-багатьох і багатьох-до-багатьох. Один-до-одного зв'язок означає, що кожен запис в одній таблиці відповідає одному запису в іншій таблиці. У випадку один-до-багатьох, один запис в одній таблиці відповідає багатьом записам в іншій таблиці, і навпаки. Багато-до-багатьох означає, що багато записів в одній таблиці відповідають багатьом записам в іншій таблиці, і навпаки.

Далі перейдемо до таблиці UserSessions. Ця таблиця відповідає за дані того, як часто користувач заходить в систему. Кожен раз при логіні додається запис в цю таблицю і тоді можна вести статистику по тому, як часто юзер заходив в систему. Наприклад, якщо студент зайшов на ресурс лише кілька разів за рік, то адмін може це побачити і тоді можна відрахувати студента.

Далі поговоримо про таблицю TelegramNotifications. Ця таблиця відповідає за зберігання повідомлень, які треба відправити, або які були відправлені. Notification Service дивиться в цю таблицю, після чого всі записи, для яких статус не відправлено стають в чергу на відправку. Також у випадку не відправленого повідомлення можна прочитати повідомлення про помилку, яке також буде в цій таблиці.

При проектуванні БД треба дотримуватися певних правил для того, щоб не використовувалася лишня пам’ять і була максимальна швидкодія. Для цього вже давно вигадані нормальні форми.

Нормальні форми в базах даних - це система правил та стандартів для організації та структурування даних в реляційних базах даних (РБД) з метою запобігання аномаліям та забезпечення ефективної роботи з інформацією. Ця

концепція була розроблена для забезпечення цілісності даних та уникнення дублювання інформації.

Нормальні форми включають ряд рівнів, від 1НФ до 5НФ (іноді 6НФ), кожен з яких встановлює певні вимоги до організації таблиць та зв'язків між ними. Наприклад, перша нормальна форма (1НФ) вимагає, щоб кожен атрибут в таблиці мав атомарне значення, тобто не містив складних даних. Друга нормальна форма (2НФ) передбачає, що атрибути, які не є частиною первинного ключа, повинні повністю залежати від цього ключа.

Використання нормальних форм допомагає створити добре структуровану базу даних, яка є ефективною для запитів, аналізу та збереження даних. Це допомагає уникнути проблем, таких як аномалії вставки, оновлення та видалення даних, і полегшує підтримку та розширення бази даних в майбутньому.

Далі можемо поговорити про таблицю, яка описує групу. Група має назву, рік вступу, і статус, чи видалена вона. Faculties, Specialities, Chairs – це таблиці, які відповідають за факультети, спеціальності та кафедри. Таблиця груп має зв’язок із кожною із цих таблиць.

Далі поговоримо про таблицю Subject, яка відповідає за предмет. В рамках даної абстракції предмет являє собою один предмет, який є в однієї групи і в одному семестрі. Тобто якщо фізика є в груп ІТ-2000 та ІТ-2001, то це будуть два записи в одній таблиці.

Для пришвидшення роботи БД використовується індексація. Індексація в базах даних - це метод швидкого пошуку інформації, який полегшує виконання структури для ефективного доступу до даних та пошуку записів в таблицях. Індекси можуть бути унікальними або неунікальними, і їхнє використання допомагає прискорити обробку даних в базі даних.

Якщо говорити про процес створення таблиць в БД, то було використано Entity Framework Core, який дозволяє створити таблиці на основі коду на C#. "Code First" в Entity Framework – це підхід для розробки додатків, де розробники

спочатку визначають модель даних у вигляді класів C# та зв'язків між ними, а потім система генерує структуру бази даних автоматично. Застосунок може взаємодіяти з даними як з об'єктами C#, і зміни в моделі можуть бути автоматично відображені в базі даних за допомогою механізму "Code First Migrations". Цей підхід спрощує розробку та обслуговування бази даних для розробників, які більше спрямовані на об'єктно-орієнтований дизайн і не хочуть займатися написанням SQL-запитів або визначенням схеми бази даних заздалегідь. Цей процес можна побачити на рисунку 3.5.

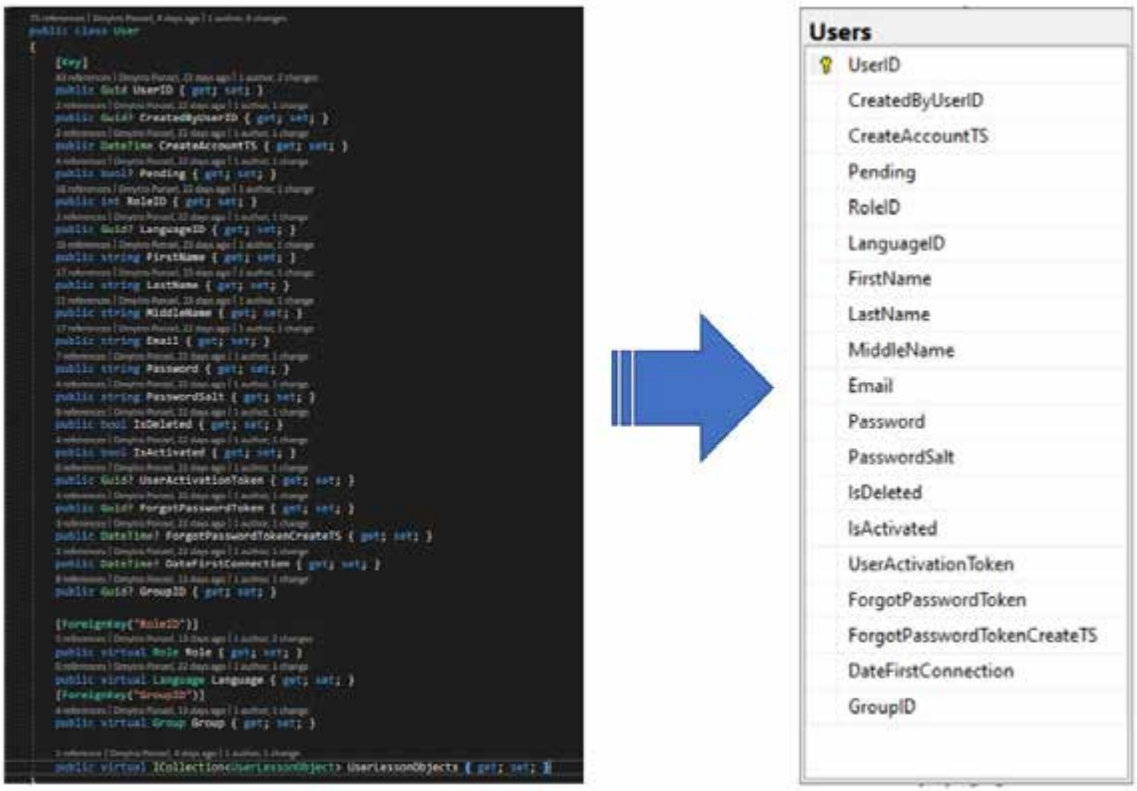

Рисунок 3.5 – Приклад створення таблиці в БД на основі класу в C#

Зліва можна побачити клас User, що описує користувача в системі, а з правого боку видно створену таблицю.

Також для покращення швидкодії використовується Redis. Він зберігає дані у швидкодії оперативній пам'яті, що дозволяє сайту швидко доступатися до часто використовуваних даних, таких як результати запитів до бази даних або сторінки. Кешування допомагає зменшити завантаження сервера та покращити відповідь сайту. Також Redis може використовуватися для зберігання сесій користувачів,

що дозволяє швидко визначати та забезпечувати доступ користувачам до їхніх особистих даних та стану сесії.

озробка бекенд частини

Бекенд частина відповідає за бізнес-логіку застосунку. В цій частині задіяні два компоненти: API та сервіс сповіщень. Спочатку поговоримо про API компонент.

Основною задачею цього компоненту є опрацювання запитів, які приходять із фронтової частини. Компонент складається із трьох основних проектів: Data, Core та WebAPI. Про проект Data йшлося в попередньому пункті. Далі поговоримо про проекти Core та WebAPI. В проекті Core знаходяться всі основні сервіси, які відповідають за логіку роботи: UserService, AccountService, GroupService, LessonService.

AccountService відповідає за всі маніпуляції із акаунтами в системі. Цей клас має залежності, які передаються через Dependency Injection, про який буде сказано пізніше.

Розглянемо метод, який відповідає за додавання користувача в групу, а саме студента. На рисунку 3.6 можна побачити цей метод.

```
public void AddUserToGroup(Guid userId, Guid groupId)
{
    var user = _repository.FirstOrDefault<User>(x => x.UserID == userId);
    if(user == null)
    {
        throw new UserNotFoundException("Користуача не знайдено");
    }

    user.GroupID = groupId;
    _repository.Update<User>(user);
    var lessonObjects = _repository.FindAll<LessonObject>(x => x.Lesson.Subject.GroupID == groupId);
    foreach (var lessonObject in lessonObjects)
    {
        var newUserLessonObject = new UserLessonObject
        {
            UserLessonObjectID = Guid.NewGuid(),
            LessonObjectID = lessonObject.LessonObjectID,
            StudentID = userId,
        };
        _repository.Insert<UserLessonObject>(newUserLessonObject);
    }
    _repository.SubmitChanges();
}
```

Рисунок 3.6 – метод AddUserToGroup

Це не складний метод, в ньому можна побачити як відбувається процес додавання. Спочатку треба перевірити чи такий користувач існує в системі. Якщо його немає, то програма поверне помилку. Якщо користувач є, то йому присвоїться група, а також будуть створені навчальні дні для всіх предметів, шо має ця група. Вхідними параметрами методу є ідентифікатор групи та користувача.

Тепер кілька слів про ін'єкцію залежностей. Dependency Injection, DI – це паттерн проектування в програмуванні, який допомагає зменшити залежність між компонентами вашої програми. В основі DI лежить ідея того, що об'єкти не повинні створювати свої залежності, але повинні отримувати їх ззовні. Це сприяє більшій гнучкості і покращує спроможність тестування програмного коду.

Завдяки DI, об'єкти отримують свої залежності через конструктори або методи, що дозволяє легко замінювати різні реалізації залежностей і підтримувати їх відокремлені від основного коду програми. Цей підхід робить код більш читабельним і підтримуваним, особливо в великих програмах.

В компоненті також є контролери, які безпосередньо обробляють HTTP запити. Вони взаємодіють із сервісами, які інжектяться в контролери за допомогою вищезгаданого DI.

Тепер поговоримо про сервіс сповіщень, основною роботою якого є регулярний перегляд БД, та відправка повідомлень в телеграм. Він працює як звичайний воркер в .NET Core. Воркери (Workers) – це спеціальний тип додатків, які призначені для виконання фонових завдань асинхронно та без користувацького інтерфейсу. Воркери дозволяють розробникам створювати і підтримувати додатки, які виконують завдання, такі як обробка черги повідомлень, періодичне виконання завдань або обслуговування інших фонових операцій. Ця архітектура стала доступною з появою Generic Host у .NET Core 3.0 і її подальшим розвитком.

Для того, щоб можна відслідкувати поведінку системи використовується логування. Логування в ASP.NET Core – це важлива частина розробки веб-

додатків, яка допомагає відстежувати і аналізувати події та дії, які відбуваються в вашому додатку. ASP.NET Core надає потужний і гнучкий механізм логування, який дозволяє вести журнали подій, помилок, викликів API та інших подій.

Для логування в ASP.NET Core зазвичай використовують бібліотеки, такі як Serilog, NLog, чи вбудований в ASP.NET Core ILogger. За допомогою цих інструментів, можна налаштувати різні джерела логування, такі як консоль, файл, база даних або сервіси моніторингу. Використання різних рівнів логування (наприклад, інформаційний, помилковий, налагоджувальний) допомагає швидко знаходити та виправляти проблеми в вашому додатку. Тому налаштоване логування було надзвичайно корисним при розробці

озробка фронтенд частини

Фронт частина є єдиною, з якою безпосередньо взаємодіє користувач. Являє собою React застосунок. На даний момент цей фреймворк підтримує два варіанти написання коду – на класах, та на функціях. Також можна змішувати підходи, використовуючи потрібні переваги обох підходів. На рисунку 3.7 можна побачити структуру проекту.

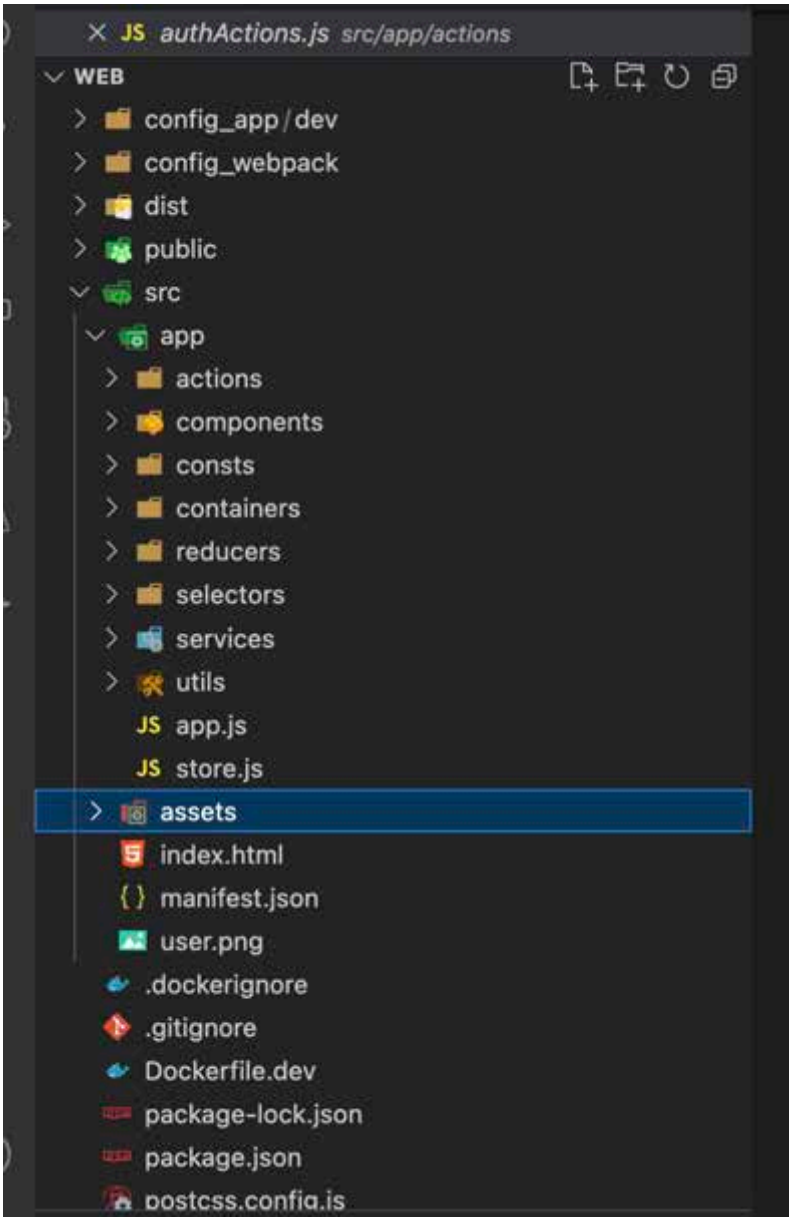

Рисунок 3.7 – структура фронт проекту

Для керування станом застосунку використовується Redux. Основною ідеєю Redux є зберігання всього стану додатку в одному об’єкті, який називається store, і використання actions для виконання змін в цьому стані. Зміни в стані здійснюються за допомогою reducers, які є чистими функціями, що обробляють дії і повертають новий стан. Цей підхід допомагає вирішувати проблеми складності та передбачуваності управління станом в додатках.

Redux також дозволяє використовувати різні розширення, такі як middleware, для обробки асинхронних запитів та інших асинхронних операцій. Тому ця бібліотека була дуже корисною при розробці.

На рисунку 3.8 можна побачити Schedule компонент.

```
@connect(mapStateToProps)
export default class Schedule extends React.PureComponent {
    onLessonClick = (lessonId) => {
        routerService.forwardTo(`${SUBJECT}/${lessonId}`)
    }

    findMaxPair = (scheduleInfo) => {
        let maxInFirst = 0;
        let maxInSecond = 0;
        if (!scheduleInfo){
            return { maxInFirst, maxInSecond };
        }

        for (let i = 0; i < 7; i++) {
            let maxPair = 0;
            scheduleInfo.days[i].pairs.forEach((x, j) => {
                if(x !== null){
                    maxPair = j + 1;
                }
            });
            if (maxInFirst < maxPair) {
                maxInFirst = maxPair
            }
        }

        for (let i = 7; i < 14; i++) {
            let maxPair = 0;
            scheduleInfo.days[i].pairs.forEach((x, j) => {
                if(x !== null){
                    maxPair = j + 1;
                }
            });
            if (maxInSecond < maxPair) {
```

Рисунок 3.8 – компонент Schedule

Компоненти відіграють роль блоків інтерфейсу в застосунку, наприклад, компонент Schedule відображає за розклад. Тобто в цей компонент входить як верстка самого компонента, так і його логіка роботи. Кожен компонент у React – це незалежний модуль, який приймає вхідні дані (props) і генерує вихідний код HTML. Компоненти можуть вкладатися один в одного, створюючи ієрархію для побудови складних інтерфейсів. Ваш додаток використовує стан (state) для збереження даних, які можуть змінюватися з часом, і React автоматично перерендерює компоненти, коли стан змінюється. Цей декларативний підхід до

розробки інтерфейсів робить React потужним і популярним інструментом для створення динамічних веб-додатків.

икористання гіперкубу для аналізу даних адміністратором

LAP-технології та сховище даних

Онлайн-аналітичний обробка (OLAP) є ключовим поняттям у сфері бізнес-аналітики і обробки даних. OLAP-куби відіграють важливу роль в цій сфері, допомагаючи організаціям аналізувати великі обсяги даних та виділяти важливі зв'язки та паттерни. OLAP-куби – це візуальна, багатовимірна модель даних, яка дозволяє швидко і ефективно аналізувати інформацію з різних точок зору.

OLAP-куби представляють дані у вигляді гіперкубів, де кожне вимірювання (наприклад, час, регіон, продукт) створює новий вимір куба. Аналізуючи ці куби, користувачі можуть виконувати різні операції, такі як згортка (roll-up), розгортка (drill-down), агрегація (summarization) та фільтрація. Це дозволяє швидко виявляти тенденції, аналізувати деталі, прогнозувати та приймати рішення на основі даних.

Однією з важливих переваг OLAP-кубів є їхнє використання в багатьох сферах, включаючи фінанси, маркетинг, логістику, галузевий аналіз та багато інших. Ці інструменти допомагають підприємствам вдосконалити стратегічне управління та зробити кращі рішення на основі даних. Вони є надійним засобом підтримки прийняття рішень і важливим інструментом для аналітиків та менеджерів у сучасному бізнесі.

Завдяки OLAP-кубам, користувачі можуть легко аналізувати великі обсяги даних, виявляти зв’язки і закономірності, що важливо для стратегічного планування та прийняття рішень. Вони створюють можливість для бізнесу бути більш гнучким та реагувати на зміни в середовищі швидше та з більшою точністю. Таким чином, OLAP-куби залишаються важливим інструментом у світі аналітики та бізнес-інтелекту [15].

Одним із компонентів системи є сховище даних. Сховища даних – це важлива складова сучасної інформаційної інфраструктури, що відіграє ключову роль у зберіганні, управлінні та забезпеченні доступу до даних. Сховища даних включають в себе різні технології та методи для збереження інформації, незалежно від обсягу та типу даних. Ці сховища даних використовуються в різних галузях, від бізнесу та медицини до науки та технологій [16].

Однією з основних функцій сховищ даних є забезпечення безпеки і надійності інформації. Забезпечення резервного копіювання та відновлення даних є ключовим аспектом сховищ даних, оскільки дозволяє відновити важливу інформацію в разі випадку втрати або пошкодження даних. За допомогою технологій реплікації і засобів контролю доступу, сховища даних дозволяють забезпечити конфіденційність та цілісність даних.

Сховища даних також відіграють важливу роль у аналітиці та бізнес-інтелекті. Вони дозволяють організаціям збирати, зберігати та аналізувати великі обсяги даних, отримуючи цінний інсайт для прийняття рішень. З використанням сучасних інструментів аналізу даних, сховища стають ключовою складовою для розвитку бізнесу та досягнення конкурентних переваг.

Сховища даних відіграють невід’ємну роль у світі сучасних технологій та інформаційного суспільства. Вони забезпечують зберігання та доступ до інформації, допомагають забезпечити безпеку та надійність даних, а також надають можливість аналізувати та використовувати дані для досягнення цілей та розвитку організацій та суспільства в цілому [17].

Під час виконання даної дипломної роботи було спроектовано сховище даних, яке можна побачити на рисунку 3.9.

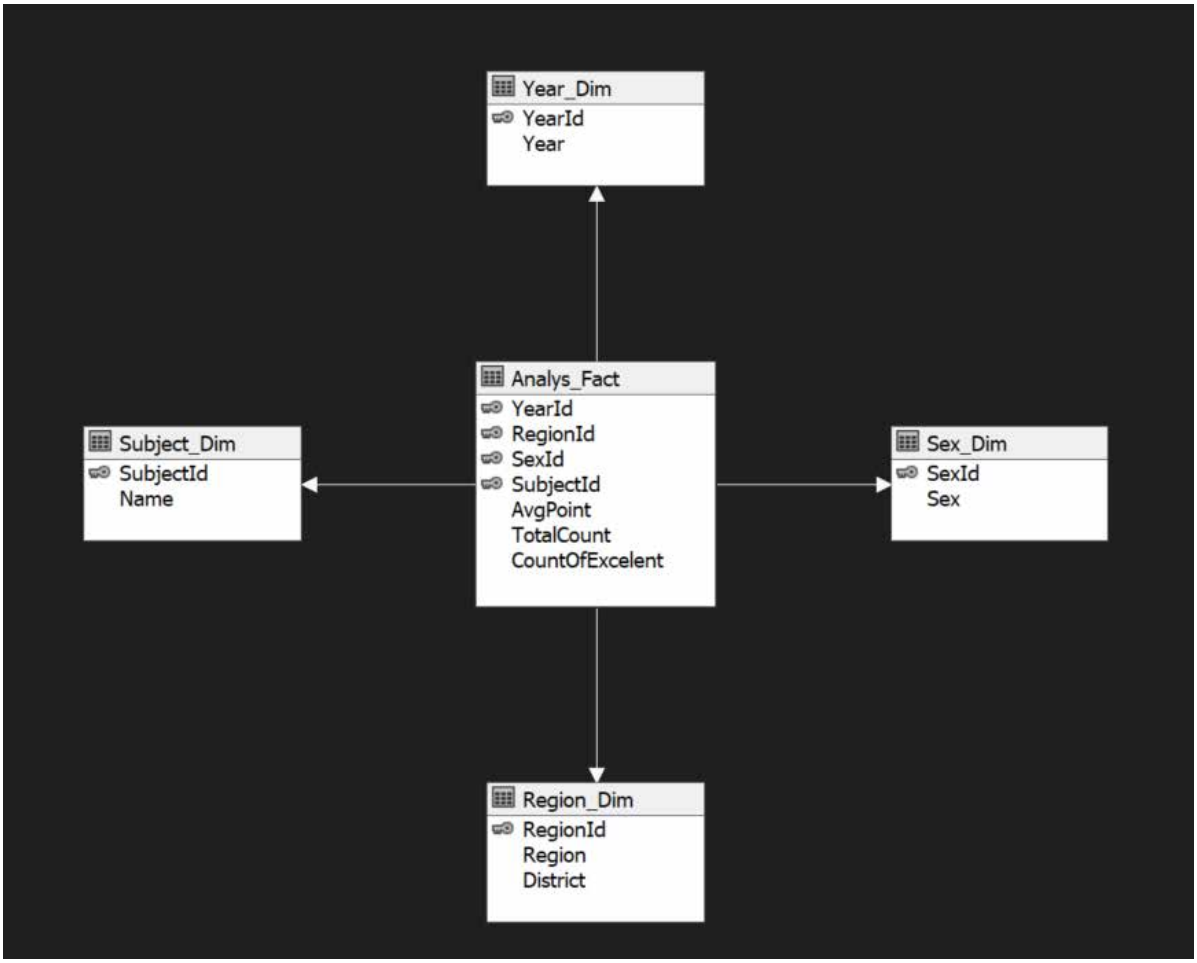

Рисунок 3.9 – структура сховища даних

В даному сховищі даних можна побачити 5 таблиць – чотири таблиці вимірів, які відповідають за рік навчання, стать студента, предмет і регіон, звідки родом студент, одна таблиця фактів, яка використовується збереження інформації про середній бал, загальну кількість студентів, а також кількість відмінників.

обудова гіперкубу та заповнення його даними

Для того, щоб побудувати гіперкуб і заповнити його даними використовувалися дві технології: SSAS та SSIS.

SQL Server Analysis Services, або SSAS, є однією з ключових компонент Microsoft SQL Server, яка дозволяє підприємствам та організаціям відкривати потенціал аналізу даних. SSAS надає потужні інструменти для створення, управління та аналізу багатовимірних моделей даних, які називаються "кубами".

Ці куби дозволяють користувачам легко експортувати та аналізувати великі обсяги даних, отримуючи цінну інформацію для прийняття рішень.
відповідає саме за створення OLAP-кубів, що дозволяють виразити дані в багатовимірній структурі. Це дає можливість аналізувати дані з різних точок зору та деталей, виконувати згортку та розгортку, агрегування та фільтрацію. Цей підхід робить аналіз даних більш інтуїтивним та ефективним для користувачів.

SSAS також підтримує різні режими роботи, такі як Multidimensional (MDX) та Tabular (DAX), що надається в залежності від потреб користувачів та характеру даних. Крім того, вона інтегрується з іншими продуктами Microsoft, такими як Excel, Power BI та SharePoint, що полегшує розповсюдження аналітичної інформації та спільну роботу з нею.

Потрібно створити проект типу Analysis Service. Потім створити представлення джерела даних (рисунок 3.10).

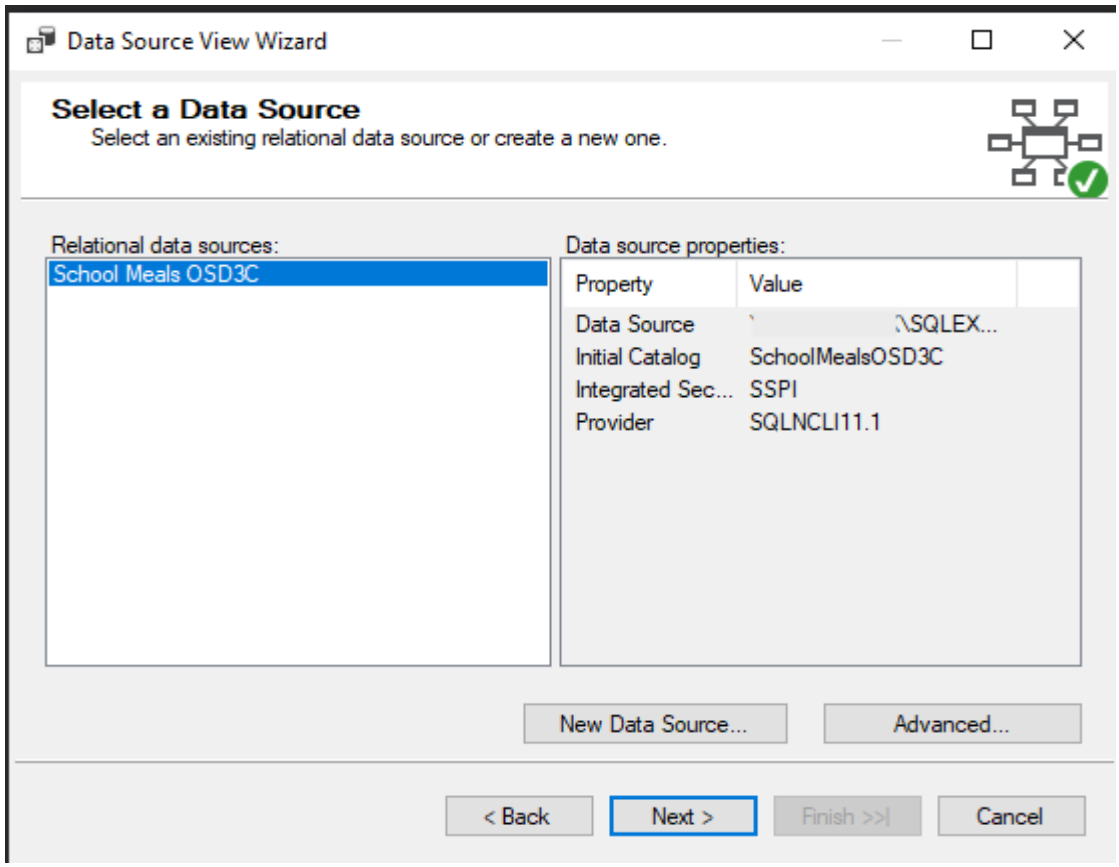

Рисунок 3.10 – створення представлення даних

Потім треба вибрати таблиці для кубу (рисунок 3.11), даних з таблиць та вимірів для кубу (рисунок 3.12).

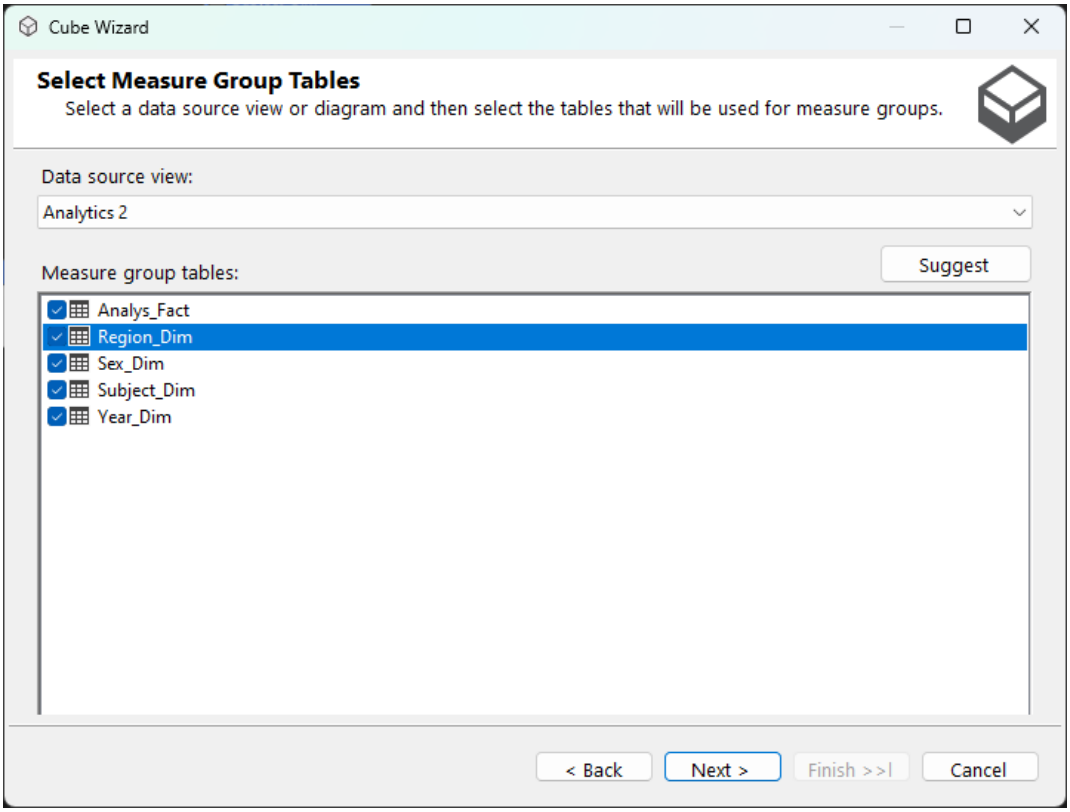

Рисунок 3.11 – вибір таблиць для кубу

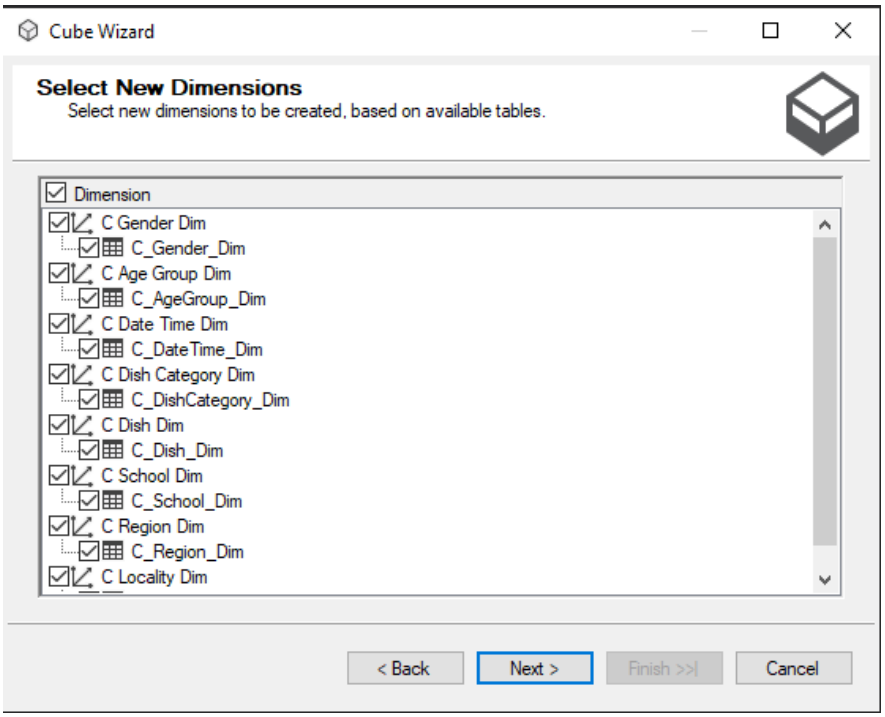

Рисунок 3.12 – вибір вимірів для кубу

В результаті ми отримали розгорнутий куб.

Далі за допомогою SSIS заповнюємо куб даними. SQL Server Integration – це потужний інструмент для інтеграції та обробки даних, розроблений корпорацією Microsoft. SSIS надає рішення для завдань імпорту, експорту, трансформації та завантаження даних між різними джерелами та цілями. Цей інструмент важливий для ефективного управління даними та забезпечення їх якості в області бізнес-аналітики та інформаційної обробки.

SSIS дозволяє створювати пакети обробки даних, які містять різні завдання та переходи між ними. Кожне завдання може включати інтеграцію з базами даних, файловими системами, веб-сервісами та іншими джерелами та призначеннями. Це дозволяє розробникам створювати складні робочі процеси для перенесення, трансформації та очищення даних з різних джерел.

Dataflow (потік даних) – це ключовий компонент, який дозволяє зчитувати, обробляти та передавати дані між різними джерелами та цілями. Dataflow в SSIS дуже корисний для імпорту, експорту, трансформації та очищення даних під час робочих процесів обробки даних.

У випадку даної курсової роботи є два завдання потоку даних: заповнення таблиць вимірів, та заповнення таблиці фактів (рисунок 3.13).

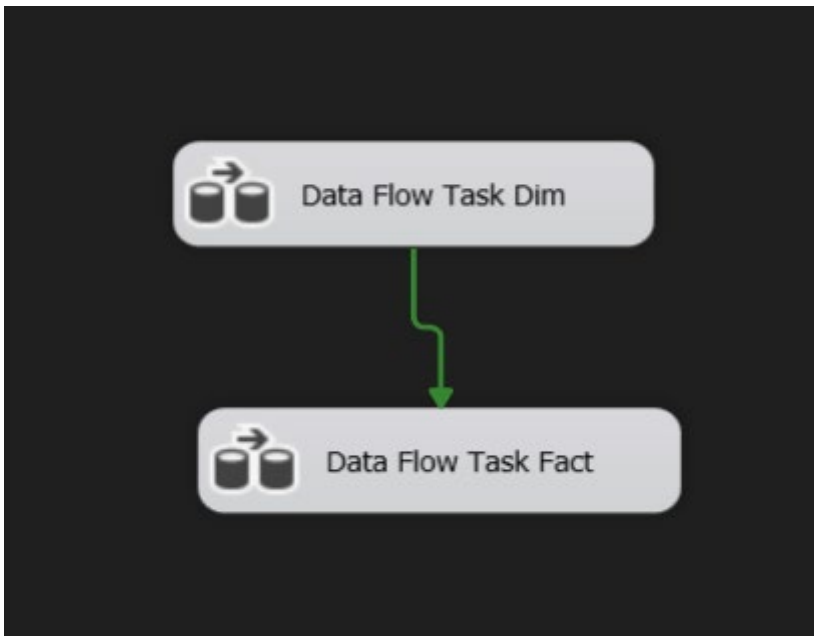

Рисунок 3.13 – завдання потоку даних

Розглянемо детальніше завдання для заповнення таблиць вимірів. Там є чотири джерела даних та чотири призначення потоку даних для заповнення всіх таблиць вимірів. Це зображено на рисунку 3.14.

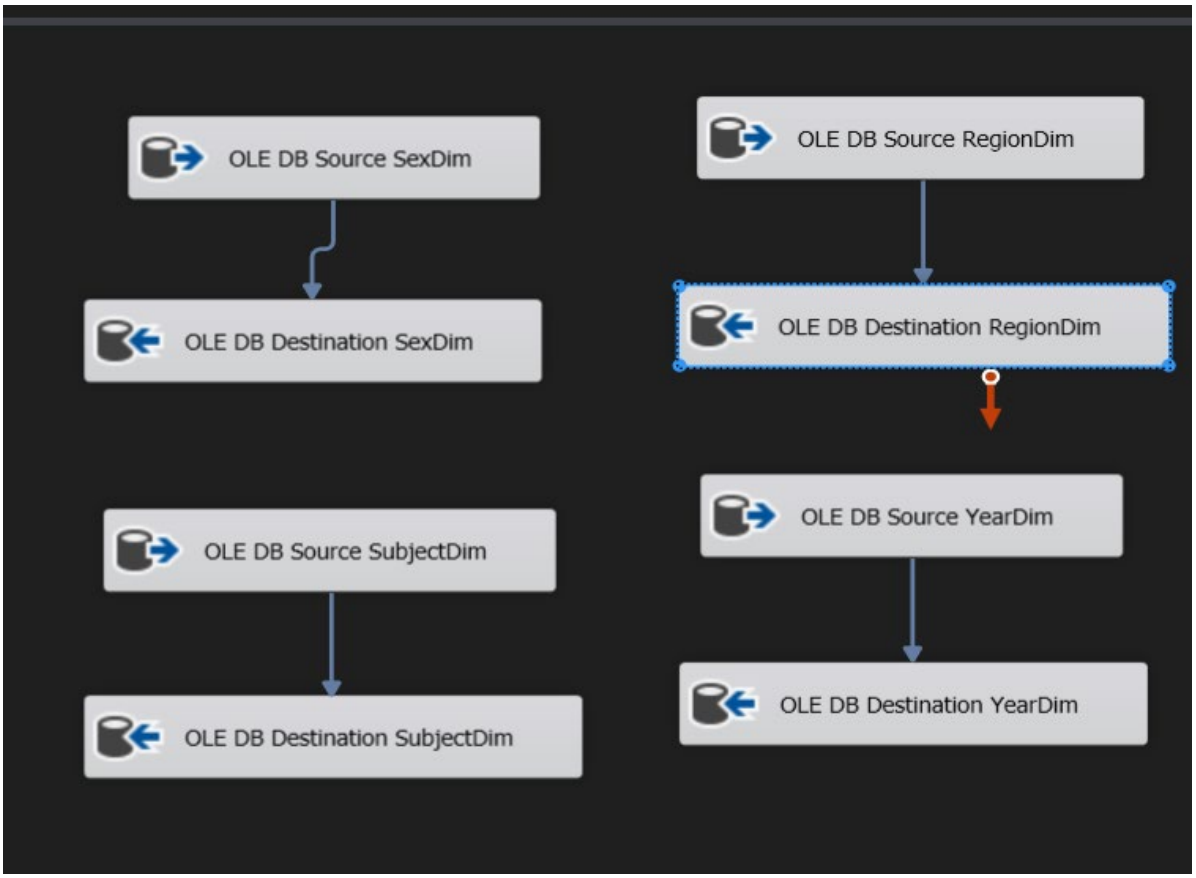

Рисунок 3.14 – детальне зображення завдання заповнення таблиць вимірів

Запустивши завдання ми отримаємо заповнений даними гіперкуб. Що знадобиться для подальшого аналізу даних в різних вимірах.

ЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

еревірка якості програмного продукту

Важливим і фінальним етапом розробки системи є перевірка якості її роботи, для цього необхідно пройтися по всіх функціях систему у ролі користувача, щоб знайти всі можливі дефекти.

Коли програма вперше запускається, то є лиш один користувач, який є адміністратором. Після першого входу пропонується змінити пароль. Це можна зробити в особистому кабінеті. Для цього треба здійснити вхід в систему, ввівши електронну пошту та пароль початкового користувача. Для цього введемо логін та пароль і тоді потрапимо в систему. На рисунку 4.1 можна побачити скріншот логін сторінки.

Рисунок 4.1 – вхід в систему

Після входу користувач потрапляє на головну сторінку. Там можна переглянути доступні розділи системи, а також побачити привітання і ім’я свого користувача та його роль. На рисунку 4.2 можна побачити, який вигляд має ця сторінка.

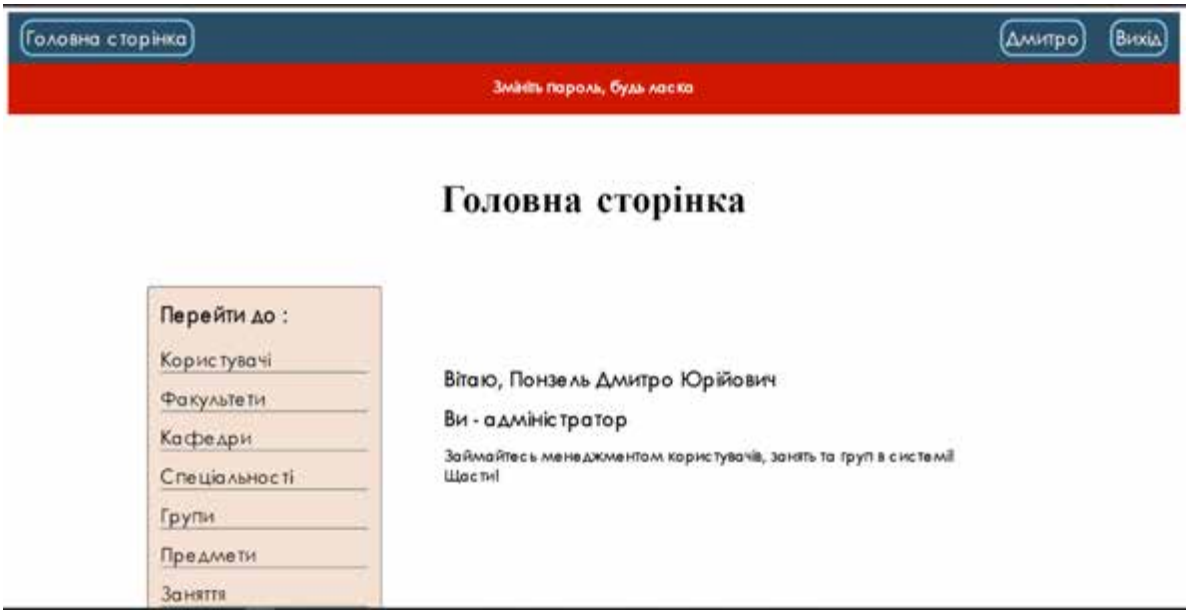

Рисунок 4.2 – головна сторінка

При першому вході в систему відображається прохання про зміну пароля, клікнувши на яке можна перейти в особистий кабінет для його зміни. Також в особистий кабінет можна потрапити через панель розділів, яка знаходиться зліва сторінки, або натиснувши на своє ім’я у верхньому лівому куті сторінки. На рисунку 4.3 можна побачити сторінку особистого кабінету.

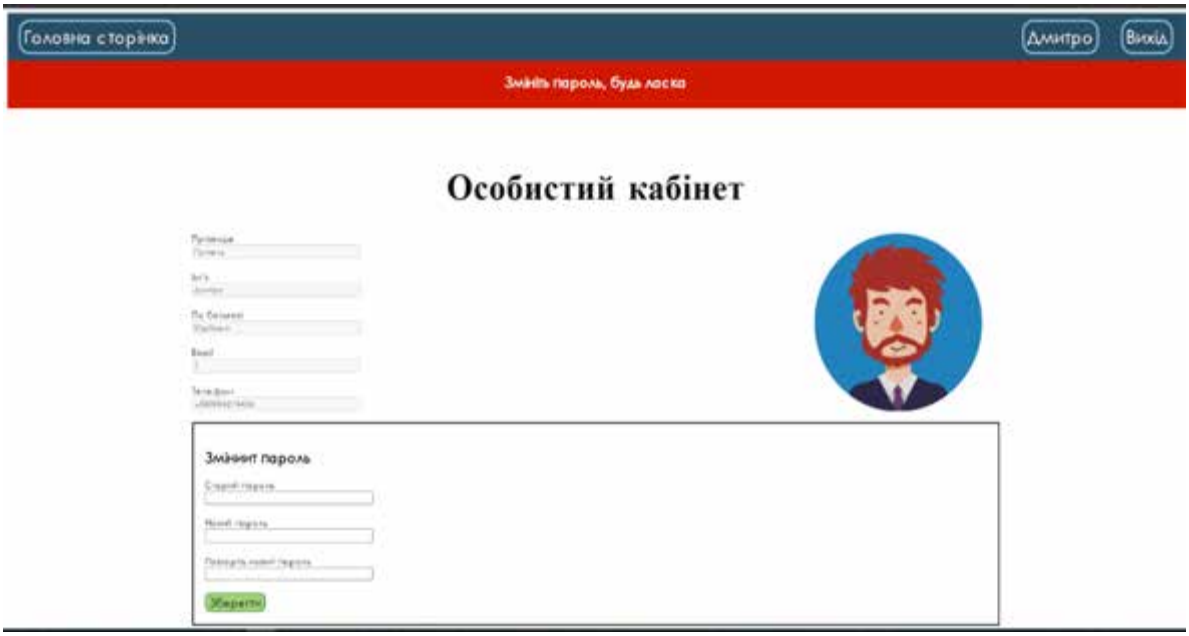

Рисунок 4.3 – сторінка особистого кабінету

На сторінці особистого кабінету користувач може переглянути дані про себе і змінити пароль. Для того, шоб змінити пароль треба ввести старий пароль, а також двічі ввести новий. Якщо старий пароль правильний і новий пароль проходить валідацію по кількості символів і типу цих символів, тоді є можливість його змінити. Після успішної зміни паролю потрібно буде заново ввійти в систему (вже під новим паролем).

Після зміни паролю можна приступити до створення сутностей в системі. Оскільки на початку даних на сайті майже не буде, то потрібно почати процес із додавання факультетів, спеціальностей та кафедр. На рисунку 4.4 можна побачити процес створення факультету.

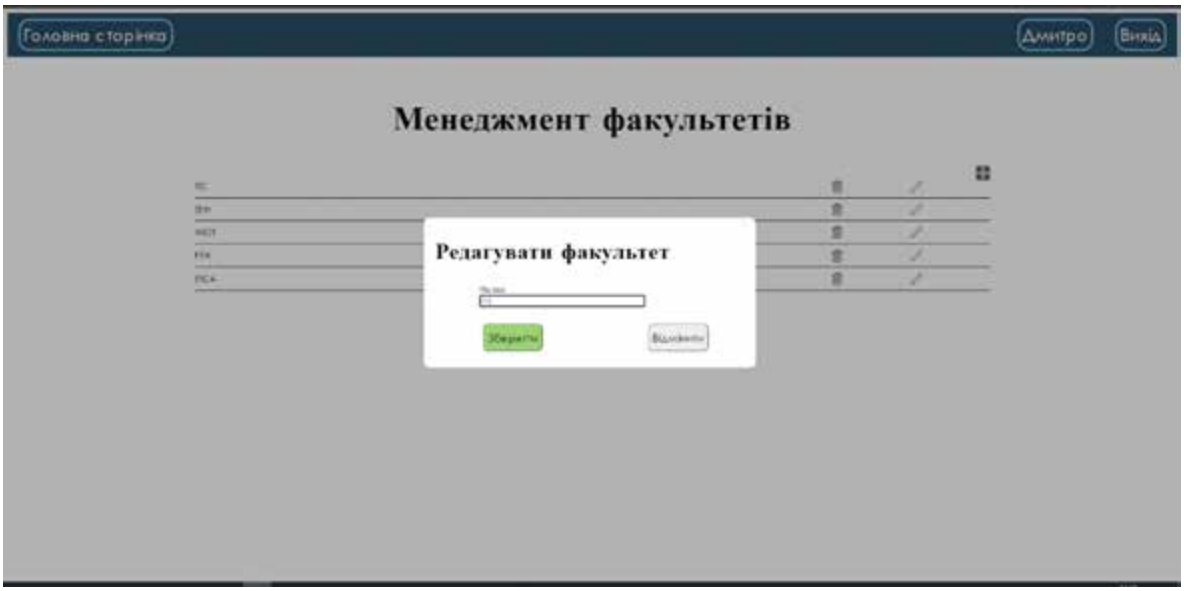

Рисунок 4.5 – додавання факультету

Для того, щоб додати факультет треба просто ввести його назву. Після створення факультетів, кафедр і спеціальностей можна приступити до створення груп. Кожна група належить до певного факультету, кафедри і спеціальності, тому при її створенні потрібно вказати ці дані із раніше створених. Також група має назву. На рисунку 4.6 зображено процес додавання групи до системи.

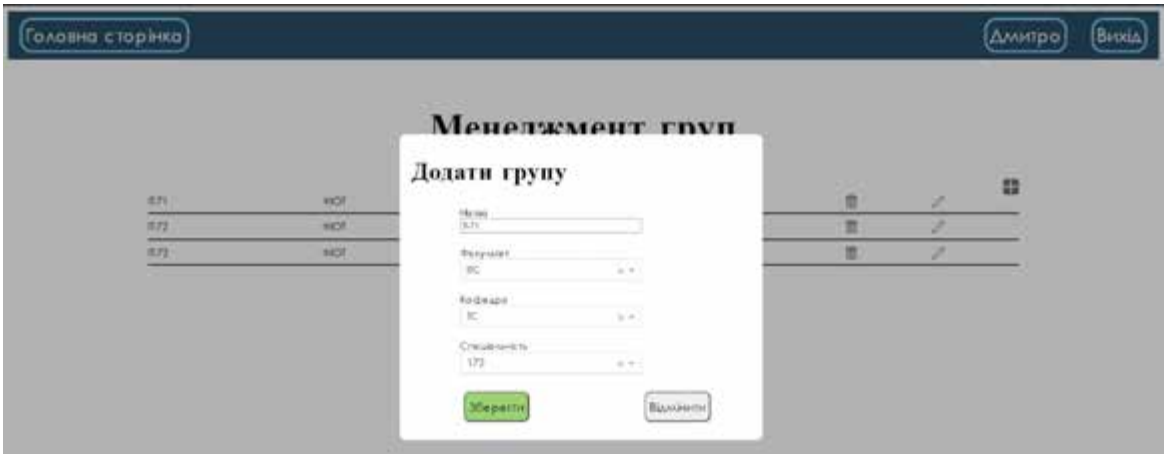

Рисунок 4.6 – створення групи

Після створення групи наступним кроком іде додавання користувачів до системи. Це можуть бути як студенти, так і викладачі. При додаванні студента потрібно вказати його персональні дані, такі як прізвище, ім’я, по батькові, пошту, номер телефону, стать. Також потрібно вибрати роль «Студент» і вказати до якої групи буде доданий студент. І необхідно вказати тимчасовий пароль.

Додавання користувача є на рисунку 4.7.

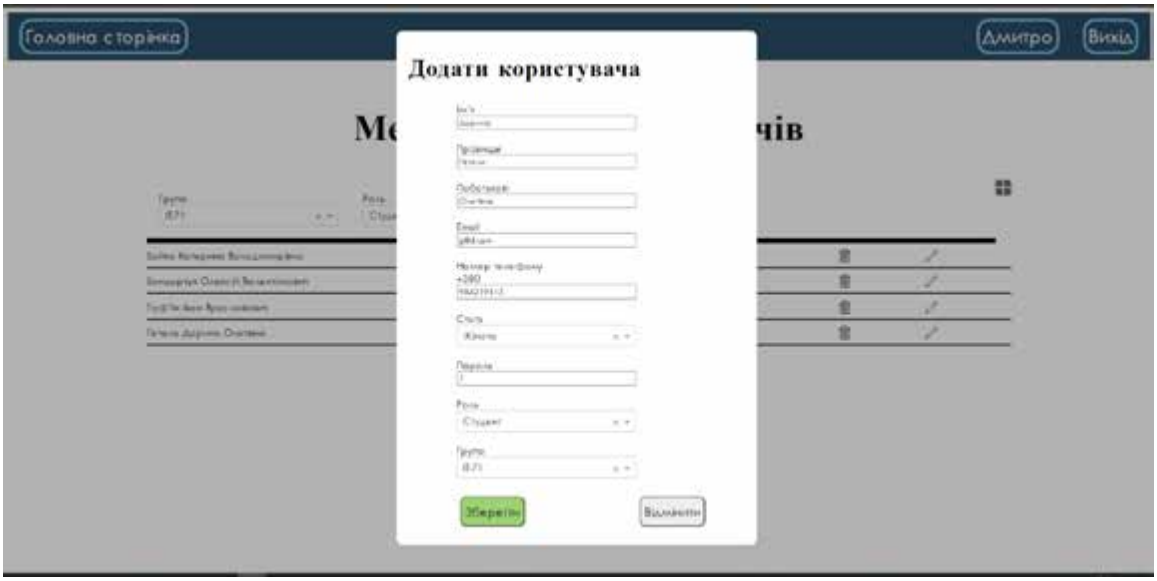

Рисунок 4.7 – додавання користувача

Так відбувається перевірка якості продукту для адміністратора.

Тепер пройдемося по основному функціоналу викладача. Коли він вперше входить в систему, то повинен змінити пароль, як і всі користувачі. Далі він

побачить привітання, свої дані та матиме можливість перейти на сторінку занять чи розкладу.

На сторінці зі розкладом відображається список занять для викладача по парах та днях. На сторінці видно розклад на два тижні (перший і другий тижні занять). В клітинках можна побачити час початку пари, а також назву предмету та групу, в якій буде проводитися пара. Для студента розклад виглядає аналогічно, тільки різниця в тому, що замість групи відображаються ініціали викладача. На рисунку 4.8 можна побачити розклад для викладача.

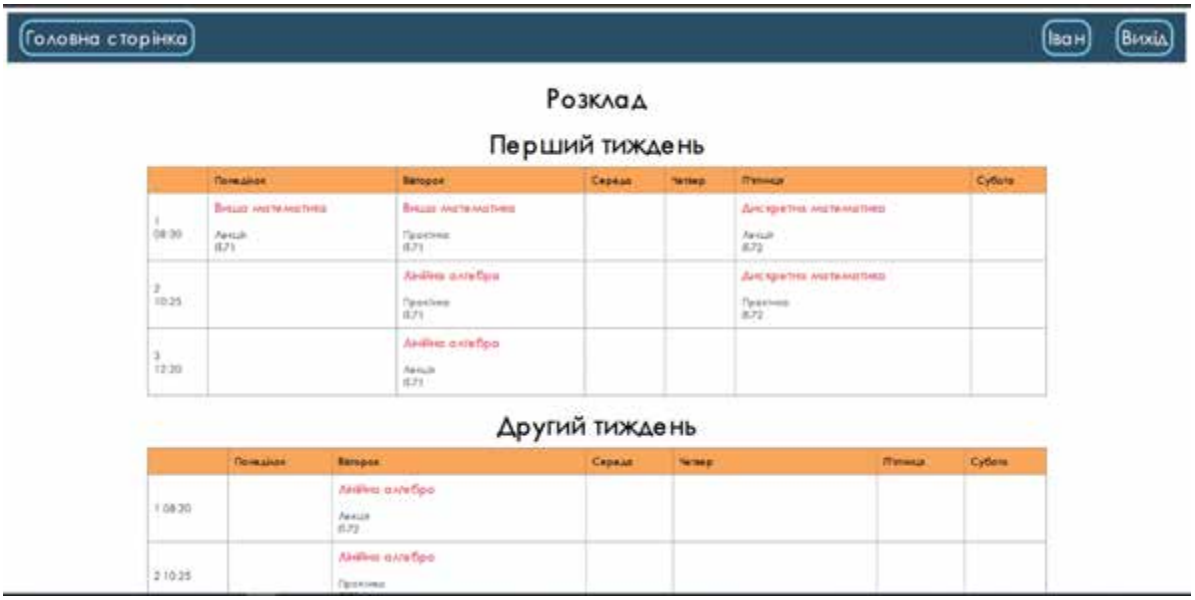

Рисунок 4.8 – розклад для викладача

Коли юзер клікає на якесь заняття в розкладі, то він потрапляє на сторінку цього заняття. Детальніше про цю сторінку йтиметься нижче. Також список занять можна побачити на сторінці «Мої заняття».

На сторінці із заняттями викладач може бачити список всіх занять із назвою, та назвою групи. У студента теж є така можливість, тільки от назви групи не буде відображатися. На рисунку 4.9 відображено сторінку із заняттями для викладача.

Рисунок 4.10 – сторінка із заняттями викладача

На сторінці із конкретним заняттям викладач та студент бачать щоденник, куди викладач може проставляти бали, відвідування та залишати коментарі до завдань. Щоб зробити це викладач повинен кліцнути на конкретну клітинку розкладу, після чого відкриться діалогове вікно, де можна це зробити. Також викладач може лишити опис про предмет (як заробити бали, посилання на пару), студент може переглядати цю інформацію. Це можна побачити на рисунку 4.11.

Рисунок 4.11 – сторінка із конкретним заняттям

Також викладач може додати до сайту файли, а студент може ці файли завантажити.

Отже, проект пройшов перевірку якості, що дає надію на хороший досвід користування юзерами. Швидкодія та інтуїтивність на високому рівні.

наліз отриманих даних

– засіб, який дозволяє створювати, керувати та розповсюджувати звіти і друковані матеріали в екосистемі продуктів Microsoft SQL Server. SSRS допомагає підприємствам та організаціям створювати різноманітні звіти для аналізу даних та інформування користувачів. Ось деякі ключові можливості та інструменти

Однією з основних особливостей SSRS є можливість гнучкого дизайну звітів. Завдяки візуальному конструктору звітів та мові розмітки RDL (Report графіками, діаграмами та іншими візуальними компонентами. Ця гнучкість дозволяє адаптувати звіти до конкретних потреб і надавати чіткий та зрозумілий огляд даних.

Система підтримує параметри та фільтри, що дає можливість користувачам взаємодіяти з звітами динамічно, вибираючи дані для перегляду та налаштовуючи їх фільтрацію. Такий підхід допомагає точно настроювати аналіз даних і надає користувачам можливість отримати глибше розуміння інформації.

Ефективність є ключовою характеристикою SSRS. Система дозволяє налаштовувати автоматичну доставку звітів для різних одержувачів, що забезпечує своєчасне отримання оновлених даних. Звіти можуть автоматично надсилатися по електронній пошті, зберігатися на файлових серверах або публікуватися в різних форматах. Автоматизація спрощує процес розподілу даних та робить інформацію легко доступною.

Крім того, SSRS безперешкодно інтегрується з іншими продуктами Microsoft, такими як SQL Server, SharePoint, Azure та Power BI. Ця інтеграція

підсилює можливості системи і дозволяє користувачам створювати розширені аналітичні та звітні рішення. Ця взаємодія дозволяє підприємствам використовувати існуючий стек технологій для покращення аналітичних можливостей.

Безпека є важливою для SSRS. Система має потужну модель безпеки, яка надає контроль над доступом до звітів та даних, забезпечуючи конфіденційність і цілісність інформації.

Адміністратор може використати цей сервіс для того, щоб будувати певні звіти. Прикладом такого звіту може бути статистична діаграма, яка показує середній бал студентів в залежності від регіону. Це можна побачити на рисунку

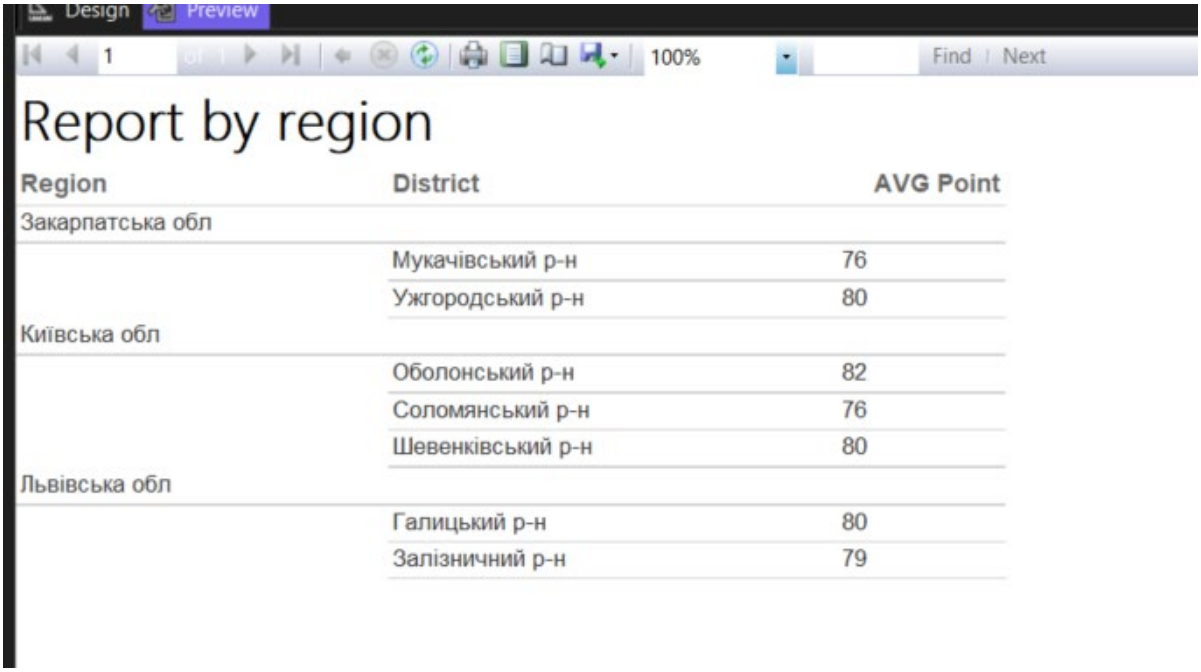

Рисунок 4.12 – середній бал за регіоном

Як видно на рисунку 4.12 можна побачити, що найвищий середній бал є в Києві, а саме в Оболонському районі. Найнижчий же середній бал є в Солом’янському районі міста Києва та Мукачівському районі Закарпатської області. Таким чином адміністратор може отримати деяку інформацію та статистику по навчальному процесу і прийняти якісь управлінські рішення.

Також додатковим прикладом використання сховища даних та кубу є отримання даних по кількості відмінників в залежності від статі та року навчання. Це можна побачити на рисунку 4.13.

	Excelent count	
	Male	Female
2020	348	450
2021	366	455
2022	312	416

Рисунок 4.13 – статистика по відмінниках в залежності від статі та року

Таким чином адміністратор може отримувати дані для аналізу. Також можна використовувати Power BI для аналізу.

ВИСНОВКИ

Отже, проаналізувавши предметну область, було виявлено, що на даний момент існує безліч різних систем, які допомагають організувати дистанційне навчання. Багато популярних систем зроблені такими іменитими корпораціями, як Microsoft або Google.

Не зважаючи на різноманітність продукції, в кожної із таких систем є свої недоліки. Тому було розроблено власну систему, яка надає весь необхідний для дистанційного навчання функціонал, та при цьому не є перенавантаженою. Також система проста в користуванні та інтуїтивна, тому будь який користувач зможе розібратися з нею. Універсальність даного застосунку дозволяє його використання широким колом навчальних закладів.

При розробці даного додатку багато уваги приділялося інтерфейсу користувача. Інтерфейс зроблено простим та інформативним. Всі дії на сайті виконуються інтуїтивно, тож програмою зручно користуватися. Підібрана палітра кольорів, яка є гармонічною і приємною для очей користувача.

Програма працює швидко і не змушує користувача довго чекати на відповідь. У системі передбачена функція отримання сповіщень в месенджер

Застосунок складається із шести компонентів. Перший компонент - це сервіс БД. Він слугує для збереження даних. Із цим компонентом взаємодіють три інші компоненти: сервіс відправки сповіщень, API-сервіс та сховище даних. Сервіс відправки сповіщень постійно дістає із БД дані про сповіщення та відправляє їх у месенджер Telegram. API-сервіс обробляє інформацію із БД та надає доступ до неї по HTTP-протоколу. Також є компонент, який відповідає за кешування даних, щоб пришвидшити роботу. Останнім компонентом в системі є компонент самого вебзастосунку. Він призначений для безпосередньої взаємодії з користувачем та отримує дані із API-сервісу.

Розробка програми велася за допомогою найновіших і суб’єктивно найкращих на даний момент засобів розробки програмного забезпечення. Всі функції системи було протестовано, ніяких помилок в роботі не було виявлено.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

и

с

т

ЄORM. URL: https://uk.wikipedia.org/wiki/SCORM

млатформ для організації дистанційного навчання. URL: https://buki.com.ua/news/5-platform-dlya-orhanizatsiyi-dystantsiynoho-navchannya/

yftware Architecture. URL: https://www.sei.cmu.edu/our-work/software-architecture/

pb Application Architecture: The Latest Guide 2022. URL: https://www.clickittech.com/devops/web-application-architecture/

6. Bertrand Mayer A full online version of Object-Oriented Software Construction, 2nd edition (1997): навчальний посібник, Prentice-Hall, Inc. D
7. Neal Ford, Mark Richards Fundamentals of Software Architecture: An Engineering Approach: навчальний посібник, O'Reilly Media, 2020, 419 с.
8. Russ Unger and Carolyn Chandler A Project Guide to UX Design: навчальний посібник, New Riders, USA, 2009, 267 с.
9. Петрик М. Р. Моделювання програмного забезпечення: науково-методичний посібник / М.Р. Петрик, О.Ю. Петрик – Тернопіль: Вид-во ТНТУ імені Івана Пулюя, 2015, 200 с.
10. Як будувати UML-діаграми. Розбираємо три найпопулярніші варіанти. URL: https://dou.ua/forums/topic/40575/
11. Створення схеми діаграми станів UML. URL: https://support.microsoft.com/uk-ua/office/створення-схеми-діаграми-станів-uml-2e46fd66-e861-4e8c-9188-36255395ebf3

і

м

Ы

R

h

Діаграма послідовності (Sequence Diagrams). URL: https://www.maxzosim.com/sequence-diagrams/

Rehman, Christopher Paul, Christopher R. Paul. The Linux Development Platform: Configuring, Using and Maintaining a Complete Programming Environment. 2002, 350 с.

14. Visual Studio 2022 IDE – Programming Tool For Software Development. URL: https://visualstudio.microsoft.com/vs/
15. Огляд онлайнової аналітичної обробки. URL: https://support.microsoft.com/uk-ua/office/огляд-онлайнової-аналітичної-обробки-olap-15d2cdde-f70b-4277-b009-ed732b75fdd6

Поняття сховищ даних та основи їх створення. URL: https://studfile.net/preview/7144845/page:29/

Технологія сховищ даних Data Warehousing. URL: https://studfile.net/preview/5118185/page:31/