math_2024_050

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРНІВЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ЮРІЯ ФЕДЬКОВИЧА

Факультет математики та інформатики
Кафедра математичного моделювання

Цифрова трансформація університету

Кваліфікаційна робота

Рівень вищої освіти – перший (бакалаврський)

Виконав:
студент 4 курсу, 407 групи
Бердник Олександр Вікторович
Керівник:
кандидат фізико-математичних наук,
доцент Горбатенко М.Ю.

До захисту допущено
на засіданні кафедри
протокол № __ від _____ 2024 р.
Зав. кафедрою _____ проф. Черевко І.М.

Чернівці – 2024

Анотація

У роботі розглянуто проблеми з якими стикаються учасники навчального процесу під час дистанційної форми навчання, наявні рішення, якими ті користуються задля отримання знань. Розглянуто стан цифрової трансформації України, визначено сильні та слабкі сторони.

Метою роботи є створення системи онлайн розкладу, а саме інтерфейсу прикладного програмування та вебзастосунку відображення розкладу для студентів та викладачів.

Ключові слова: цифрова трансформація, вебзастосунок, вебсервер, прикладний програмний інтерфейс, чиста архітектура, шаблон, база даних.

Abstract

The paper examines the problems faced by participants in the educational process during distance learning, the existing solutions that they use to obtain knowledge. The state of digital transformation in Ukraine is analyzed, and strengths and weaknesses are identified.

The aim of the work is to create an online timetable system, namely an application programming interface and a web application for displaying timetables for students and teachers.

Keywords: digital transformation, web application, web server, application programming interface, clean architecture, template, database.

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів наукових досліджень інших авторів мають посилання на відповідне джерело.

_____ О.В. Бердник

Зміст

Вступ

Цифрова трансформація в освітньому секторі означає вдосконалення навчального середовища та системи управління шляхом застосування нових технологій. Це дозволяє скоротити витрати та задовольнити щораз вищі потреби всіх учасників навчального процесу. Усі навчальні заклади стикаються з такими загальними проблемами, як трудомісткі процедури приймання студентів, реєстрація на програми та курси, планування навчального плану, складання розкладу, розподіл викладачів тощо. Усі ці проблеми зараз ефективно вирішуються впровадженням різноманітних програмних рішень, спрямованих на скорочення часу, оптимізацію трудових та матеріальних витрат на адміністрування.

Одним з напрямків трансформації є доступність розкладу занять з можливістю вносити зміни залежно від форми навчального процесу: очна чи змішана в цей момент часу, та можливістю оперативного інформування учасників навчального процесу.

Зважаючи на освітні потреби метою дипломної роботи було створення системи онлайн розкладу, яка містить зручний розклад як для студентів університету, так і для викладачів. Нова система розроблена на заміну застосунку що існує для факультету математики та інформатики. Чинна версія має ряд недоліків, пов’язаних зі швидкістю роботи та масштабованістю на увесь університет. Вимоги до нової системи наступні: швидкодія, масштабованість, гарний дизайн.

На основі встановленої мети було визначено такі блоки з яких складатиметься система:

- мобільний застосунок;
- вебзастосунок для відображення розкладу;
- вебзастосунок для створення та модифікації розкладу;
- прикладний програмний інтерфейс(API), який надає можливість взаємодіяти з вебсервером; вебсервер надає різні функції через API, такі як доступ до бази даних, зміна даних тощо.

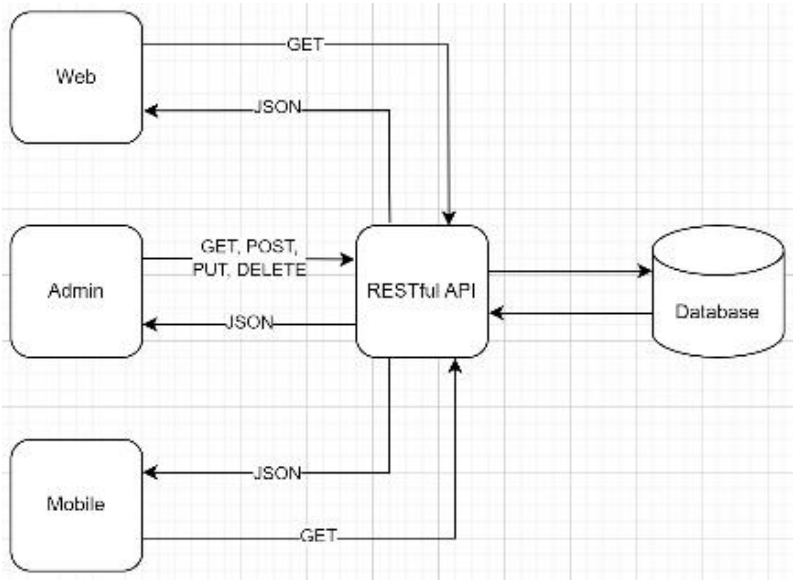

Рисунок 1. Схема системи

Важливим нюансом є те, що запити на зміну, додавання або видалення даних можуть робити тільки авторизовані користувачі з відповідними правами з панелі адміністратора. Застосунок відображення розкладу та мобільний застосунок можуть робити тільки запити на отримання даних та відображати їх. У відповідь на запити, вебсервер повертає файл json.

У системі передбачені такі ролі авторизованих користувачів:

- студент, має доступ до персоналізованого розкладу;
- адміністратор розкладу, може створювати розклад, не може редагувати інформацію про аудиторії або освітні програми;
- адміністратор факультету, може вводити інформацію за освітніми програмами та аудиторіями, може складати розклад;
- адміністратор, немає обмежень у доступі.

Мій внесок у реалізацію даної системи є створення прикладного програмного інтерфейсу та вебзастосунку відображення розкладу.

Результатом роботи є інтерфейс, що дозволяє вносити дані про університети, факультети, групи та розклад для них тощо, та застосунок, що дозволяє зручно та швидко переглядати розклад викладачам та студентам.

У технічному аспекті вибір технологій, та шаблонів реалізації відбувались з метою створити проєкт з можливістю легкої масштабованості та довгострокової підтримки.

Публікації:

- студентської наукової конференції Чернівецького національного університету імені Юрія Федьковича 2024;
- VII Міжнародна науково-практична конференція “Professional development: theoretical basis and innovative technologies”;
- V міжнародна науково-практична конференція “Інформаційні моделюючі технології, система ти комплекси”(ІМТСК-2024).

Участь у наукових конференціях:

- студентська наукова конференція Чернівецького національного університету імені Юрія Федьковича 2024;
- V міжнародна науково-практична конференція “Інформаційні моделюючі технології, система ти комплекси”(ІМТСК-2024).

Розділ 1: Сучасні тренди

1.1 Наявні рішення

Термін “цифрова трансформація” широко використовували та границі використання були розмиті. Однак спалах COVID-19 та його поширення привернули увагу до необхідності цифрової трансформації, оскільки майже всі навчальні заклади були змушені застосовувати її тією чи іншою мірою. Перейшовши на дистанційне навчання, учасники освітнього процесу були вимушені активно використовувати різні навчальні платформи, інтерактивні онлайн дошки та сервіси відеозв’язку.

До найпопулярніших платформ дистанційного навчання в Україні можемо віднести Moodle та Classroom. В системі Moodle, викладачі можуть створювати авторські курси, розміщувати матеріали, до яких відносяться лекції, презентації, книги, довідники у різних форматах, включаючи .pdf, .doc, .html тощо, а також відео, аудіо та іншу інформацію. Також є можливість здійснювати тестування. Ще однією перевагою системи є те, що викладачі можуть вести облік студентів з можливістю розмежування прав на доступ до навчальних матеріалів. Студенти можуть надсилати виконані завдання, переглядати журнал оцінок, де перелічено всі створені викладачем завдання та оцінки за них, та створити особистий чат з викладачем. Загалом, функціонал платформи більш ніж достатній для того, щоб студенти отримали максимум користі від навчання.

Навчальне середовище Google Classroom є простішим у використанні через мінімалістичний інтерфейс. Має можливість інтеграції із більшістю застосунків Google, в той час, як Moodle має певні обмеження. Недоліком платформи можна вважати труднощі управління курсом при великій кількості учасників.

Не зважаючи на можливості покращення платформ, вони добре виконують свою основну функцію. Moodle є дещо досконалішим рішенням, але в той самий час важчим для опанування.

Найпоширенішими сервісами відеозв'язку є Google Meet та Zoom, які дають змогу проводити пари в режимі онлайн. Однією з переваг Google Meet є те, що він працює у веббраузерах і не потребує встановлення додаткового програмного забезпечення. Наступним плюсом є сумісність з Google-календарем, викладач створює подію і всі учасники зможуть долічитися до відеозустрічі за посиланням, яке їм автоматично приходить на пошту. Кількість користувачів у відеозустрічі Meet – до 250 осіб, в той час, як у Zoom – до 100.

Перевагою Zoom є можливість створення окремих кімнат для переговорів у рамках однієї зустрічі, що дає викладачу можливість організувати більш інтерактивне та продуктивне заняття. Корисною функцією також є можливість вимкнути мікрофони усім учасникам зустрічі залишаючи за ними можливість вмикати мікрофон за потреби. Демонстрація екрана в Zoom дещо розширена, викладач може демонструвати всім учасникам і свій екран і Whiteboard – інтерактивну дошку на якій можна разом писати, малювати. Також є можливість заборонити учасникам демонструвати свої екрани у налаштуваннях сесії.

Таким чином, Google Meet є простішим й полегшує етап інформування всіх учасників про відеозустріч, в той час, як Zoom має дещо більший функціонал для самих зустрічей. Проте найбільшою проблемою другого є те, що у безплатній версії тривалість дзвінку до 40 хвилин.

Найбільше проблем виникає при пошуку зручної та функціональної інтерактивної онлайн дошки. Більшість дошок мають недостатній функціонал для комфортної роботи з ними. Окрім того, однією з проблем, які можна відмітити у деяких дошках – це те, що створена дошка, видаляється після того, як всі учасники її покинули.

Однією з популярних інтерактивних онлайн дошок є Ziteboard – масштабована віртуальна дошка, яка працює через веббраузери. Викладач може створити дошку та поділитись посиланням на неї, яке буде постійним. Дошка дозволяє малювати фігури, вводити текст, створювати помітки та todo-

листи. Також, перевагою дошки є те, що її можна запустити на різних пристроях включаючи смартфони та планшети.

Ще однією популярною дошкою є Google Jamboard, яка, як і попередня дошка, дозволяє взаємодіяти викладачу з групою в режимі реального часу. Дошка легко інтегрується з Google Classroom. Недоліком дошки є робоча поверхня, яка складається зі сторінок-слайдів фіксованого розміру, яких може бути максимально 20 одиниць, у межах одного документу.

1.2 Цифрова трансформація України в період пандемії

За формування та реалізацію державної політики у сфері цифровізації відповідальний центральний орган виконавчої влади – Міністерство цифрової трансформації України. Міністерство утворилось шляхом перетворення Міністерства агентства з питань електронного урядування України.

Одним з яскравих прикладів цифрової трансформації в Україні під час пандемії було започаткування платформи “Всеукраїнська школа онлайн” [1]. Завдяки платформі, школярі отримали вільний доступ до навчальних матеріалів з таких дисциплін, як математика, історія України, Українська мова та багато інших. Щоб почати навчання, учню достатньо перейти на офіційний сайт платформи [2] або завантажити мобільний застосунок, вибрати клас та предмет.

У державній стратегії регіонального розвитку на 2021-2027 роки, що набула чинності 14 вересня 2020 року [3], зазначено про низький, на той момент, рівень цифровізації регіонів та цифрової обізнаності. Були встановлені конкретні кроки покращення цифрової грамотності населення. Одним з таких кроків є створення проєкту єдиного державного порталу цифрової освіти - “Дія. Цифрова освіта” [4]. Освітня платформа налічує велику кількість відео та аудіо матеріалів для людей різних професій та віку. Після проходження освітнього серіалу, користувач отримує сертифікат, який є офіційним підтвердженням завершення серіалу. Окрім освітніх матеріалів, на платформі представлені подкасти, вебінари, гайди, симулятори та тести. Щоб почати навчання на платформі, потрібно тільки авторизуватись на сайті. Після

авторизації користувачу доступні для проходження всі навчальні матеріали сайту. Є можливість проходити декілька серіалів паралельно.

1.3 Стан цифрової трансформації України

Згідно з результатами цифрової трансформації в регіонах України за 2023 рік [5] середній показник власноруч розробленого індексу цифрової трансформації становить 0.632 бала з 1 можливого, що наштовхує на висновки про високу, але не достатню цифровізацію країни. Крім того, порівнюючи зі звітом 2022 року, індекс цифрової трансформації став меншим.

Станом на 2023 рік, попри повномасштабне вторгнення, Україна є одним з найбільших експортерів IT-послуг в Європі, експортний виторг у сфері IT становив 6,7-7.1 млрд доларів та залишається єдиною сферою в Україні, яка наразі повноцінно працює [6].

Global Innovation Index – це щорічне видання, яке оцінює та порівнює інноваційні можливості країн світу. При оцінці, індекс використовує близько 80 показників, серед яких вимірювання інвестицій в науку та інновації, освіти, інфраструктури, людського капіталу.

З 2020 року по 2022 рік спостерігається спадання рейтингу України. Так, у 2020 році Україна займала 45 місце у загальному рейтингу та була на другому місці у рейтингу у групі країн з доходом нижче середнього, а станом на 2022 рік 57 місце в загальному рейтингу та 4 в групі країн з доходом нижче середнього. Варто зазначити, що у 2023 році – році останнього видання індексу, країна піднялась в рейтингу на 55 місце та третє у рейтингу у групі країн з доходом нижче середнього [7].

1.4 Світові тренди

Беззаперечно, цифровізація торкається всіх сфер життя, від взаємодії між людьми, до промислових виробництв. Відтак, згідно з дослідженням Digital 2024: Global Overview Report 69,4% населення світу на початку 2024 року користуються мобільними пристроями [8]. Зрозуміло, що з часом відсоток буде тільки зростати. Інтернет речей – система фізичних пристроїв, що взаємодіють між собою через мережу, міститиме дедалі більше пристроїв.

Цифрові технології підвищують продуктивність праці, добробут населення, покращують рівень освіти. Однак, варто зазначити, що цифровізація породжує проблеми, основними з яких є:

- ризик втрати конфіденційності;
- ризик поширення неточної інформації та нестача прозорості;
- екологічні проблеми;
- соціальна віддаль.

Паралельно до розповсюдження цифрових технологій, люди пробують розв'язати наявні проблеми.

Цифрові тренди – це напрямок розвитку цифрових технологій. Основними світовими трендами станом на 2024 рік є:

- генеративний штучний інтелект;
- квантові обчислення;
- імерсивні технології;
- цифрові платформи.

Таким чином, цифровізація є одним із головних рушіїв зростання світової економіки в найближчі роки. Від цифрових технологій, крім прямого підвищення продуктивності, яке отримують компанії, навчальні заклади та урядові установи є багато непрямих переваг, як-от економія часу, створення нового попиту на нові товари й послуги, нова якість та цінність тощо.

Розділ 2: Вибір технологій

2.1 База даних

У сучасному світі важко уявити застосунок, якому б не довелось взаємодіяти зі збереженими даними. Дані можуть бути представлені у числовому, текстовому, булевому форматі тощо. Створювана система не виключення. Виникає питання де та як зберігати дані. База даних – організована структура, яка призначена для зберігання, зміни та обробки інформації. Зазвичай бази даних використовують коли є потреба зберігати великий обсяг інформації. Перевагою баз даних є те, що у них можна зручно працювати з даними, а саме зберігати, змінювати їх, а також швидко знаходити потрібні дані. Завдяки спеціальним алгоритмам пошук необхідних даних відбувається дуже швидко.

Сьогодні існує багато різних типів баз даних, окрім того, можна зберігати дані у текстових файлах. У таких файлах для поділу полів застосовують різні розділові знаки, це може бути пробіл, кома, крапка з комою та двокрапка. Однак такі бази підходять не для всіх типів інформації й коли даних дуже багато, операції пошуку, видалення або додавання нових даних є трудомісткими та займають багато часу.

Ієрархічна база даних надає можливість задавати зв’язки типу “предок-нащадок”. Такий тип бази даних можна зобразити як дерево, що складається з об’єктів різних рівнів. У такій базі даних кожен об’єкт може мати декілька нащадків і тільки одного предка. Можна одразу помітити недолік у вигляді повільного доступу до сегментів даних, що знаходяться на нижніх рівнях ієрархії. Також, проблемою є складність оновлення записів. Окрім того, така організація даних не передбачає зв’язку “багато-до-багатьох”, що створює обмеження при зберіганні даних.

Мережеві бази даних розширюють функціональність ієрархічних, тут, на відміну від ієрархічної моделі даних, кожен об’єкт може мати будь-яке число предків. Такий тип бази даних можна зобразити як граф. Таким чином,

мережева база даних дає більшу гнучкість моделювання даних. Проте обмеження ієрархічного підходу зберігаються.

У реляційних базах даних безпосередньо дані організовані у вигляді набору формально описаних таблиць. Всі операції над даними зводяться до операцій над таблицями. Відносини між об’єктами моделюються за допомогою зовнішніх ключів, тобто посиланнями на інші таблиці. Дана організація даних зумовила створення нормалізації. Нормалізація – це процес розбиття однієї великої таблиці на декілька менших, кожна з яких містить один вид інформації, що дозволяє мінімізувати надлишковість даних. Перевагами реляційних баз даних є зручність використання, швидкий доступ до даних, масштабованість. Таким чином, реляційна база даних є найкращим вибором для реалізації дипломної роботи.

Наступним питанням, яке потрібно було вирішити – це яку систему керування реляційними базами даних вибрати. До популярних систем керування реляційними базами даних належать MySQL, Microsoft SQL Server, PostgreSQL тощо. Більшість систем є подібними, вони використовують структуровану мову запитів(SQL) як інтерфейс для роботи з даними, мають функції резервного копіювання даних, контролю доступу. Наприклад, порівнюючи MySQL з PostgreSQL, в той час, як перша суто реляційна база даних, друга є об’єктно-реляційною, що дає змогу зберігати в неї об’єкти з властивостями. PostgreSQL також підтримує інші додаткові типи даних, такі як масиви, XML та бінарний JSON. MySQL підтримує B-tree і R-tree індексування, яке зберігає ієрархічно проіндексовані дані в той час, як PostgreSQL підтримує дерева, індекси виразів, часткові індекси та хеш-індекси. Є більше варіантів для точного налаштування вимог до продуктивності бази даних у міру масштабування. Microsoft SQL Server є комерційною пропозицією, містить у собі інструменти бізнес-аналітики для аналізу даних і звітності. На противагу Microsoft SQL Server, PostgreSQL підтримує кращу паралельність. Таким чином, вибір зроблено на користь

PostgreSQL, яка має хорошу продуктивність, великий перелік підтримуваних даних та без надлишковості для дипломної роботи.

2.2 Вебсервер та API

При виборі технології для створення вебсервера та прикладного програмного інтерфейсу або ж web API вибір пав на .NET версії 7.0, останньої версії на момент початку розроблення системи, з використанням мови програмування C#. Технологія має багато переваг:

- автоматичне збирання сміття: зменшує ймовірність виникнення помилки та спрощує розробку застосунку;
- реалізація моделі асинхронного виконання коду: дозволяє легко писати код з вищою продуктивністю;
- мультиплатформність: .NET застосунки можна запускати на різних операційних системах, включаючи Windows, Linux, macOS;
- широкий спектр інструментів: платформа має велику кількість інструментів та бібліотек для розробки, тестування та управління програмним забезпеченням;
- активна підтримка корпорацією Microsoft.

C# було вибрано, через те, що вона є об’єктно-орієнтованою мовою програмування на відміну від F#, а також стрімко розвивається.

Прямим конкурентом C# є Java. Поява даних мов програмування пов’язана з переходом від низькорівневих мов програмування до вищого рівня. Дані мови програмування мають подібності, до них відносяться безпека типів, збирання сміття, одинарне наслідування між класами. У порівнянні з Java, C# виграє завдяки своїй ефективності та легкості використання, наявністю вбудованої мови запитів LINQ та зручної імплементації асинхронності завдяки ключовим словам async/await.

На PHP створена велика частина сайтів, проте PHP є інтерпретованою мовою, саме тому PHP швидке працює з малими та середніми проєктами, в той час, як .NET має хорошу продуктивність з проєктами будь-якої величини. Ще однією перевагою .NET над PHP є те, що перша має кращий рівень безпеки.

Багато функцій захисту одразу включені до пакета під час розробки застосунку. У разі появи проблем з захистом програма припиняє свою роботу та видає попередження про проблему.

Python являється однією з найпопулярніших мов програмування сьогодення та має фреймворк для створення вебзастосунків. Через популярність мови, фреймворк активно підтримується. Проте динамічна типізація стає перешкодою для виявлення проблем, працює повільніше ніж конкуренти у розробці вебзастосунків.

Тож, у нашому випадку Python, PHP та Java не підходять для реалізації вебсервера та web API.

2.3 Вебзастосунок відображення розкладу

Мета застосунку – це відображення розкладу, саме тому немає потреби використовувати важкі та перенасичені на різні інструменти, як для такого завдання, технології. Натомість можна використати вебфреймворки, такі як Next.js, Vue.js, Angular.js. Загалом, вебфреймворків є велика кількість. Всі вони, так чи інакше підходять для створення потрібного застосунку. При виборі фреймворку я керувався наступними запитаннями:

1. наскільки стабільним є фреймворк;
2. наскільки хорошою є документація;
3. чи підтримується TypeScript;
4. наскільки швидко працюють застосунки розроблені на даному фреймворку;
5. наскільки складно масштабується;

Vue.js є JavaScript-фреймворком, що має такі переваги:

1. простий для вивчення через детальну документацію;
2. простий процес інтеграції з іншими бекенд-фреймворками;
3. висока продуктивність.

Основний аспект фреймворку – це його простота, проте в його простоті криється й головний його недолік, не знаючи кращих практик структурування файлів, масштабування застосунку і багатьох інших нюансів, можна створити

застосунок зі складною структурою яку буде важко підтримувати та масштабувати.

Angular – це фронтенд-фреймворк з відкритим вихідним кодом. Написаний фреймворк мовою програмування TypeScript, саме тому розробка застосунків з допомогою Angular здійснюється з використанням цієї мови. Це є перевагою, оскільки додає продуктивності розробникам, які працюють з такими мовами програмування як C#, або Java. Він також має хорошу документацію та чудово масштабується. Проте, через свою “вагу” фреймворк інколи може видавати низьку швидкість роботи. Angular спрямований на великомасштабні проєкти.

Next.js – це фреймворк створений на основі React, JavaScript-бібліотеки для створення інтерфейсів користувача. Фреймворк надає широкий набір інструментів для створення вебзастосунків.

Однією з ключових особливостей Next.js є підтримка серверного рендерингу(SSR). Даний механізм дозволяє генерувати всю сторінку на сервері під час першого запиту. Це покращує продуктивність застосунків та підвищує SEO-оптимізацію. Також є підтримка статичної генерації, що дозволяє згенерувати сторінку під час побудови застосунку і дану, вже готову, сторінку сервер буде повертати клієнту щоразу при запиті. Таким чином, застосунки, що не потребують частого оновлення, з використанням даного типу генерації сторінок будуть працювати швидше. Вбудована маршрутизація полегшує механізм створення маршрутів. Створивши папку з потрібною назвою й у тій папці файл page.tsx, дана сторінка доступна, адреса сторінки відповідає назві папки. Також фреймворк має вбудовану обробку зображень, це забезпечує швидше завантаження зображень, та сайту в цілому. Окрім переліченого вище, фреймворк має високий рівень безпеки. Таким чином, даний фреймворк підходить для проєктів будь-яких розмірів.

Фреймворк підтримує як JavaScript, так і TypeScript. Незважаючи на те, що фреймворк молодший за конкурентів, застосунки написані з допомогою Next.js працюють дуже швидко і в пошуковий видачі займають високі позиції.

Опираючись на тренди npm – тренди завантажування фреймворку, та тренди stack overflow – відсоток питань на місяць за різними технологіями, бачимо, що статистика next.js стабільно зростає з кожним роком, а отже, фреймворк має популярність серед розробників. Тож, можемо зробити висновок, що хвилюватись про занепад Next.js в найближчі роки не варто.

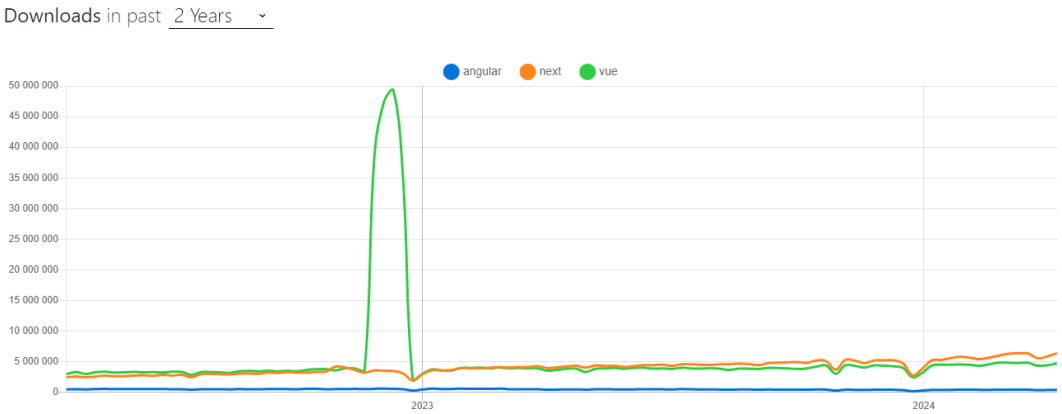

Рисунок 2. Статистика завантажувань фреймворків за останні 3 роки

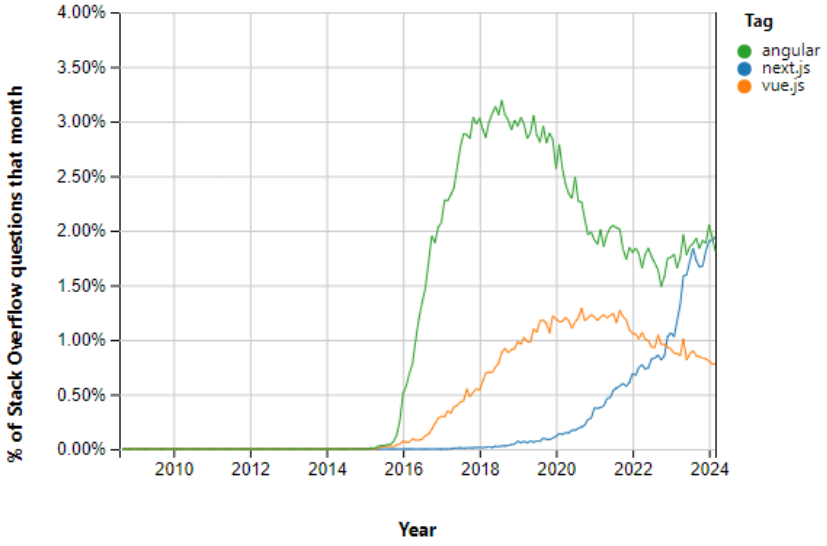

Рисунок 3. Популярність технологій на основі кількості питань на платформі stack overflow

Розділ 3: Реалізація

3.1 Проєктування бази даних

Як було вказано раніше, база даних використовується реляційна, отже потрібно працювати з таблицями. Для роботи з базою даних було вибрано структуру об'єктно-реляційного відображення(ORM) Entity Framework, структура, що дозволяє проєктувати класи з застосунку в таблиці в базі даних. Основні поняття, які потрібно розуміти під час роботи з ORM:

1. DbContext – основний клас, відповідає за взаємодію з базою даних;
2. DbSet – представляє колекцію всіх об'єктів певного типу що будуть записані в базу даних;
3. Entity або сутність – клас, який представляє таблицю в базі даних.

Таким чином, для роботи з базою даних були створені відповідні класи-сутності, створений клас, що відповідає за взаємодію з базою даних, шляхом наслідування від класу DbContext й у цей клас, з допомогою DbSet, були додані всі класи-сутності.

ORM має два підходи до способу створення моделей є database first та code first. Згідно з database first, робота починається зі створення бази даних з усіма таблицями, зв'язками між ними. Ми створюємо сутності та клас DbContext на основі наявної бази даних. Інший підхід полягає у тому, щоб спочатку створити в застосунку сутності, а потім використати їх для створення бази даних. Я використав другий підхід. Таким чином я зміг зосередитись на розробці моделей домену, а фреймворк виконав всю роботу по створенні бази даних з потрібними таблицями та зв'язками між ними. Щоб виконати команду оновлення бази даних потрібно застосовувати міграції – механізм оновлення схеми бази даних, щоб база даних і моделі були синхронізовані. Щоб створити базу даних, або змінити її схему потрібно в консоль увести наступні команди: Add-Migration <Name> та Update-Database. Після першої команди у застосунку створиться окремий клас з описом сутностей, зв'язків та типів даних до кожної колонки. Друга команда застосує ці зміни до бази даних. Даний механізм застосовувався неодноразово, коли були потреби змінити типи значень які

можуть містити колонки, назви колонок тощо. Всі класи міграцій зберігаються в коді застосунку, таким чином, при запуску системи на новій платформі всі міграції можна застосувати та відновити потрібний стан бази даних. В проєкті створений окремий клас, який відповідає за автоматичну активацію всіх міграцій. Це дозволяє легко, запустити застосунок на віртуальній машині без ручного створення бази даних та застосування міграцій, все відбувається автоматично.

В системі працюємо з такими основними сутностями:

- університет – освітній заклад, який здійснює підготовку спеціалістів.
- факультет – адміністративна одиниця університету;
- аудиторія – місце, де відбувається заняття;
- кафедра – структурний підрозділ університету, що займається проведенням навчальних та наукових робіт у певній галузі знань;
- освітня програма – комплекс освітніх компонентів, що сплановані закладом освіти для досягнення студентами результатів навчання;
- дисципліна – окремий предмет або курс, може бути обов’язковою або вибірковою;
- студент – здобувач освіти;
- диплом студента – документ, що засвідчує закінчення студентом навчального закладу за певною спеціальністю;
- викладач – особа, що проводить навчальну та наукову діяльність;
- група – студенти що навчаються за однією освітньою програмою;
- заняття – форма організації навчального процесу, типи: лекція, практична, семінарська, лабораторна.

Метою процесу нормалізації є мінімізація дублювання даних, що дозволяє зменшити ризик втрати даних або виникнення неузгодженостей, забезпечення структурованості, що робить базу даних зрозумілою і легкою у використанні. Саме мінімізація даних, оскільки при усуненні надлишковості повністю, базу даних неможливо буде підтримувати як одне ціле. Саме з цієї

причини база даних проєктувалась з дотриманням вимог перших чотирьох нормальних форм. Згідно з першою нормалізацією, кожна таблиця містить первинний ключ, кожен атрибут містить неподільні значення та відсутні групи атрибутів з однаковими за змістом значеннями у межах одного рядку. Згідно з другою нормальною формою, має бути перша нормальна форма та кожен неключовий атрибут таблиці не приводимо залежить від первинного ключа таблиці. Згідно з третьою нормальною формою, повинна бути друга нормальна форма та кожен неключовий атрибут нетранзитивно залежить від кожного потенційного ключа. Нормальна форма Бойса-Кодда є розширеною версією третьої нормальної форми й полягає в наступному: не існує атрибутів які залежать від часткової складової ключа. Четверта нормальна форма – наступний рівень після нормальної форми Бойса-Кодда. Таблиця перебуває в четвертій нормальній формі, якщо вона не має багатозначної залежності. При проєктуванні застосовані такі зв’язки між таблицями: один до одного, один до багатьох, багато до багатьох.

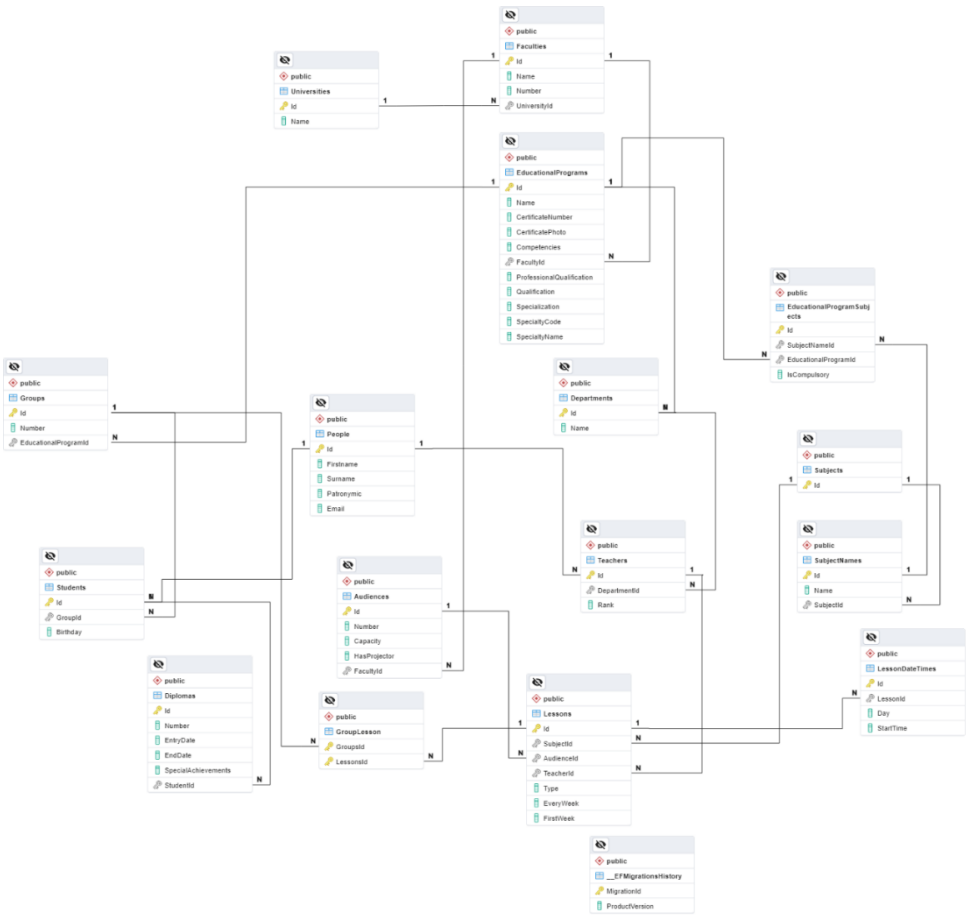

Рисунок 4. Діаграма бази даних.

3.2 Створення вебсервера

Вебсервер, як основа створюваної системи, буде постійно зазнавати змін, а отже, потрібно створити такий застосунок, який можна легко масштабувати, тестувати та підтримувати. Архітектура програмного забезпечення – це заходи, спрямовані на те, щоб чітко визначити структуру системи. Простіше кажучи, це підхід, який визначає які функції за що відповідають та як вони взаємодіють між собою. Основна ціль архітектури полягає у тому, щоб створити структуру, яка буде відповідати вимогам проєкту та забезпечуватиме оптимальну працездатність, надійність та масштабованість. Окрім того, створення застосунку без визначеної наперед архітектури може призвести до наступних проблем:

1. труднощі розвитку та підтримки: відсутність архітектури спричинить безлад у програмному коді, як наслідок, буде важко внести зміни без порушення функціональності;
2. погана масштабованість: можливий варіант створення системи, яка не здатна легко масштабуватися, через велику зв’язність коду;
3. низька якість коду: оскільки немає чіткого поділу який модуль за що відповідає, написаний код може бути складним для розуміння та модифікації;
4. низька продуктивність: через неефективне використання ресурсів або неоптимальну організацію процесів;
5. підвищена вразливість: система може мати ряд проблем через які стає більш схильна до атак.

Натомість перевагами правильно вибраної архітектури є те, що застосунок можна легко підтримувати, масштабувати та тестувати. Використання широко розповсюджених архітектур програмного забезпечення дає змогу легко розібратись у коді новому розробнику, через велику кількість документації про дану архітектуру.

Чиста архітектура – концепція проєктування програмного забезпечення, яка розділяє розв’язання проблем між різними рівнями програми [9]. Головна

ідея полягає у виокремленні бізнес-логіки від технологій, які використовуються для її реалізації. Це дозволяє розробникам працювати з одним рівнем програми не впливаючи на інші. Наприклад, якщо виникає потреба змінити базу даних, то під час внесення змін у шар, що відповідає за взаємодію з базою даних, бізнес-логіка не зміниться. Загалом, розглядаючи переваги даної архітектури, можемо зазначити, що вона забезпечує структурований підхід до проєктування застосунків, які легко тестувати модифікувати, а також масштабувати.

Згідно з концепцією, застосунок розбивається на 5 рівнів:

1. presentation layer: відповідає за обробку взаємодії з користувачем;
2. infrastructure layer: відповідає за взаємодію із зовнішніми службами;
3. persistence layer: відповідає за конфігурацію бази даних;
4. application layer: містить бізнес-логіку та випадки використання програми
5. domain layer: являється ядром програми, містить бізнес-правила, сутності та специфічну для домену логіку.

Головне правило, яке не можна порушувати: внутрішні рівні не можуть посилатись на зовнішні.

Окрім того, application layer створений з використанням шаблону CQRS, відповідно до якого операції читання та запису розділені на запити та команди. Команди працюють з доменними об’єктами – об’єктами, що представлені в базі даних, а запити повертають об’єкт Data Transfer Object скорочено DTO. DTO – це копія доменного об’єкта, але спрощена, без додаткової логіки, що потрібна у рамках роботи з базою даних. Таким чином DTO не відкриває користувачу схеми бази даних, що є більш безпечно. Основна мета цього шаблону – підвищити простоту та продуктивність системи. Використання різних моделей для операцій читання та запису дозволяє оптимізувати кожен тип операції окремо.

Також було використано бібліотеку FluentValidation для створення правил валідації для кожної команди та запиту в зрозумілому та зручному для читання форматі. Функціонал бібліотеки використовується для того, щоб

перевіряти чи всі поля класу відповідають заданим критеріям. Даний функціонал не містить в собі специфічної логіки перевірки, оскільки нічого не знає про об’єкти в базі даних натомість дає змогу перевіряти примітиви. Наприклад, містить перевірку чи певні поля містять достатню кількість символів, команди, що порушують дану вимогу не будуть виконані. У випадку, коли вебсервер отримує команду або запит, з даними, які не проходять валідацію, той зі свого боку видає користувачу помилку з описом які поля, які критерії порушують.

В застосунку передбачена обробка конфліктів паралелізму. В більшості сценаріїв декілька користувачів одночасно працюють з системою. Кожен користувач працює зі своєю копією даних і вони можуть змінювати одні й ті самі дані, наприклад, змінювати інформацію про університет, якщо користувачі застосують свої зміни одночасно або майже одночасно то виникне конфлікт, які дані вебсерверу записати у базу даних, а які ні. Існує не один спосіб вирішити конфлікт паралелізму, я припускаю, що дані ситуації є відносно рідкісними, і при появі конфліктної ситуації сервер видає помилку з повідомленням про те, що декілька користувачів пробують модифікувати одні й ті ж дані. Даний метод називається оптимістичним блокуванням або оптимістичне керування паралелізмом. У Entity Framework даний метод реалізується шляхом додавання до сутності маркера паралелізму. Сутності мають додаткове поле під назвою Version, яке в конфігурації відмічене як маркер, і дане поле автоматично керується базою даних. Таким чином, поле автоматично змінюється в базі даних, коли змінюється рядок. Під час команди збереження змін до бази даних Entity Framework генерує SQL код, у якому пошук відповідного рядка відбувається за Id та Version. При одночасному оновленні різними користувачами одних і тих самих даних Entity Framework створює виняток, який сервер обробляє і повертає користувачам код 403 – заборона виконати операцію.

Отож, застосунок поділений на рівні, кожен з яких має свою зону відповідальності. Завдяки цьому вдається писати чистий код з дотриманням принципів об’єктно орієнтованого програмування SOLID [10].

Принципи SOLID – це 5 принципів, дотримання яких дозволяє уникнути залежності між компонентами коду. Принцип єдиної відповідальності стверджує, що кожен клас повинен виконувати один обов’язок, тобто всі його методи повинні бути спрямовані на розв’язання однієї задачі. Якщо клас, містить методи, що відповідають за інше завдання, то дані методи потрібно винести в інший клас. За принципом відкритості/закритості клас має бути побудований таким чином, щоб нові зміни реалізовувались додаванням нового коду, а не зміною наявного. Принцип підставлення Лісков вимагає, щоб у випадку підставлення замість об’єкта базового класу об’єкт-нащадка, застосунок продовжував працювати й працювати коректно. Тобто, нащадки повинні розширювати поведінку базового класу, а не змінювати її. Принцип розділення інтерфейсу стверджує, що краще мати багато спеціалізованих інтерфейсів ніж один загальний. Останній принцип, принцип інверсії залежностей містить у собі два твердження:

- абстракції не повинні залежати від реалізацій;
- модулі низького рівня та високого рівня мають залежати від абстракцій, і модулі високого рівня не повинні залежати від модулів низького рівня.

Також під час створення застосунку, враховувались принципи YAGNI, основна ідея якого полягає у відмові від додавання зайвих функцій, та KISS, в основі якого лежить ідея про уникнення зайвої складності з допомогою використання тільки тих технік які дозволяють досягнути поставленої мети.

Дотримання даних принципів дає змогу писати чистий код, та не допускати лавиноподібного підвищення складності системи.

3.3 Створення Web API

Інтерфейсом прикладного рівня виступає шар presentation та являється мостом між клієнтом та серверною частиною. Шар містить контролери для сутностей. У кожному контролері є кінцеві точки, або ж методи що

обробляють запити на сервер. Кінцеві точки викликають методи бізнес-логіки, що знаходяться в шарі application, проте, щоб зменшити кількість прямих зв’язків між об’єктами використано бібліотеку Mediator яка реалізує шаблон проєктування “Посередник”. Завдяки шаблону, об’єкти спілкуються через окремий об’єкт-посередник, саме тому об’єкти даного шару залежать від посередника, а не від великої кількості інших компонентів, що представлені в інших шарах. Коли бізнес-логіка відпрацювала, кінцева точка повертає користувачу статусний код 200 – код успішного виконання запиту, і при потребі може повернути Json файл з потрібним вмістом.

REST – це підхід до архітектури мережевих протоколів, що використовує HTTP-запити для взаємодії з даними, такі як GET, POST, PUT, DELETE. Основні принципи REST:

1. робота без збереження стану: кожен запит клієнта має бути незалежним та має містити всю необхідну інформацію для обробки на сервері.
2. клієнт-сервер: клієнтська частина займається відображенням даних та приймає запити, а серверна займається зберіганням та оновленням даних.
3. кешування: дані, які передаються містять інформацію про те чи можна їх кешувати та на скільки часу.
4. однорідний інтерфейс: чотири основні інтерфейси: ресурси, HTTP-методи, представлення ресурсів, посилання між ресурсами.

Інтерфейс прикладного рівня, що використовує REST архітектуру називається RESTful API.

Swagger – набір інструментів для документування RESTful API, який був використаний при створенні проєкту. Інструмент надає інтерфейс користувача, у якому легко орієнтуватись. Це полегшує процес створення та тестування застосунку, оскільки при спробі зробити запит на модифікацію або створення сутності, користувач може проаналізувати яким має бути запит, задати необхідні дані та здійснити запит безпосередньо з інтерфейсу користувача. Крім того, Swagger полегшує управління різними версіями API та їх змінами.

3.4 Створення застосунку відображення розкладу

Односторінковий застосунок – це вебсайт який динамічно перебудовує поточну вебсторінку новими даними з вебсервера замість завантаження окремої цілої сторінки. Всі необхідні дані завантажуються під час початкового завантаження сторінки. Дані застосунки використовують асинхронний JavaScript та XML для обміну даними з сервером і оновлення тільки потрібної частини сторінки. Перевагами односторінкових застосунків є висока швидкість завантаження, багатоплатформність, простіша підтримка.

Застосунок відображення розкладу є односторінковим. Серед елементів користувацького інтерфейсу, які завжди присутні є шапка сторінки, на якій зображено логотип факультету, вибір сторінки для кого буде відображатись розклад та кнопка входу.

Застосунок розділений на дві частини, бекенд – частини застосунку, що не змінюються та фронтенд – таблиця розкладу, розкривні списки тощо. Під час запуску застосунку, на бекенд стороні виконується запит на прикладний програмний інтерфейс для отримання список усіх університетів. Ці дані відображається у першому розкривному списку. Під час вибору елемента з розкривного списку, застосунок робить наступний запит для отримання інших даних. Даний механізм зроблений з використанням хука ефекту. Хук дозволяє відстежувати стан компонента та реагувати на його зміни. Задля побудови розкладу було створено окремий клас, який містить усю необхідну логіку для заповнення таблиці відображення потрібними елементами.

Сторінка типово є бекенд компонентом, і вона точково заповнюється фронтенд компонентами.

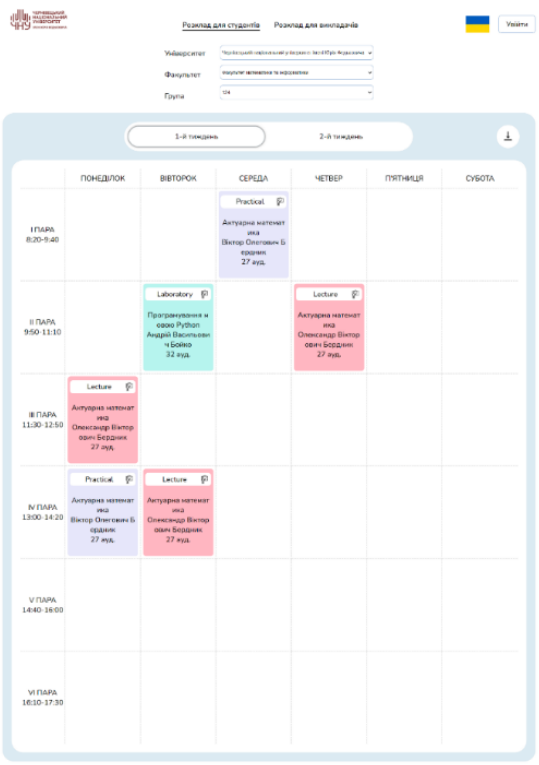

Рисунок 5. Вигляд застосунку на комп’ютерах

Дизайн інтерфейсу користувача, включаючи колірну гаму та шрифти, зроблений з урахуванням дизайну офіційного сайту ЧНУ.

Додатково, була реалізована авторизація в систему з допомогою протоколу OAuth 2.0. Протокол дає можливість одним вебзастосункам отримати доступ до ресурсів іншого. OAuth 2.0 дає змогу увійти на сайт через свій обліковий запис на іншому сайті, при цьому, не розкривати свої логін та пароль. Протокол працює на базі токенів, коли користувач погоджується надати доступ до його облікового запису, Google створює токен, який передається на вебзастосунок, в який ви хочете увійти. Використовуючи цей токен, вебзастосунок може отримати доступ до певних даних вашого облікового запису Google. Задля впровадження даного механізму авторизації в Google console cloud було створено відповідний проєкт, задано ім’я, які користувачі можуть використовувати авторизацію, та доступ до яких даних буде запитувати вебзастосунок. Використання даного протоколу дає змогу не зберігати у базі даних всіх користувачів, отже, уникнути реєстрації та проблем з шифруванням паролів. Всі студенти та викладачі університету мають

університетську пошту, яку можна використати для авторизації. Під час авторизації користувач надає доступ до таких даних: ім’я, електронна пошта та фото. Таким чином даний підхід авторизації є зручним та безпечним. Надалі, авторизація повинна бути на стороні вебсервера, проте реалізація авторизації саме на стороні застосунку відображення дало змогу краще зрозуміти нюанси роботи даного протоколу.

3.5 Використання інструментів для управління проєктом

Канбан являє собою метод управління розробкою програмного забезпечення, який базується на візуальному представленні завдань у вигляді карток на дошці. Цей метод дозволяє команді відстежувати прогрес роботи та керувати завданнями в режимі реального часу. Використання даного методу дозволяє досягти максимальної гнучкості та ефективності розробки.

Канбан являється представником популярної методології Agile-сімейства, разом з методологіями скрам, XP та іншими. Agile має такі основні принципи [11]:

1. взаємодія між людьми важливіша, ніж процеси та інструменти;
2. робочий продукт важливіший, ніж вичерпна документація;
3. співпраця з замовником важливіша, ніж контрактні зобов’язання;
4. реакція на зміни важливіша, ніж дотримання плану.

Класична канбан-дошка має 3 колонки: “Треба зробити”, “Робиться”, “Виконано”. Створені завдання поміщаються в першу колонку. Коли учасник команди береться за виконання завдання, він переміщує картку у другу колонку, коли завдання виконане то картка потрапляє у третю колонку.

Trello – це інструмент для управління проєктами, організований за принципом канбан. Так, інтерфейс Trello складається з дошок, кожна з яких відповідає за окремий проєкт. У дошці є можливість створювати довільну кількість колонок. У колонці можна створювати картки із завданнями. Кожна картка має назву, може мати опис, вкладення, дату до якої потрібно завдання виконати. Додатково, користувач може створювати власні мітки та застосовувати їх до карток.

Під час виконання дипломної роботи було визначено наступні колонки, для ефективного виконання завдань та більшого контролю:

1. беклог: список всіх завдань, які потрібно виконати у рамках розробки системи;
2. спринт: список завдань, які потрібно виконати в рамках проміжку часу у два тижні;
3. робиться: поточні завдання, які розробники виконують;
4. тестується: після виконання завдання, картку переміщують у цю колонку, на певний проміжок часу;
5. виконано: якщо виконані завдання протестовані й не виникає ніяких помилок то картка з завданням переміщається у дану колонку.

Даний розподіл колонок взятий з методу керування проєктами скрам. Методи скрам і канбан близькі, тому в рамках виконання дипломної роботи вони використовувались в комбінації.

У сучасному світі важко уявити розробку складного продукту без використання системи контролю версій. Основні принципи використання системи контролю версій включають роботу зі створеннями гілок, комітами та злиттям гілок. Задля усунення хаосу під час роботи з гілками використовувались наступні правила іменування гілок:

1. дотримання малого регістру;
2. використання тільки літер, цифр та дефісів;
3. назва має бути короткою та ідеально відображати завдання, яке виконується.

У назвах гілок використовувались префікси, які допомагають швидко визначити призначення гілок. В рамках виконання дипломної роботи було визначено 3 префікси: для додавання нових функцій - feature/, виправлення багів - bugfix/, реліз - release/. Використання Trello дало змогу автоматично генерувати назву вітки згідно з визначеними правилами. Кожна картка в Trello має свій унікальний ідентифікатор, який використовується до доступу до неї. Ідентифікатор складається з номера, що генерується автоматично, та назви

картки яку користувач вказав під час створення. Назва вітки виглядає наступним чином: feature/1-create-webapi-project.

Висновки

Під час виконання дипломної роботи було проведено аналіз сучасної цифровізації університету та країни загалом, аналіз та вибір технологій, шаблонів та архітектури, планування етапів розробки. Під час розробки застосунків використовувались розподілена система контролю версій Git та система управління проєктами Trello, для ефективного керування робочим процесом.

У результаті виконання даної курсової роботи було спроєктовано базу даних та розроблено два застосунки. Перший – відповідає за взаємодію з базою даних та надає прикладний програмний інтерфейс. Другий – відповідає за відображення розкладу.

Список використаних джерел

1. Міністерство освіти і науки України. Всеукраїнська школа онлайн [Електронний ресурс] / Міністерство освіти і науки України// mon gov.- Режим доступу: https://mon.gov.ua/ua/tag/vseukrayinska-shkola-onlajn
2. Всеукраїнська школа онлайн [Електронний ресурс] – Режим доступу: https://lms.e-school.net.ua/
3. Верховна рада України. Про затвердження Державної стратегії регіонального розвитку на 2021-2027 роки [Електронний ресурс] / Верховна рада України // zakom rada gov. – Режим доступу: https://zakon.rada.gov.ua/laws/show/695-2020-%D0%BF#Text
4. Дія освіта [Електронний ресурс] – Режим доступу: https://osvita.diia.gov.ua/
5. Міністерство цифрової трансформації України. Результати цифрової трансформації в регіонах України за 2023 рік [Електронний ресурс] / Міністерство цифрової трансформації України // Урядовий портал. – Режим доступу: https://www.kmu.gov.ua/news/rezultaty-tsyfrovoi-transformatsii-v-rehionakh-ukrainy-za-2023-rik
6. Львівський IT кластер. Динаміка IT-індустрії під час війни: Результати IT Research Ukraine 2023. [Електронний ресурс] / Львівський IT кластер// Lviv It cluster. – Режим доступу: https://itcluster.lviv.ua/dynamika-it-industriyi-pid-chas-vijny-rezultaty-it-research-ukraine-2023/
7. Wipo. Global Innovation Index. [Електронний ресурс] / Wipo // wipo int. – Режим доступу: https://www.wipo.int/global_innovation_index/en/
8. Simon Kemp. Digital 2024: Global Overview report [Електронний ресурс]/Simon Kemp // Datareportal. – Режим доступу: https://datareportal.com/reports/digital-2024-global-overview-report
9. Мартін Р. С. The clean architecture [Електронний ресурс] / Р. С. Мартін // The Clean Code Blog. – Режим доступу: https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html

10. Мартін Р. С. The principles of OOD [Електронний ресурс] / Р. С. Мартін // But UncleBob .com. – Режим доступу: http://butunclebob.com/ArticleS.UncleBob.PrinciplesOfOod
11. Welcome to Agile essentials[Електронний ресурс] // Agile Allliance. – Режим доступу: https://www.agilealliance.org/agile-essentials/