НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет
Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:
Завідувач кафедри
_____ Олександр Коваль
«___»_____2021 р.

Дипломна робота

на здобуття ступеня бакалавра

спеціальності 121 «Інженерія програмного забезпечення»

освітня програма «Інженерія програмного забезпечення розподілених систем»

на тему: «Web-портал для організації доступу до e-learning ресурсів (організація доступу для вчителя)»

Виконав:
студент IV курсу, групи ТІ-72
Бортнічук Нікіта Олександорвич _____

Керівник:
Доцент, кандидат технічних наук
Гагарін Олександр Олександрович _____

Рецензент:
_____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2021 року

Національний технічний університет України

“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

спеціальності 121 «Інженерія програмного забезпечення»

освітня програма «Інженерія програмного забезпечення розподілених систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Олександр Коваль (підпис)

”___”_____2021р.

ЗАВДАННЯ

на дипломну роботу студенту

_____Бортнічуку Нікіті Олександровичу_____

(прізвище, ім’я, по батькові)

1. Тема роботи _____

_____

_____

керівник роботи_____

(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”____” травня 2021р. № _____

2. Строк подання студентом роботи _____

3. Вихідні дані до роботи _____

_____

_____

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

_____

_____

_____

_____

_____

_____

_____

5. Перелік ілюстративного матеріалу

_____

_____

_____

_____

_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання ”___”_____202___ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи		
2.	Вивчення та аналіз задачі		
3.	Розробка архітектури та загальної структури системи		
4.	Розробка структур окремих підсистем		
5.	Програмна реалізація системи		
6.	Оформлення пояснювальної записки		
7.	Захист програмного продукту		
8.	Передзахист		
9.	Захист		

Студент _____ _____

(підпис) (прізвище та ініціали,)

Керівник роботи _____ _____

(підпис) (прізвище та ініціали,)

Анотація

Обсяг дипломної роботи складає 65 сторінок, 31 рисунок, 3 додатки та 9 посилань.

Метою дипломної роботи є створення систему онлайн курсів і організації доступу до e-learning ресурсів, яка надає можливість викладачам створювати курси, а студентам їх проходити.

У ході розробки було проаналізовано існуючі системи, що надають доступ до онлайн ресурсів, виділені основні переваги та недоліки. Сформовані завдання, що потрібно розв’язати для вирішення цих недоліків розробленій системі. Для реалізації програмного продукту розроблено серверну частину за допомогою мови програмування Java та фреймворку Spring, та дві реалізації клієнтської частини: веб та мобільний застосунки.

Результатом роботи стало створення програмного засобу ведення навчального процесу за допомогою онлайн курсів.

Ключові слова: навчальний процес, курси, веб застосунок, Android, викладач, ресурси, доступ.

Abstract

Thesis consists of 65 pages, 31 figures, 3 annexes and 9 references.

The purpose of the thesis is to create system of online courses and access to e-learning resources, which allows teachers to create courses and students to pass them.

During the development, the existing systems that provide access to Internet resources, highlighted by the main advantages and disadvantages, were analyzed. Problems that require solving these shortcomings of the system, were formed. To implement the cross platform software of the product, a server part was developed using the Java programming language and the Spring framework, and two implementations of the client part: web and mobile applications.

The result was the creation of a software tool for conducting the educational process with the help of online courses.

Keywords: educational process, courses, web application, Android, teacher, resources, access.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

e-learning – онлайн навчання.

Фреймворк — зовнішній програмний продукт.

FDD – метод розробки програмного забезпечення.

UML – уніфікована мова моделювання.

MVC – шаблон проектування.

СКБД – система керування базами даних.

ПЗ – програмне забезпечення.

SQL – структурована мова запитів

Front-end – веб інтерфейс

HTML – мова розмітки гіпертексту

CSS – каскадні таблиці стилів

ВСТУП

Ефективність навчального процесу в університеті багато в чому залежить від правильної організації академічних матеріалів та надання належного доступу до сховищ, де такі матеріали знаходяться. На даний момент у багатьох університетах є власні портали для збереження та отримання інформації, але часто вони є не дуже зручними для ефективного вивчення матеріалів.

Аби зберегти зацікавленість студента у здобутті необхідних знань, університету варто слідувати сучасним та актуальним методам ведення навчального процесу. Оскільки сучасне навчання в університеті передбачає відведення більшості часу студента для самостійного вивчення тих чи інших дисциплін, правильна організація поза аудиторного навчання є значущим фактором ефективності організації всього процесу здобуття вищої освіти. Останнім часом ця проблема стала особливо гострою, адже через пандемію більшість університетів змушені були перейти на дистанційне навчання і частка самостійного опрацювання інформації значно зросла.

Через це викладачам варто було швидко пристосовуватись до нових обмежень. Часто їм доводилось надавати посилання на зовнішні ресурси або надсилати окремі файли з необхідною інформацією. Такий підхід не є зручним, адже матеріали не були добре організовані, деякі з них могли загубитись, і при цьому не було єдиного засобу обміну цими даними. Також значна частка часу витрачалась на перевірку домашнього завдання.

Одним з актуальних методів ведення навчального процесу є організація курсів для отримання нових знань та навичків у відповідній предметній області. Дійсно, проходження курсів є досить ефективним способом засвоєння вивченого матеріалу та збереження зацікавленості студента у самостійному виконанні завдань. Тому викладачі пропонували зовнішні ресурси, де студентам надавалась можливість проходити курси онлайн. Головним недоліком таких ресурсів є недостатня функціональність і неможливість реалізувати нові особливості для системи, що використовується. Також

часто вони є платними, а отже університет має витрачати додаткові кошти на забезпечення достатнім функціоналом для викладачів та студентів. При цьому викладачі не могли бути впевнені у якості наданих навчальних матеріалів, адже нерідко не було можливості створити новий курс або додати власні матеріали до тих, що вже існують.

Отже, беручи до уваги вищенаведені проблеми, було вирішено створити веб-портал, де кожен викладач зможе додати власний навчальний курс з відповідної дисципліни. При цьому надається можливість завантажувати різні матеріали, в тому числі текст, зображення, відео тощо. Таким чином, викладач може сам нести відповідальність за надану інформацію, а не покладатися на зовнішні ресурси. Також усі файли структуровані за уроками, до яких студенти мають зручний доступ. Перевірка домашнього завдання може займати значно менше часу, адже в даній системі побудований функціонал, за яким для курсів з програмування викладач може написати модульні тести до класів, що має реалізувати студент, які згодом будуть автоматично запускатись і показувати результат правильності виконання завдання. Отже, відпадає потреба у перевірці домашнього завдання для таких курсів. Студент, в свою чергу, має змогу пройти доступні курси і отримати необхідні знання з відповідних дисциплін.

Дана система може бути використана в навчальному процесі усіх ступенів здобуття освіти, в тому числі в середньому, вищому та професійному, адже веб-портал має зручний та зрозумілий користувацький інтерфейс та є універсальним на рівні галузь та дисциплін, до яких створюються курси.

1 ПОСТАНОВКА ЗАДАЧІ

Ідея створення веб-порталу для організації доступу до e-learning ресурсів виникла з початку карантину, коли майже усі навчальні заклади перейшли на дистанційне навчання, і усім учасникам навчального процесу довелося підлаштовуватися під нові умови його ведення. Саме тоді вже існуючі системи онлайн курсів почали пропонувати свої послуги, а університети ними користуватись. Але часто не можна було бути впевненим в якості інформації з наданих ресурсів. Тим паче, іноді бракувало функціоналу для забезпечення змістовного ведення процесу отримання освіти.

Тому метою даної роботи стало створення сервісу, який надавав би можливість викладачам та студентам завантажувати та отримувати доступ до e-learning ресурсів за допомогою організації системи онлайн курсів. Це допоможе зробити навчальний процес зручним та цікавим, а ще це частково вирішить проблему дистанційного навчання. Також однією з головних цілей є створення системи, що може бути легко розширеною шляхом додавання нового функціоналу та її подальшого вдосконалення

1.1 Основні компоненти

Аби реалізувати даний сервіс необхідно побудувати архітектуру системи що складається з двох головних частин: клієнтської та серверної.

1.1.1 Клієнтська частина

Клієнтська частина – це програма, яка включає в себе всі елементи, з якими користувач взаємодіє. При цьому таких програм може бути декілька. Наприклад, веб сайт у браузері або мобільний додаток можуть одночасно являти собою фронтенд частину застосунку. Одна з основних цілей розробки клієнтського інтерфейсу - створити плавний інтерфейс, що призначений для користувача. Іншими словами, інтерфейс програми або веб сайту повинен бути інтуїтивно зрозумілим і простим у використанні. Хоча це звучить, як проста ціль, вона може бути напрочуд складною, оскільки не всі користувачі або пристрої однакові. Наприклад, застосунок, розроблений для мобільного пристрою, вимагає істотно іншого зовнішнього інтерфейсу, ніж настільний застосунок.

Веб сайти повинні добре працювати на різних пристроях і екранах різних розмірів, тому сучасні вимоги до програмних сервісів зазвичай передбачають адаптивний дизайн.

В результаті в процесі розробки клієнтської частини має бути побудований веб застосунок, який користувач може запустити у браузері, а також мобільний застосунок, яким можна користуватися з власного смартфону.

1.1.2 Серверна частина

Серверна частина – це частина комп'ютерної програми або програмного коду, що дозволяє програмі працювати і до якої користувач не може отримати доступ. Більшість даних та операційний синтаксис зберігаються та доступні в серверній частині комп’ютерної системи. Зазвичай код складається з однієї або декількох мов програмування. Серверна частина також називається рівнем доступу до програмних чи апаратних даних і включає будь-які функції, до яких потрібно отримати доступ та отримати доступ цифровими засобами. В системі, що розроблялася, серверна частина також містила базу даних, та підпрограму, яка обробляла основні операції над даними.

База даних – це організований набір даних, які зазвичай зберігаються та мають електронний доступ з комп’ютерної системи. Там, де бази даних є більш складними, вони часто розробляються з використанням формальних методів проектування та моделювання[1]. В контексті баз даних варто розглянути поняття СКБД. Система керування базами даних (СКБД) - це комплекс програмних засобів, необхідних для створення структури нової бази, її наповнення, редагування вмісту і відображення інформації.

1.2 Архітектура

Важливим завданням є організація спілкування між головними частинами системи, адже правильно побудована взаємодія є запорукою швидкого отримання коректних даних та їх відображення користувачам. При цьому спілкування між ними має відбуватися за допомогою зручних та сучасних методів реалізації таких зв’язків (Рисунок 1.1).

З боку користувача правильно побудована взаємодія між клієнтською та серверною частинами є запорукою плавної та коректної роботи інтерфейсу застосунку. Для самої системи – це можливість розділити роботу між розробниками так, щоб кожний відповідав за побудову свого компоненту.

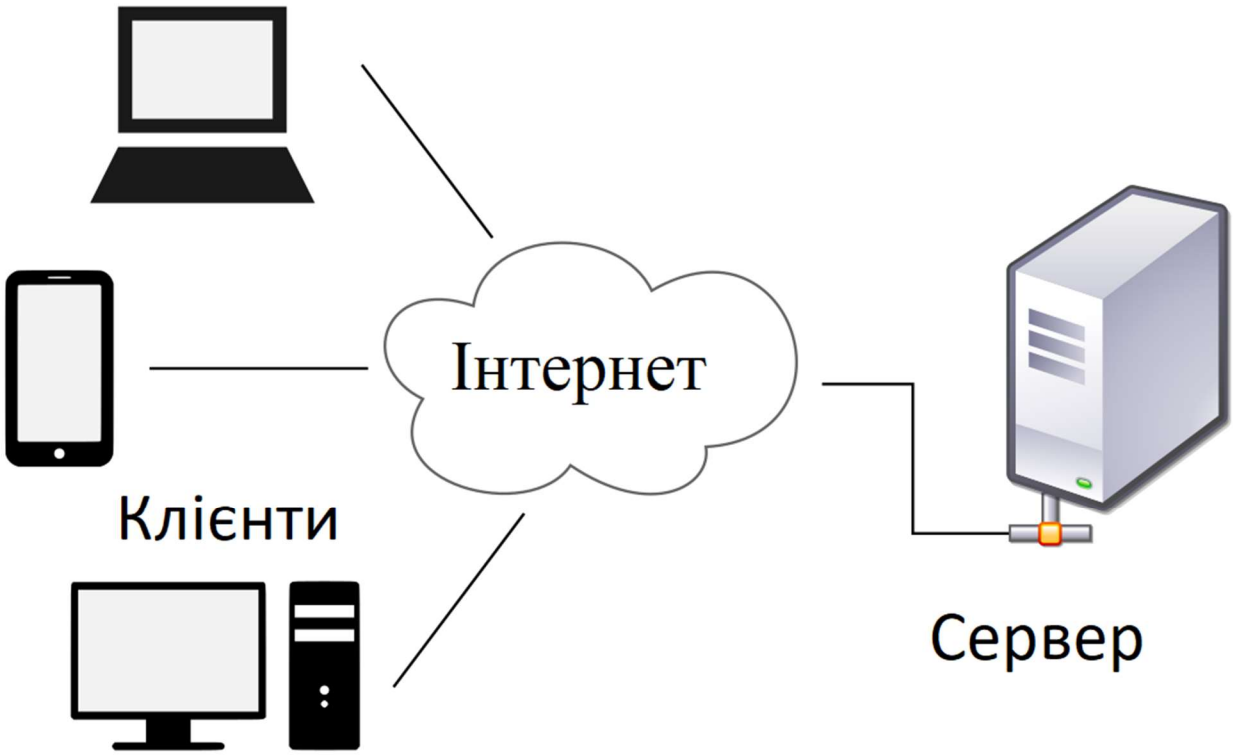


Рисунок 1.1 – Клієнт-серверна взаємодія

З іншого боку організація архітектури бази даних системи також є важливою частиною побудови застосунку загалом. Отже, задля її реалізації потрібно:

1. Обрати вид бази даних за моделлю організації, що буде відповідати вимогам системи. Таких видів є декілька:
    - Ієрархічна. Ієрархічна база даних може бути представлена як дерево, що складається з об'єктів різних рівнів. Між об'єктами існують зв'язки типу «предок-нащадок». При цьому можлива ситуація, коли об'єкт не має

нащадків або має їх кілька, тоді як у об'єкта-нащадка обов'язково тільки один предок.

- Мережева. Така база даних подібна до ієрархічної, за винятком того, що кожен об'єкт може мати більше одного предка.
- Реляційна. Реляційна база даних зберігає дані у вигляді таблиць. Найбільш уживані СУБД використовують реляційну модель даних.
- Об'єктно-орієнтована. У базі даних цього виду дані оформляють у вигляді моделей об'єктів.

2. Побудувати архітектуру взаємодії між сервером та базою даних(Рисунок 1.2)

Рисунок 1.2 – Взаємодія між сервером та базою даних

1.3 Розширюваність системи

Аби забезпечити можливість розширення для системи, необхідно було сформувати суворі правила при розробці програмного забезпечення.

По-перше, для того, щоб розробники, які в майбутньому будуть приймати участь у подальшій реалізації нових функцій системи, могли розуміти, як працює програма, а

також за що відповідає кожен підкомпонент, програмний код має бути написаний зрозуміло та з дотриманням правил «чистого» коду.

По-друге, мають бути використані шаблони побудови програмного забезпечення, що дасть можливість усунути заплутаність роботи системи, та зробить процес додавання нових модулей в системі легким, і при цьому не буде необхідності в зміні вже працюючого коду.

По-третє, кожен модуль системи слід забезпечити повним покриттям юніт та інтеграційними тестами, адже це є запорукою того, що при додаванні нового функціоналу, всі помилки будуть виявлені автоматично при запуску тестів(Рисунок 1.3). Тобто, система продовжить працювати стабільною. При цьому тести користувацького інтерфейсу є опціональним, і вони можуть бути заміненими мануальними тестами.

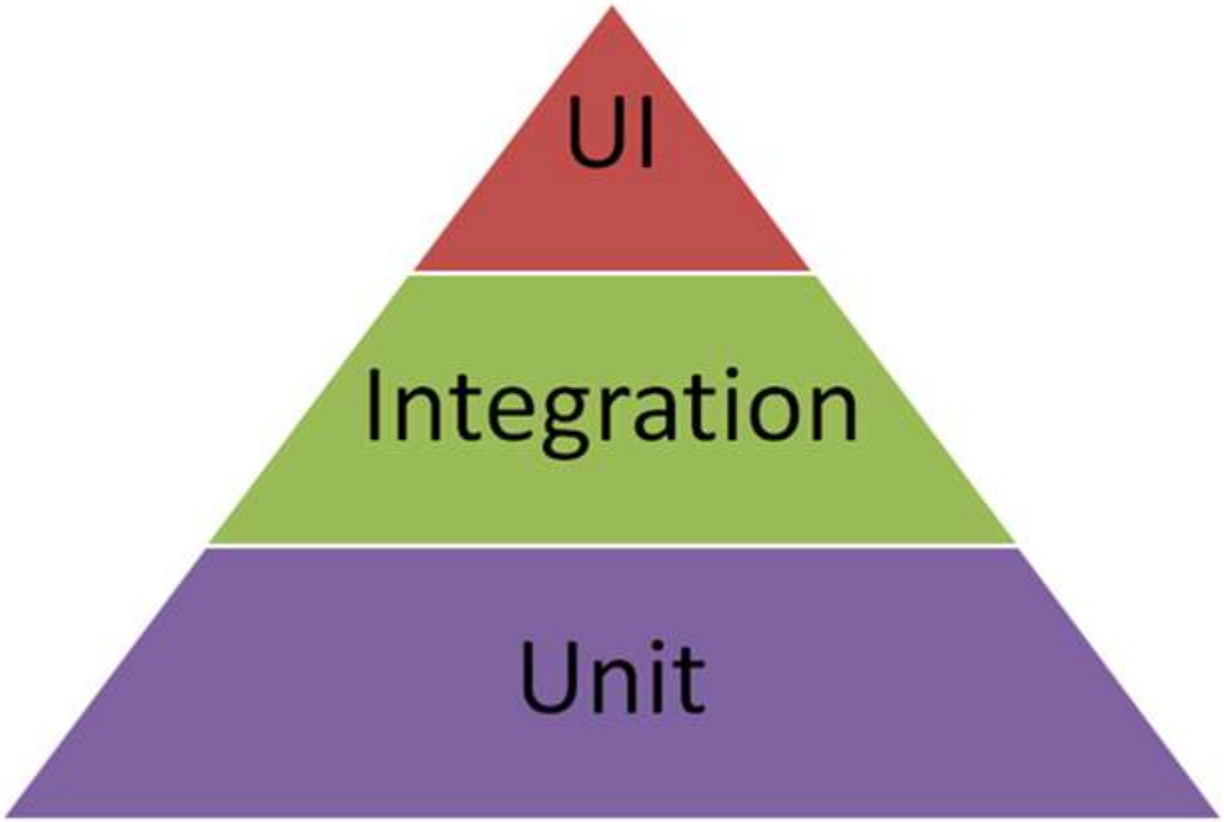


Рисунок 1.3 – Піраміда тестування Майка Кона[2]

2 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

2.1 Загальний опис

Проблема організації навчального процесу за допомогою онлайн курсів є надзвичайно актуальною в сучасному світі. Все більша кількість людей віддає перевагу вивченню нових матеріалів саме таким способом. Власне, онлайн-курс – це спосіб засвоїти нові навичку або отримати нові знання, не виходячи з власного будинку. Вони можуть бути платними, але також часто їх пропонують безкоштовно. Деякі пропонуються навчальними закладами, а інші - фахівцями у своїй галузі. Найголовніше для онлайн-курсу – це зацікавленість та чіткий результат.

Онлайн-курси можуть мати багато різних форм. Але на базовому рівні спільним для всіх курсів є те, що вони навчають знань чи навичок того, хто їх приймає. Онлайн-курси можна проходити через веб-сайт, а також їх можна переглядати на мобільному пристрої, або планшеті. Це дозволяє студентам зручно отримувати доступ до них де завгодно та в будь-який зручний час.

Вони можуть мати різні матеріали, включаючи навчальні відео, аудіо файли, зображення, тексти чи інші документи. Більшість онлайн-курсів також мають дискусійні форуми, групи громад або варіанти обміну повідомленнями, що дозволяє певним чином спілкуватися студентам між собою та / або викладачем.

Кожен курс складається з уроків та, як правило, уроки попередньо розробляються та записуються перед тим, як завантажити та розмістити їх на веб-сайті до курсу. Вони, як правило, викладаються в послідовному порядку, за яким слід дотримуватися студент. Також може існувати певна форма оцінювання для вимірювання успішності та досягнень студента, або курс може бути повністю самокерованим.

Перш за все, онлайн-курс повинен бути цікавим. Це дозволяє зробити так, щоб той, хто навчається, насолоджувався уроками та міг зберігати інформацію та застосовувати її у своєму власному житті.

Чудовий курс - це той, коли студент відчуває вкладеність у навчальний процес і має почуття спільності з іншими студентами та викладачами. Щоб курс був успішним, він повинен мати інтерактивну складову, яка допомагає оживити навчання. Тому виділяють основні правила створення курсів:

- **Якісний вміст.**

Перш за все, курс повинен мати високоякісний вміст і забезпечувати те, що він обіцяє. Без цього після завершення онлайн-курсу можна навіть не відчути, що студент отримав знання та навички, які він очікував. Або такий курс буде просто нудним без будь-яких реальних та корисних знань.

- **Використання мультимедії.**

Ми більше не обмежуємось лише підручниками та дошкою. Завдяки технологіям існує велика кількість ефективних та цікавих способів представити інформацію студентам. Онлайн-курс повинен використовувати такі речі, як відео, подкасти, інтерактивні веб-сторінки та навіть мобільні програми, щоб реально заохотити студентів. Це робить навчання набагато приємнішим, ніж просто читання довгого документу.

- **Правильна стимуляція.**

Онлайн-курс потрібно чітко налаштувати так, щоб студенти не були перевантаженими та не відчували нудьгу. Інформацію слід розбивати на певну кількість уроків, які мають сенс. Якщо на курсі є якісь великі проекти, потрібно виділити достатньо часу, щоб їх виконати, щоб уникнути напруги та стресу. Менші завдання повинні мати обґрунтування , а не просто призначатись, щоб якось зайняти студентів.

Отже, впровадження системи онлайн курсів в навчальний процес університету дозволить заохотити студентів на виконання освітнього плану та засвоєння нових матеріалів з цікавістю та заохоченням.

2.2 Огляд існуючих рішень

тому існує багато ресурсів, які надають таку можливість. Кожна з них має свої переваги та недоліки, і задачею є побудувати таку систему, яка б зберегла всі переваги і при цьому усунула і нівелювала недоліки вже існуючих систем.

2.1.1 Coursera

Однією з таких програм є веб сайт coursera.org, який надає змогу користувачам проходити курси, передивляючись навчальні відео та текст до них, при цьому виконуючи домашні завдання(рисунок 2.1). Coursera пропонує своїм користувачам сотні безкоштовних онлайн-курсів з різних дисциплін, у разі успішного закінчення яких користувач отримує сертифікат про проходження курсу.

Coursera співпрацює з університетами з різних країн світу для викладання курсів цих навчальних закладів онлайн. Наразі пропонуються курси в наступних галузях: інженерія, гуманітарні науки, медицина, біологія, суспільні науки, математика, бізнес, інформатика та інші.

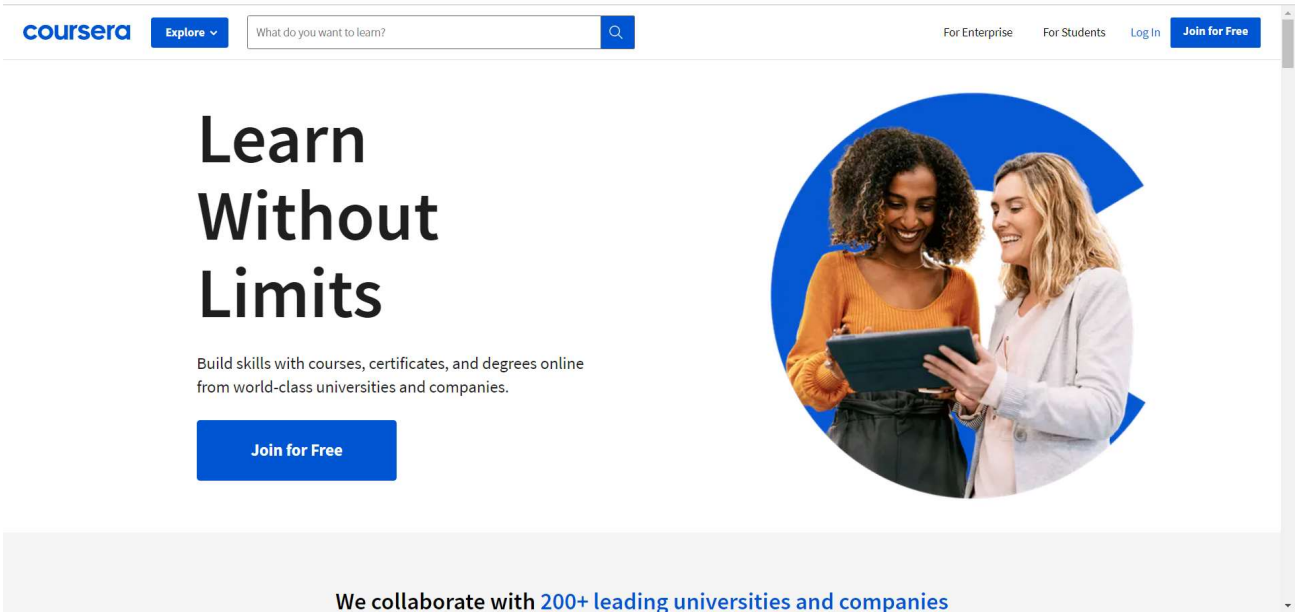


Рисунок 2.1. Головна сторінка веб сайту www.coursera.org

Серед основних функції даного сервісу можна виділити:

- проходження онлайн курсів через всесвітню мережу інтернет;
- завантаження відео або аудіо матеріалів на пристрій;
- можливість переглядати завантажені матеріали без доступу до інтернету;
- участь в обговоренні завдань між студентами та можливість спілкуватися на форумі;
- після успішного проходження курсів студент отримує сертифікат, який він може надалі використовувати у резюме при прийомі на роботу;
- доступ до програм найбільших та найпрестижніших університетів світу(Рисунок 2.2);
- велика варіативність в плані вибору предметної області дисципліни, від курсів

Отже, портал є доволі зручним засобом самостійного навчання, і користується досить великим попитом. Але головною проблемою є те, що більшість курсів є платними, що не дає змогу використовувати таку систему в університеті. Також, створити новий курс є досить нетривіальною задачею, адже для цього потрібно мати акаунт вчителя, який також є платним.

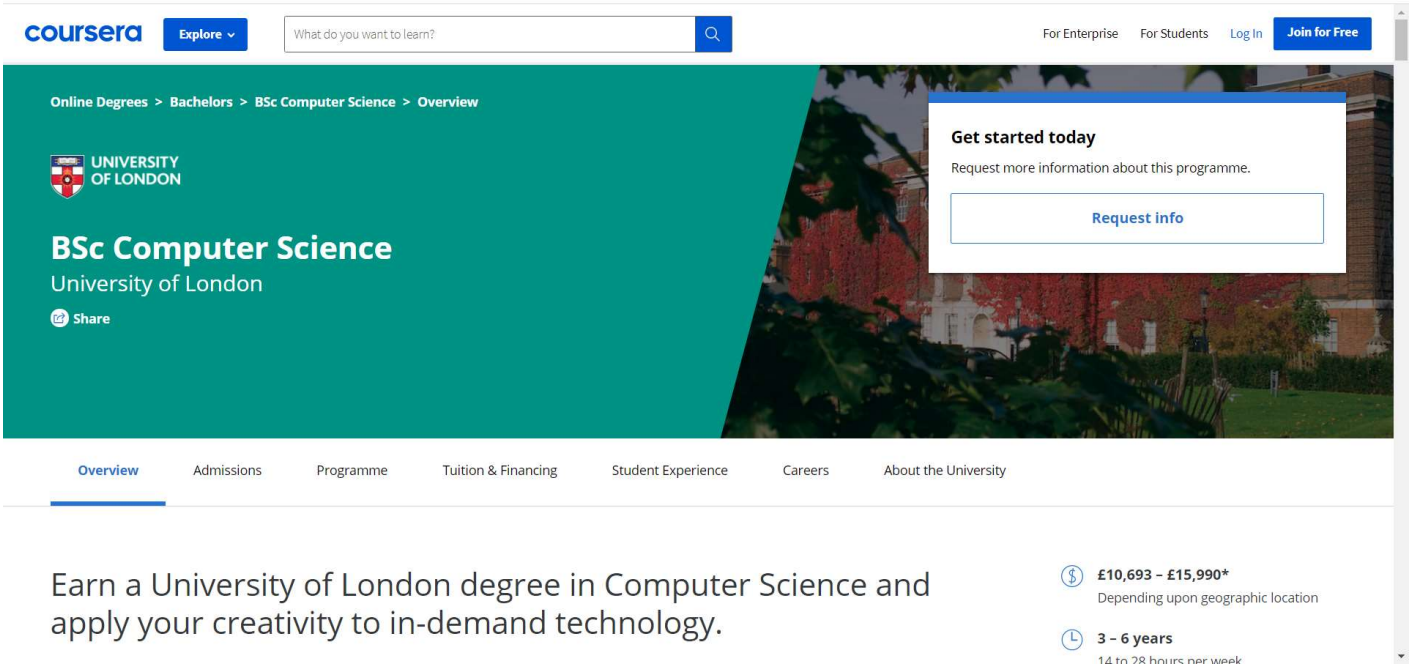


Рисунок 2.3 Навчальна програма для бакалаврів Лондонського університету

2.1.2 KPI Campus

Наступним прикладом з даної предметної області є система, що була розроблена студентами Національного Технічного Університету України "Київський політехнічний інститут імені Ігоря Сікорського". KPI Campus – це система підтримки навчального процесу(рисунок 2.4). Campus надає можливість користуватися багатьма корисними функціями, такими як:

- Переглядати бали за поточні дисципліни, атестації, сесії тощо.
- Викладачі можуть завантажувати методичні матеріали, а студенти їх переглядати.
- Адміністрація університету має можливість проводити опитування якості викладання та робити різні важливі оголошення.

Campus дійсно є досить зручним сервісом, але при цьому має декілька проблем. По-перше, система була написана досить давно, і тому для її побудови використовувалися вже досить застарілі методи та засоби розробки програмного забезпечення. Через це, досить часто робота даного сервісу не є стабільною. По-друге,

веб портал не є досить зручним для вивчення методичних матеріалів, тому студенти доволі рідко користуються саме навчальною інформацією.

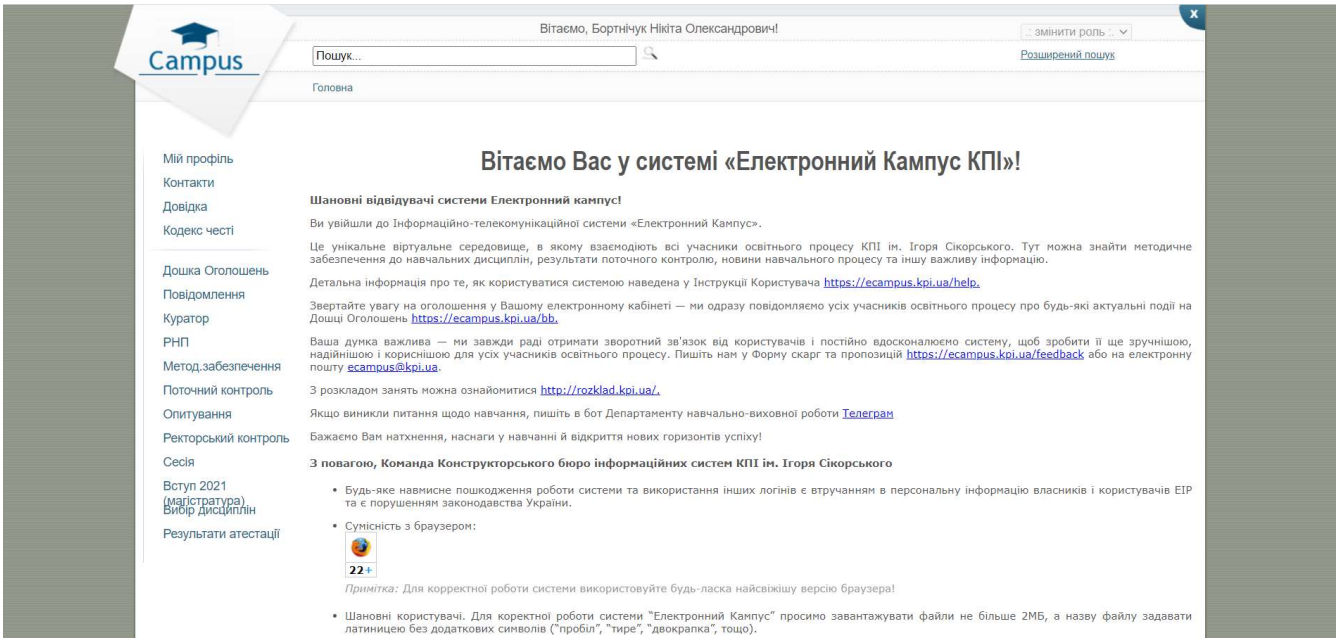


Рисунок 2.4 Головна сторінка платформи KPI Campus

2.1.3 Google Classroom

Ще одним з існуючих рішень є сервіс, розроблений компанією Google, що має назву Google Classroom(Рисунок 2.5). Цей сервіс можна використовувати у своїй школі або в університеті для впорядкування завдань, активізації співпраці та сприяння спілкуванню між викладачем та студентом. До Classroom можна доступитись через браузер або за допомогою мобільного застосунку. Ви можете використовувати Classroom із багатьма інструментами, представленими компанією Google, такими як Gmail, Docs та Calendar. Даний сервіс є досить зручним інструментом ведення навчального процесу, і вже багато хто з викладачів користуються даним застосунком.

Розробники даної системи розробляли застосунок так, щоб і вчитель, і студент мали в першу чергу зручний користувацький інтерфейс. Це забезпечило швидкість у засвоєнні основних можливостей системи.

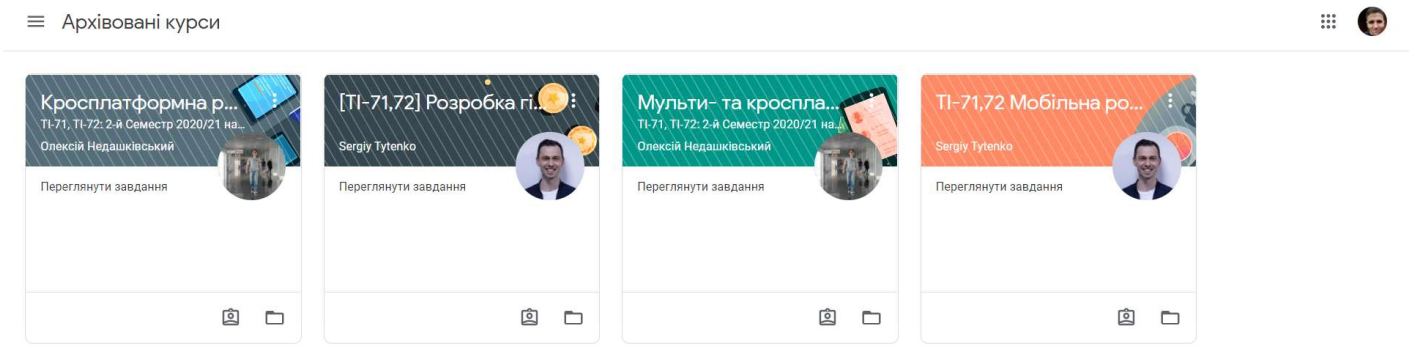


Рисунок 2.5 – Архівовані курси з веб порталу Google Classroom

Основними ролями в даній системі є викладач і студент, а тому слід розділяти їх головні функції.

Для викладача:

- можливість створити курс;
- додавання уроків до курсу;
- завантаження матеріалів до уроків;
- додавання домашнього завдання да строків здачі;
- можливість спілкуватись із студентом за допомогою вбудованого онлайн чату.

Для студента:

- можливість приєднатись до курсу;
- переглядання уроків та наданих матеріалів до них;
- додавання виконаного домашнього завдання у вигляді файлу або посилання
- якщо у студента виникають питання щодо завдань або матеріалів, він може поставити запитання викладачу.

Також для даної системи розроблений мобільний застосунок, який, по суті, надає можливість отримати доступ до тих самих функції, що реалізовані для комп’ютерної версії(Рисунок 2.6)

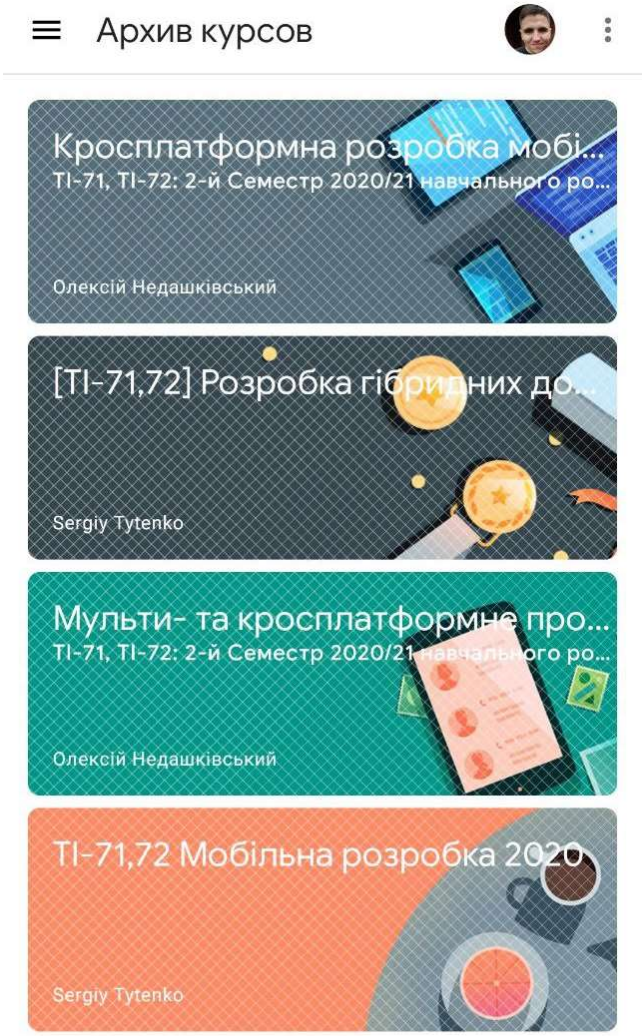


Рисунок 2.6 – мобільний застосунок Google Classroom

Головним недоліком даної системи є те, що вона є зовнішньою, і тому можна стикнутися з проблемою неможливості розширення функціоналу даної системи.

3 ОПИС МОДЕЛІ ТА МЕТОДІВ РЕАЛІЗАЦІЇ СИСТЕМИ

3.1 Аналіз моделі розробки ПЗ

У якості моделі розробки програмного забезпечення був обраний Agile принцип розробки.

3.1.1 Agile методологія

Методологія розробки програмного забезпечення Agile - це один із найпростіших та найефективніших процесів, що дозволяє перетворити поставлені задачі до системи на програмні рішення. Agile - це термін, що використовується для опису підходів до розробки програмного забезпечення, які використовують постійне планування, навчання, вдосконалення, співпрацю в команді, еволюційний розвиток та раннє впровадження. Такий підхід не тільки дозволяє на початкових етапах проектування не робити чітких планів, а навпаки заохочує гнучкі реакції на зміни в майбутньому.

Agile розробка програмного забезпечення наголошує на чотирьох основних цінностях:

- Індивідуальна та командна взаємодія понад процесами та інструментами;
- Працююче програмне забезпечення понад всебічною документацією;
- Співпраця клієнтів понад переговорами про контракт;
- Відповідь на зміни понад дотримання початкового плану[3].

Один з принципів - взаємодія - має на увазі, що замовник взаємодіє з командою, команда з замовником - усі між собою. Це дозволяє обмінюватися досвідом між учасниками команди і клієнтом і кожному з них впливати на прийняття рішень. За рахунок такого підходу знижуються ризики втрати часу і грошей і підвищується здатність команди вирішувати складні нестандартні завдання з високим ступенем невизначеності.

Однак взаємодії всіх і з усіма можуть вилитися у хаос, що впливає на всі сфери розробки. Тому використовуючи Agile потрібно розуміти обмеження: команди повинні бути невеликі, учасники повинні бути компетентні та мотивовані, ітерації короткі з

максимально зрозумілими цілями, встановлені чіткі обмеження за часом і кінцевий результат повинен бути очевидним.

Agile чудово справляється з невизначеністю, зумовлюючи майбутнє на більш короткий період. Правило таке: чим вища невизначеність - тим коротша ітерація, причому її тривалість може бути навіть у рамках 24 годин, як і відбувається на хакатоні. На початку кожної ітерації неминуче виконується контроль, ретроспектива, оцінка та аналіз результатів, планування наступної ітерації.

При всій повноті поняття Agile підходу до розробки ПЗ, такий підхід не може бути застосований до конкретної програмної реалізації, адже в ньому описані тільки загальні принципи даного підходу. Але існує багато конкретних імплементацій цього підходу(Рисунок 3.1).

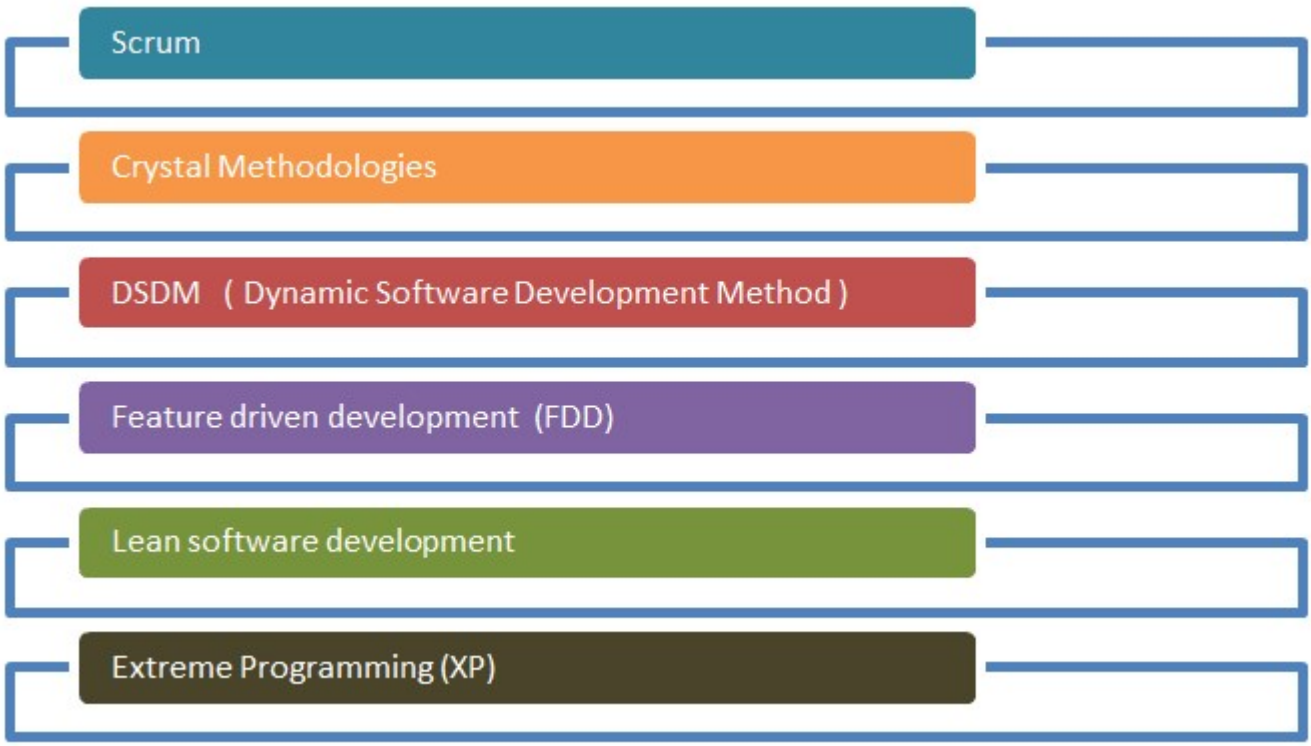


Рисунок 3.1 – Agile методи розробки ПЗ

При розробці програмного продукту був використаний Feature Driven Development(FDD) підхід.

3.1.2 Feature Driven Development

Цей метод зосереджений на "проектуванні та побудові" особливостей. На відміну від інших agile методів, FDD описує дуже конкретні та короткі етапи роботи, які потрібно виконувати окремо для кожної частини функціоналу. Він включає покрокове керівництво предметної області, тестування дизайну, просування збірки, написання та перевірку коду[4]. FDD розробляє продукт, дотримуючись наступних порад:

- Моделювання об'єкта предметної області;
- Розробка окремої частини функціоналу;
- Клас відповідає компоненту
- Тестування
- Управління конфігурацією
- Регулярні збірки
- Видимість прогресу та результатів(Рисунок 3.2)

Рисунок 3.2 – Feature Driven Development

3.2 Проектування системи

Етап проектування є важливою частиною процесу розробки програмного продукту. Якщо до розробки специфікацій, які включають і алгоритми, поставитися недостатньо уважно, то в подальшому, на етапі реалізації, можуть виникнути проблеми. Специфікації і алгоритми зажадають додаткового доопрацювання, витрат тощо. На етапі ж налагодження програми може з'ясуватися, що алгоритм недосконалий, помилковий або взагалі нездійсненний.

Проектування системи – це процес визначення елементів системи, таких як модулі, архітектура, компоненти та їх інтерфейси та дані для системи на основі визначених вимог. Результат проектування повинен задовольняти конкретні потреби та вимоги бізнесу чи організації.

Однією з основних частин проектування системи є процес побудови UML-діаграм. Уніфікована мова моделювання (UML) - це загальноприйнята, розвиваюча мова моделювання в галузі програмної інженерії, яка викликана забезпечити стандартний спосіб візуалізації дизайну системи та дозволяє описувати програмне забезпечення як структурно, так і поведінково, з графічними позначеннями[5].

3.2.1 Функціональні вимоги до системи. Діаграма прецедентів

В Уніфікованій мові моделювання (UML) діаграма прецедентів може узагальнювати функціональні можливості користувачів вашої системи (також відомих як актори) та їх взаємодії з системою. Для її створення потрібно використовувати набір спеціалізованих символів та сполучників. Дієва схема ефективної побудови діаграми прецедентів дозволить проставити чіткі межі у функціональних можливостях кожного користувача.

Метою діаграми прецедентів є відображення динамічного аспекту системи. Дані діаграми використовуються для збору вимог системи, включаючи внутрішні та зовнішні впливи. Ці вимоги в основному є вимогами до проектування. Отже, коли система

аналізується для збору її функціональних можливостей, готуються прецеденти та визначаються дійові особи. Коли початкове завдання завершено, діаграми використання моделюються для подання зовнішнього вигляду. Підсумовуючи цілі діаграм прецедентів можуть бути такими:

- Збір вимог системи;
- Отримання зовнішнього вигляду системи;
- Визначення зовнішніх та внутрішніх факторів, що впливають на систему;
- Організація взаємодії між вимогами акторів.

В процесі проектування системи було побудовано діаграму прецедентів(Рисунок 3.3)

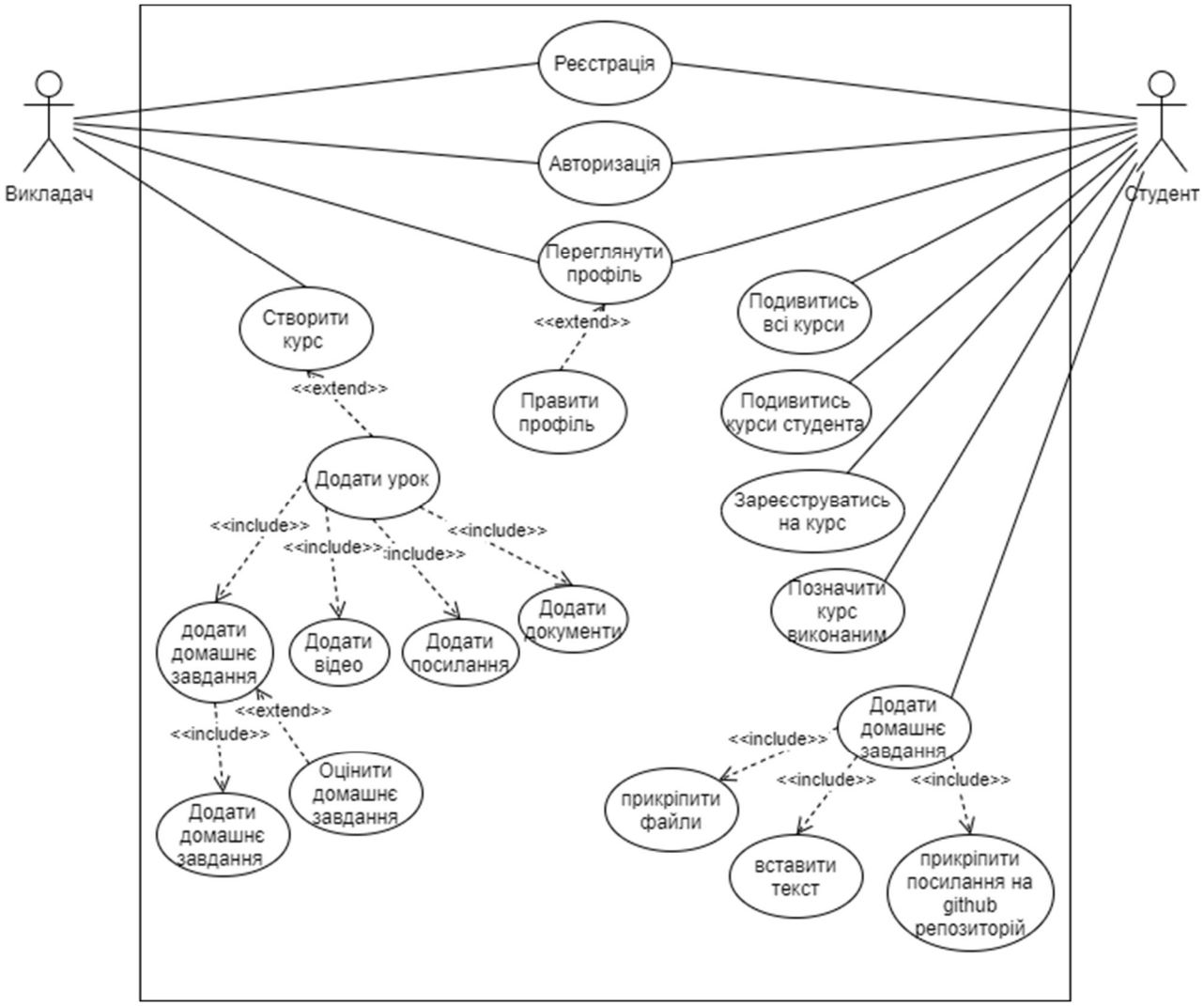


Рисунок 3.3 – Діаграма прецедентів системи

Таким чином в системі можна виділити два основних актори(користувачі системи):

1. Викладач
2. Студент

Як видно на Рисунку 3.3 обоє акторів системи мають декілька однакових функцій, а саме:

- Реєстрація користувача;
- Авторизація;
- Можливість переглянути профіль та правити його

Оскільки задачею є організація доступу тільки для викладача, можна окремо виділити його основні можливості в системі. Тобто у кожен викладач може:

- Створити курс;
- Додати урок;
- Додати матеріали до урок, такі як текст, відео, аудіо тощо;
- Надати домашнє завдання до уроку
- Перевірити домашнє завдання після його здачі студентом

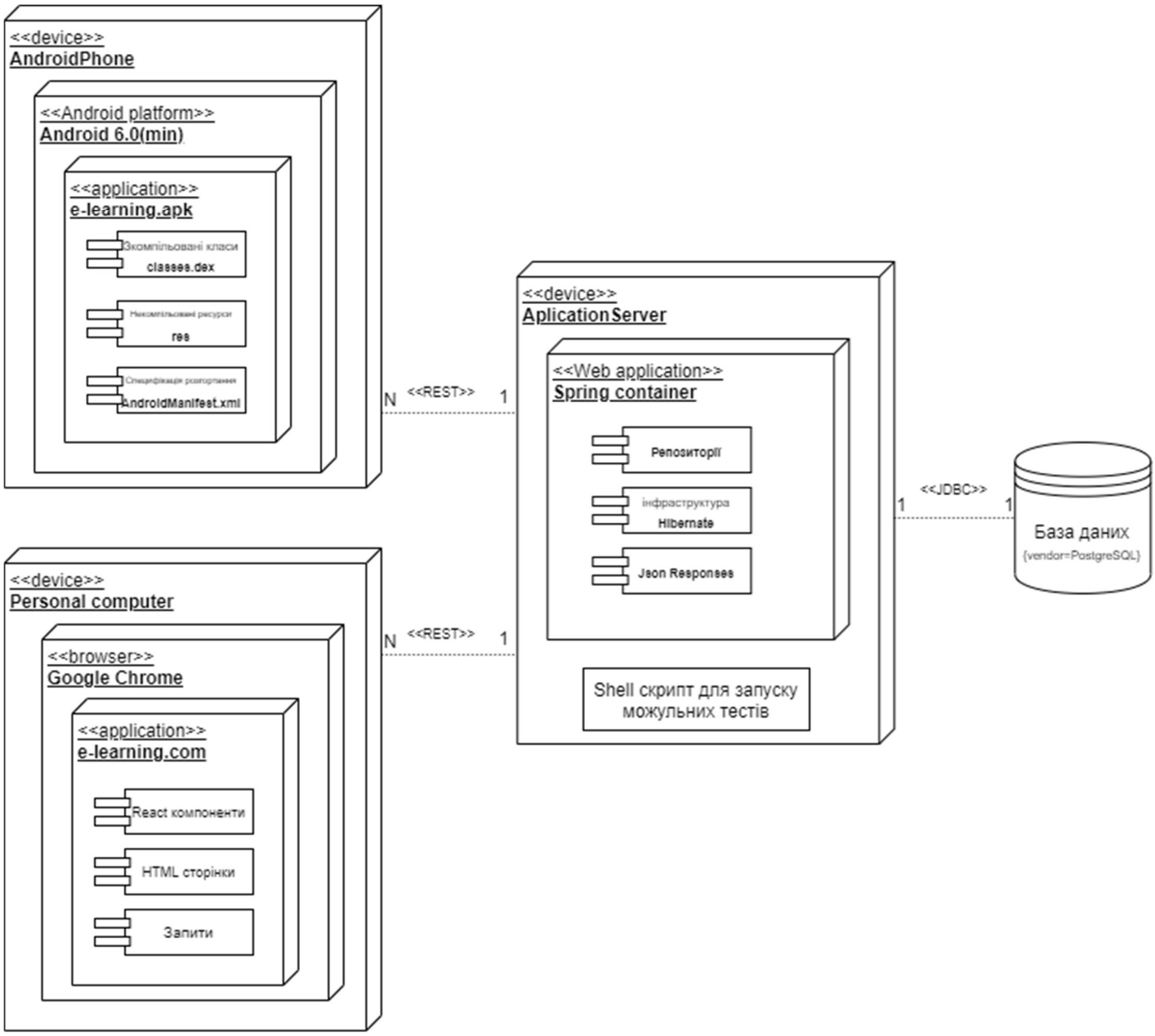
3.2.2 Діаграма розгортання

Іншим етапом проектування стало створення діаграми розгортання. Діаграма розгортання - це тип діаграми UML, який показує архітектуру виконання системи, включаючи такі вузли, як середовища виконання апаратного чи програмного забезпечення, та проміжне програмне забезпечення, що їх з'єднує.

Діаграми розгортання зазвичай використовуються для візуалізації фізичного обладнання та програмного забезпечення системи. За його допомогою ви можете зрозуміти, як система буде фізично розгорнута на апаратному забезпеченні.

Діаграми розгортання допомагають моделювати апаратну топологію системи порівняно з іншими типами діаграм UML, які в основному окреслюють логічні компоненти системи.

В результаті представлена діаграма розгортання розробленої системи(Рисунок 3.4)



3.4 – Діаграма розгортання системи

На спроектованій діаграмі, можна побачити, що система розгорталася на трьох основних пристроях:

- Сервер;
- Персональний комп’ютер;
- Android смартфон.

Основним пристроєм, без якого програма не буде працювати, є сервер. Там зберігається програма, що оперує серверною частиною та управлінням базою даних. Також кожен користувач може запустити програму на персональному комп’ютері, перейшовши за відповідним посиланням. Крім цього, у кожного є можливість завантажити мобільний додаток на власний Android смартфон та вільно користуватись програмою.

3.2.3 Проектування бази даних

Проектування бази даних визначається як сукупність етапів, які допомагають розробляти, створювати, впроваджувати та підтримувати системи управління даними системи. Основною метою проектування бази даних є створення фізичних та логічних моделей конструкцій для запропонованої системи баз даних.

Хороший процес проектування бази даних регулюється певними правилами. Перше правило вимагає уникати зайвих даних, оскільки це витрачає простір і збільшує ймовірність несправностей та розбіжностей у базі даних. Наступне правило полягає в тому, що точність та вичерпність інформації надзвичайно важливі. Якщо база даних містить помилкову інформацію, будь-які документи, що отримують дані з такої бази даних, також включатимуть неточну інформацію. Отже, будь-які рішення, засновані на цих документах, будуть вводити в оману, тим самим збільшуючи важливість дизайну бази даних, який відповідає усім вищезазначеним правилам.

Отже, добре розроблена база даних - це та, яка:

- Розподіляє дані у таблиці на основі конкретних предметних областей, щоб зменшити надмірність даних;
- Надає інформацію, необхідну для зв’язку даних у таблицях;
- Забезпечує підтримку та гарантує точність та надійність даних;
- Задовольняє вимоги щодо обробки інформації та звітності.

З урахуванням цих правил була розроблена схема бази даних(Рисунок 3.5)

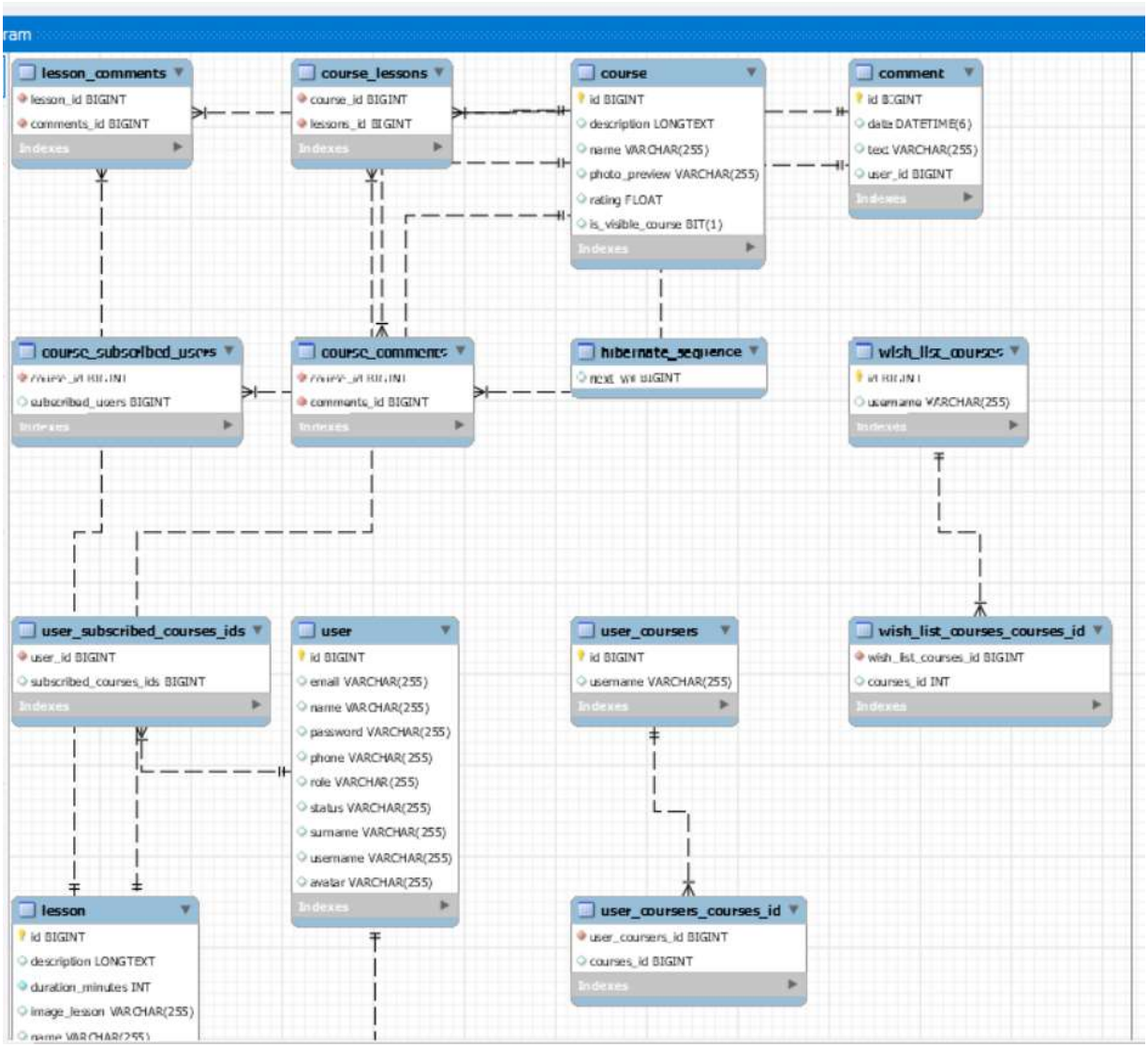


Рисунок 3.5 – Схема бази даних, розробленої системи

3.2.4 Діаграма моделі сутностей

Також на основі діаграми бази даних була побудована діаграма моделі сутностей(Рисунок 3.6)

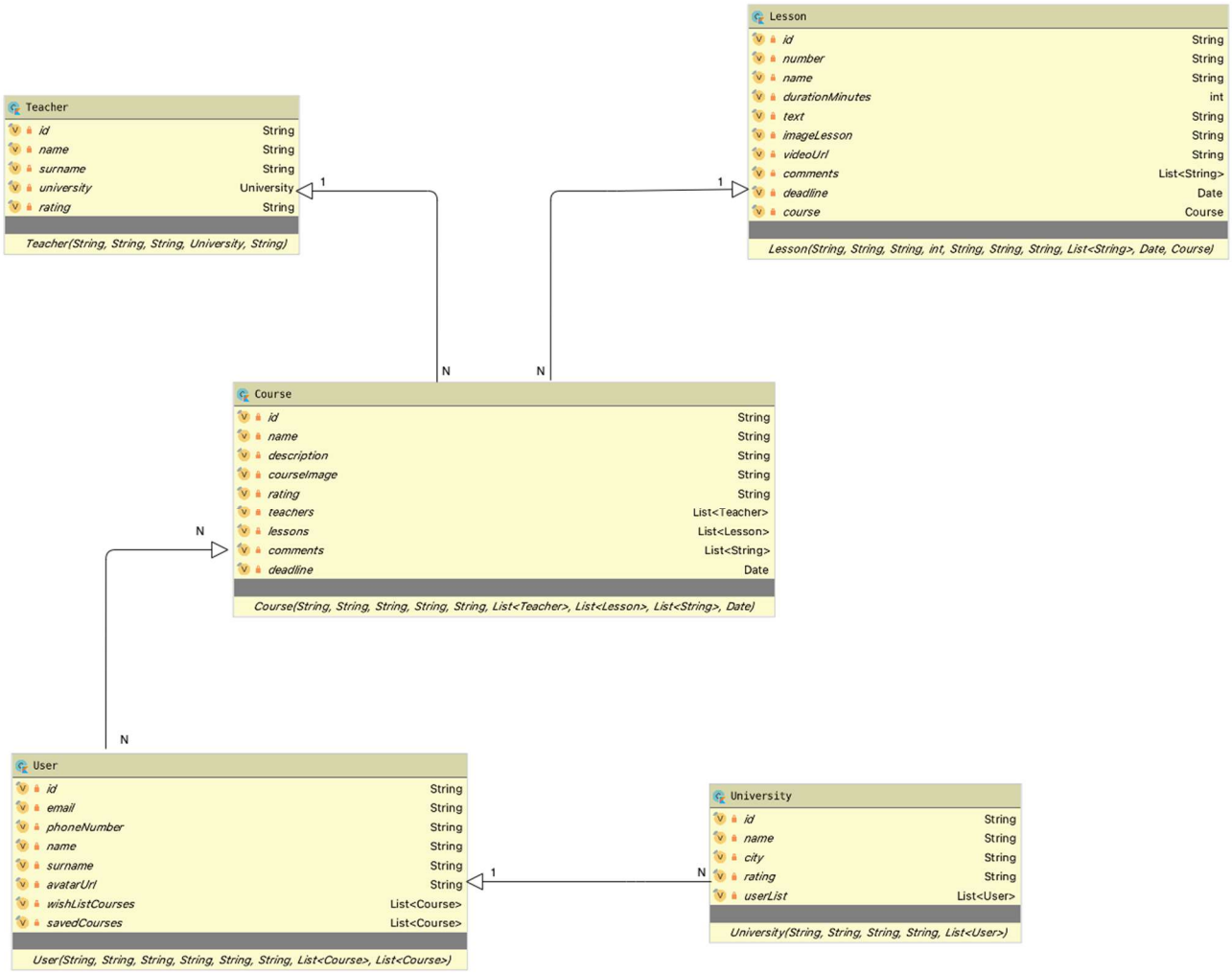


Рисунок 3.6 – Діаграма моделі сутностей

3.3 Тестування

Тестування програмного забезпечення – це процес перевірки відповідності заявлених до продукту вимог і реально реалізованої функціональності, здійснюваний шляхом спостереження за його роботою в штучно створених ситуаціях і на обмеженому наборі тестів, обраних певним чином[6].

Може оцінюватись:

- відповідність вимогам, якими керувалися проектувальники та розробники;
- правильна відповідь для усіх можливих вхідних даних
- виконання функцій за прийнятний час
- практичність
- сумісність з програмним забезпеченням та операційними системами відповідність задачам замовника.

Існує багато типів тестування програмного забезпечення, що відповідають за перевірку різних можливостей програми. У розробленому застосунку використовувалися такі типи тестування, як White Box тестування, Black Box тестування, модульне тестування, та інтеграційне тестування.

3.3.1 White Box тестування

White Box тестування – це техніка тестування програмного забезпечення, при якій внутрішня структура, дизайн та кодування програмного забезпечення тестуються для перевірки потоку вхідних даних та покращення дизайну, зручності використання та безпеки. У тестуванні білої коробки код видно тестувальникам, тому його також називають тестуванням Clear box, тестуванням у відкритих коробках, тестуванням прозорих коробок, тестуванням на основі коду та тестуванням Glass box. Це одна з двох частин підходу Box Testing до тестування програмного забезпечення. Його аналог, тестування Black Box, включає тестування з точки зору зовнішнього або кінцевого користувача. З іншого боку, тестування «White Box» в програмній інженерії базується на внутрішній роботі програми та обертається навколо внутрішнього тестування.

3.3.2 Black Box Тестування

Black Box тестування, також відоме як поведінкове тестування, - це метод тестування програмного забезпечення, при якому внутрішня структура, дизайн, реалізація предмета, що тестується, невідома тестувальнику. Ці тести можуть бути функціональними або нефункціональними, хоча зазвичай є функціональними.

Цей метод названий так, тому що програма в очах тестувальника схожа на чорний ящик; всередині якого нічого не видно. Цей метод намагається знайти помилки в таких категоріях:

- Неправильні або відсутні функції;
- Помилки інтерфейсу;
- Помилки в структурах даних або доступі до зовнішньої бази даних;
- Помилки поведінки або продуктивності;
- Помилки ініціалізації та припинення.

3.3.3 Модульне тестування

Модульне тестування, також відоме як тестування компонентів, – це рівень тестування програмного забезпечення, де тестуються окремі блоки – компоненти програмного забезпечення. Мета полягає в тому, щоб перевірити, чи працює кожна одиниця програмного забезпечення за задумом.

Модуль – це найменша частина будь-якого програмного забезпечення. Зазвичай він має один або кілька входів і, як правило, один вихід. У процедурному програмуванні модулем може бути окрема програма, функція, процедура тощо. При об’єктно-орієнтованому програмуванні найменшою одиницею є метод, який може належати до базового суперкласу, абстрактного класу або похідного дочірнього класу. Зазвичай дефекти виправляються, як тільки їх виявляють, і вони офіційно не повідомляються та не відстежуються.

У проекті модульне тестування проводилося за допомогою фреймворку Junit 5 та Mockito(рисунок 3.7).

Рисунок 3.7 Логотип фреймворку модульного тестування Junit 5

4.3.4 Інтеграційне тестування

Інтеграційне тестування – це етап тестування програмного забезпечення, в якому окремі програмні модулі поєднуються та перевіряються як група.

Інтеграційне тестування проводиться для оцінки відповідності системи або компонента заданим функціональним вимогам. Це відбувається після модульного тестування та перед тестуванням для перевірки. Інтеграційне тестування приймає за вхідні модулі, які були протестовані одинично, групує їх у більші сукупності, застосовує тести, визначені в плані інтеграційного тестування, до цих сукупностей та забезпечує як вихідну інформацію інтегровану систему, готову до тестування системи.

4 ОПИС ПРОГРАМНИХ ЗАСОБІВ

Після проведення аналізу предметної області та існуючих рішень, було вирішено розробити застосунок з клієнт-серверною архітектурою. При цьому на даному етапі розроблено дві різні програми-клієнти, що можуть запускатися як веб застосунок або мобільний застосунок відповідно. В даному розділі будуть розглянуті програмні засоби, за допомогою яких відбувалась побудова кожного з компонентів системи. Були реалізовані такі компоненти:

- Серверна програма
- Веб застосунок
- Мобільний застосунок

4.1 Серверна частина

В процесі розробки програми для серверної частини були використані сучасні та надійні засоби. В масштабі розробки серверної частини була створена модель бази даних та побудовані зв’язки для проведення операцій над базою даних. Також реалізовано API(Application Programming Interface) для спілкування серверу з клієнтами.

4.1.1 Мова програмування

У якості мови програмування була використана популярна мова для побудови складних серверних застосунків – Java(Рисунок 4.1).

Java – це високорівнева, об'єктно-орієнтована мова програмування, що базується на класах, і розроблена так, щоб мати якомога менше залежностей від реалізації. Це мова програмування загального призначення, призначена для того, щоб розробники програм могли писати один раз, запускати їх де завгодно (WORA), що означає, що скомпільований код Java може працювати на всіх платформах, що підтримують Java, без необхідності перекомпіляції. Програми Java зазвичай компілюються в байт-код, який може працювати на будь-якій віртуальній машині Java (JVM), незалежно від базової архітектури комп'ютера[7].

Рисунок 4.1 – Логотип мови програмування Java

Java використовує автоматичний збирач сміття для управління пам’яттю в життєвому циклі об’єкта. Програміст визначає, коли створюються об'єкти, а час виконання Java відповідає за відновлення пам'яті, коли об'єкти більше не використовуються. Як тільки посилань на об'єкт не залишаються, недоступна пам'ять стає придатною для автоматичного звільнення збирачем сміття. Щось подібне до витоку пам’яті все ще може статися, якщо код програміста містить посилання на об’єкт, який більше не потрібен, як правило, коли об’єкти, які більше не потрібні, зберігаються в контейнерах, які все ще використовуються. Якщо викликаються методи для неіснуючого об'єкта, викидається нульовий виняток покажчика.

На синтаксис Java значною мірою впливають мови C ++ та C. На відміну від C ++, який поєднує синтаксис для структурованого, загального та об'єктно-орієнтованого програмування, Java будувалася майже виключно як об'єктно-орієнтована мова. Весь код пишеться всередині класів, і кожен елемент даних є об'єктом, за винятком примітивних типів даних (тобто цілих чисел, чисел з плаваючою комою, булевих значень та символів), які не є об'єктами з міркувань продуктивності.

Саме через свою зручність, легкість у засвоєнні, та, що найголовніше, надійність Java є дуже популярним засобом розробки серверних частин складних веб застосунків.

4.1.2 Фреймворк для побудови серверного веб застосунку

Наразі існує небагато зручних веб фреймворків, що використовують мову Java. Безперечні лідерські позиції в цій сфері займає сімейство фреймворків Spring. Саме вони були використані для побудови веб застосунку. Перший з них є Spring Boot - це проект, який побудований на вершині Spring Framework. Він забезпечує простіший і швидший спосіб налаштування, налаштування та запуску як простих, так і складних веб-програм.

Це модуль Spring, який забезпечує функцію RAD (швидка розробка додатків) для Spring Framework. Він використовується для створення окремого додатку на основі Spring, який ви можете просто запустити, оскільки йому потрібна мінімальна конфігурація Spring. Загалом, Spring Boot – це поєднання Spring Framework та вбудованих серверів.

У Spring Boot немає вимог до конфігурації XML (дескриптор розгортання). Він використовує домовленість щодо парадигми дизайну програмного забезпечення конфігурації, що означає зменшення зусиль розробника.

Рисунок 4.2 – Логотип Spring Framework

Інший фреймворк, який хотілось би виділити, має назву Spring Security. Spring Security - це потужна система автентифікації та контролю доступу, що легко налаштовується. Це фактичний стандарт захисту програм на основі Spring.

Spring Security - це структура, яка зосереджена на забезпеченні автентифікації та авторизації програм Java. Як і всі проекти Spring, справжня сила Spring Security полягає в тому, як легко її можна розширити, щоб задовольнити власні вимоги.

Особливості фреймворку:

- Комплексна і розширювана підтримка як для автентифікації, так і для авторизації;
- Захист від атак, таких як фіксація сеансу, розбивка кліків, підробка міжсайтових запитів тощо
- Інтеграція API сервлетів
- Необов’язкова інтеграція з Spring Web MVC

Також варто відмітити ще один фреймворк – Spring MVC. Оскільки в застосунку використовується MVC(Model-View-Controller) архітектура, цей фреймворк чудово підходить для допомоги у розробці такої архітектури.

MVC - це шаблон проектування програмного забезпечення, який зазвичай використовується для розробки користувацьких інтерфейсів, що розділяє відповідну логіку програми на три взаємопов'язані елементи. Це робиться для відокремлення внутрішнього подання інформації від способів подання та прийняття інформації від користувача[8].

Основні компоненти(Рисунок 4.3):

Модель. Центральний компонент візерунка. Це динамічні структуру даних програми, незалежні від інтерфейсу користувача. Вона безпосередньо управляє даними, логікою та правилами програми.

Відображення. Будь-яке подання інформації, такої як діаграма або таблиця. Можливі декілька подань однієї і тієї ж інформації, таких як гістограма для управління або таблиця для бухгалтерів.

Контролер. Приймає введені дані та перетворює їх на команди для Моделі чи Відображення.

Додатково до розподілу програми на ці компоненти, шаблон MVC визначає взаємодію між ними:

- Модель відповідає за управління даними програми. Вона отримує дані користувача від контролера.
- Відображення робить презентацію моделі в певному форматі.

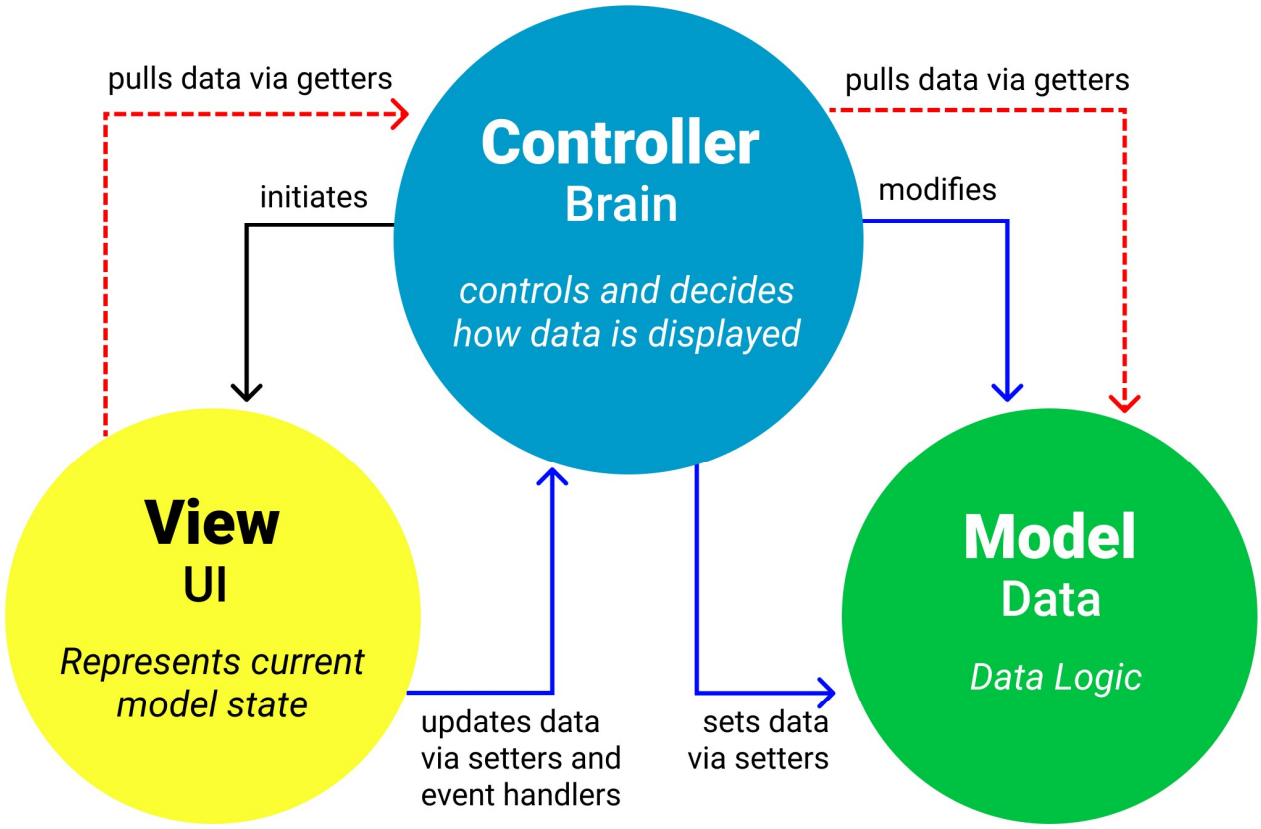


Рисунок 4.3 – Графічне представлення шаблону MVC

Фреймворк Spring MVC, по суті, реалізує даний шаблон, що дозволяє використовувати вже готовий та надійний підхід до розробки ПЗ.

4.1.3 Система керування базами даних

Система керування базами даних (СКБД) - це програмне забезпечення, яке взаємодіє з кінцевими користувачами, програмами та самою базою даних для збору та аналізу даних. Програмне забезпечення СКБД додатково охоплює основні засоби, що

надаються для адміністрування бази даних. В загальному бази даних, СКБД та пов'язані з нею додатки можна об’єднати під однією назвою "система баз даних". Часто термін "база даних" також використовується для вільного посилання на будь-яку СКБД, систему баз даних або додаток, пов'язаний з базою даних[9].

Після проведення аналізу було вирішено використовувати MySQL систему керування базами даних, адже вона є досить надійною, перевіреною, та задовольняє усі вимоги, що були визначені для системи.

MySQL – це система управління реляційними базами даних з відкритим кодом. Він базується на структурі мови запитів (SQL), яка використовується для додавання, видалення та модифікації інформації в базі даних. Стандартні команди SQL, такі як ADD, DROP, INSERT та UPDATE, можна використовувати з MySQL(Рисунок 4.4).

MySQL може використовуватися для різноманітних програм, але найчастіше його можна знайти на веб-серверах. Веб сайт, який використовує MySQL, може включати веб сторінки, які отримують доступ до інформації з бази даних. Ці сторінки часто називають "динамічними", тобто вміст кожної сторінки генерується з бази даних під час завантаження сторінки. Веб-сайти, які використовують динамічні веб сторінки, часто називають веб сайтами, керованими базами даних.

Рисунок 4.4 – Логотип MySQL

4.1.4 Середовище розробки Intellij Idea

IntelliJ IDEA – комерційне інтегроване середовище розробки для різних мов програмування (Java, Python, Scala, PHP та ін.) від компанії JetBrains. Система поставляється у вигляді урізаної по функціональності безкоштовної версії «Community Edition» і повнофункціональної комерційної версії «Ultimate Edition», для якої активні розробники відкритих проектів мають можливість отримати безкоштовну ліцензію. Сирцеві тексти Community-версії поширюються рамках ліцензії Apache 2.0. Бінарні збірки підготовлені для Linux, Mac OS X і Windows(рисунок 4.5).

Кожен компонент IntelliJ IDEA створений для того, щоб максимально підвищити продуктивність розробки. Розумний редактор коду в поєднанні з ергономічним дизайном роблять розробку не тільки ефективною, а й приємною. Після індексування вихідного коду IntelliJ IDEA надає масу можливостей для швидкої і ефективної розробки: розумне автодоповнення, аналіз коду в реальному часі і надійні рефакторингом. IntelliJ IDEA не тільки допомагає писати код в редакторі, але і спрощує роботу в цілому. Ви зможете легко і швидко заповнити поле, знайти елемент у списку, відкрити потрібне вікно, поміняти настройки і т.д.

При побудові програмного комплексу середовище розробки Intellij IDEA використовувалося для побудови серверної частини застосунку.

Рисунок 4.5 – Логотип Intellij IDEA

4.1.5 Середовище управлінням баз даних MySQL Workbench

Задля зручної роботи з базами даних, а саме візуалізація та додавання нових даних була використана спеціальна програма, розроблена компанією Microsoft, MySQL Workbench.

MySQL Workbench - це візуальний інструмент проектування баз даних, який інтегрує розробку, адміністрування, проектування, створення та обслуговування SQL в єдине цілісне середовище розробки системи баз даних MySQL. Цей інструмент надає наступні можливості:

1) Підключення до бази даних та управління екземплярами;
2) Перегляд, перевірка та пошук об’єктів бази даних, таких як таблиці та поля;
3) Можливість створення діаграм моделей;
4) Створення та редагування структур баз даних;
5) Заповнення бази тестовими даними та ін.

4.2 Веб застосунок

В процесі розробки програмного забезпечення для веб застосунку потрібно було виконати наступні задачі:

- Побудувати дизайн макетів сторінок веб порталу;
- Зверстати сторінки за розробленими шаблонами;
- Розробити веб застосунок, що отримує дані з серверу та динамічно відображає їх на сторінках веб сайту.

Для цього використовувалися сучасні засоби та методи розробки.

4.2.1 Засоби побудови веб застосунку

Головною мовою програмування для розробки веб застосунку є JavaScript. Це одна з найпопулярніших мов програмування у світі, адже вона досить потужна для створення складних програм, але при цьому досить легка в засвоєнні.

JavaScript - це скриптова мова програмування, що дозволяє впроваджувати на веб-сторінках складні функції. Таким чином, веб-сторінка не просто відображає статичну інформацію, а й відображає своєчасне оновлення вмісту, інтерактивні карти, анімовані 2D / Тривимірна графіка, прокрутку відео тощо. Якщо ви бачите такі елементи на веб сайті, можете бути певні, що, ймовірно, на сторінці задіяний JavaScript. Це третій шар шару стандартних веб-технологій(інші два HTML та CSS будуть розглянуті у наступних підрозділах).

Для верстки макетів інтерфейсу були використані HTML та CSS. HTML (Hyper Text Markup Language) є найосновнішим будівельним блоком Інтернету. Він визначає значення та структуру веб-вмісту. Інші технології, крім HTML, зазвичай використовуються для опису зовнішнього вигляду / презентації веб-сторінки (CSS) або функціональності / поведінки (JavaScript).

"Гіпертекст" відноситься до посилань, які з'єднують веб-сторінки між собою, або в межах одного веб-сайту, або між веб-сайтами. Посилання є основним аспектом Інтернету. Завантажуючи вміст в Інтернет і посилаючи його на сторінки, створені іншими людьми, ви стаєте активним учасником Всесвітньої павутини.

Каскадні таблиці стилів (CSS) — це мова таблиць стилів, що використовується для опису презентації документа, написаного HTML або XML (включаючи діалекти XML, такі як SVG, MathML або XHTML). CSS описує, як елементи повинні відображатися на екрані, на папері, в мові чи на інших носіях.

CSS є однією з основних мов відкритого Інтернету і стандартизований для веб-браузерів відповідно до специфікацій W3C. Раніше розробка різних частин специфікації CSS здійснювалась синхронно, що дозволяло встановлення версій останніх рекомендацій. Можливо, ви чули про CSS1, CSS2.1, CSS3. Однак CSS4 ніколи не став офіційною версією.

Починаючи з CSS3, обсяг специфікації значно збільшився, і прогрес у різних модулях CSS почав настільки різнитися, що стало ефективніше розробляти та випускати

рекомендації окремо для кожного модуля. Замість версії специфікації CSS, W3C тепер періодично робить знімок останнього стабільного стану специфікації CSS. JavaScript відповідає за анімацію та опрацьовує дані на боці користувача. Разом ці технології створюють веб-сайт(Рисунок 4.6).

Рисунок 4.6 – Головні засоби побудови веб сторінки

4.2.2 Допоміжні технології

React.js — це бібліотека JavaScript з відкритим кодом, інтерфейс, для створення інтерфейсів користувача або компонентів інтерфейсу(Рисунок 4.7). Він підтримується Facebook та спільнотою окремих розробників та компаній. React можна використовувати як основу при розробці односторінкових або мобільних додатків.

React створює кеш-структуру даних в пам’яті, обчислює отримані різниці, а потім ефективно оновлює відображуваний DOM браузера. Цей процес називається примиренням. Це дозволяє програмісту писати код так, ніби вся сторінка відображається при кожній зміні, тоді як бібліотеки React відображають лише підкомпоненти, які насправді змінюються. Цей вибірковий візуалізація забезпечує значне підвищення продуктивності. Це економить зусилля з перерахунку стилю CSS, макета сторінки та рендерингу для всієї сторінки.

Однак React займається лише управлінням станом і наданням цього стану DOM, тому для створення програм React зазвичай потрібно використовувати додаткові бібліотеки для маршрутизації організації потоку даних. Деякі з таких бібліотек, що були використані у застосунку стали React Router та Redux.

React Router - це сукупність навігаційних компонентів, які декларативно складаються з вашим додатком. Простіше кажучи – це бібліотека, яка надає можливість дуже зручно керувати переходами і правильно налаштовувати URL маршрутизацію. В проекті, що розроблявся, ця бібліотека використовувалася по всій системі, тому це стало майже незамінним засобом розробки.

Redux – це передбачуваний контейнер стану для програм JavaScript.

Це допомагає писати програми, які поводяться послідовно, запускти код у різних середовищах (клієнтських, серверних та власних) і робить програми легкими для тестування. Крім того, це забезпечує чудовий досвід для розробників, такий як редагування коду в реальному часі в поєднанні з налагоджувачем часу.

Ви можете використовувати Redux разом з React або з будь-якою іншою веб бібліотекою.

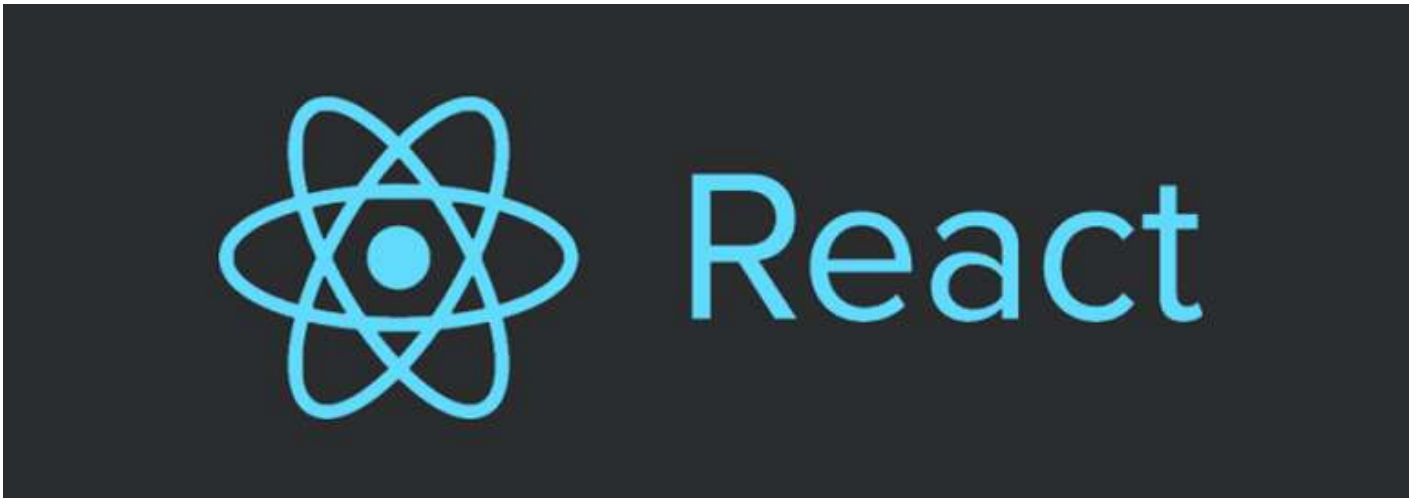


Рисунок 4.7 – Логотип фреймворку React.js

Враховуючи усі переваги, що надає React, цей фреймворк став зручним інструментом для побудови front-end частини проекту.

4.2.3 Середовище розробки Visual Studio Code

Visual Studio Code — засіб для створення, редагування та зневадження сучасних вебзастосунків і програм для хмарних систем. Visual Studio Code розповсюджується безкоштовно і доступний у версіях для платформ Windows, Linux і OS X.

Редактор містить вбудований зневаджувач, інструменти для роботи з Git і засоби рефакторингу, навігації по коду, автодоповнення типових конструкцій і контекстної підказки. Продукт підтримує розробку для платформ ASP.NET і Node.js, і позиціюється як легковагове рішення, що дозволяє обійтися без повного інтегрованого середовища розробки. Серед підтримуваних мов і технологій: JavaScript, C++, C#, TypeScript, jade, PHP, Python, XML, Batch, F#, DockerFile, Coffee Script, Java, HandleBars, R, Objective-C, PowerShell, Luna, Visual Basic, Markdown, JSON, HTML, CSS, LESS і SASS, Haxe(Рисунок 4.8).

При побудові програмного коплексу Visual Studio Code використовувався для побудови фронт-енд частини.

Рисунок 4.8 – Логотип Visual Studio Code

4.3 Мобільний застосунок

Мобільний застосунок був розроблений на мові програмування Kotlin для операційної системи Android.

4.3.1 Операційна система Android

Android – це мобільна операційна система, заснована на модифікованій версії ядра Linux та іншого програмного забезпечення з відкритим кодом, призначена в основному для сенсорних мобільних пристроїв, таких як смартфони та планшети(Рисунок 4.9).

Це безкоштовне програмне забезпечення з відкритим кодом; його вихідний код відомий як Android Open-Source Project (AOSP), який в основному ліцензується за ліцензією Apache. Однак більшість пристроїв Android постачаються з попередньо встановленим власним програмним забезпеченням, зокрема Google Mobile Services (GMS), яке включає такі основні програми, як Google Chrome, платформа цифрового розповсюдження Google Play та пов'язана з ними платформа розробки служб Google Play. Близько 70 відсотків смартфонів Android управляють екосистемою Google.

Рисунок 4.9 – Логотип операційної системи Android

4.3.2 Мова програмування Kotlin

В свою чергу, Kotlin – це крос-платформна, статично набрана мова загального призначення з програмуванням типу(Рисунок 4.10). Kotlin розроблений для повноцінного взаємодії з Java, а версія JVM стандартної бібліотеки Kotlin залежить від бібліотеки класів Java, але умовивід типу дозволяє його синтаксису бути більш стислим. Kotlin головним чином орієнтована на JVM, але також компілюється в JavaScript (наприклад, для інтерфейсних веб-додатків, що використовують React) або власний код (через LLVM); наприклад, для власних додатків iOS, що діляться бізнес-логікою з додатками Android.

Рисунок 4.10 – Логотип мови програмування Kotlin

Використовуючи Kotlin для розробки Android застосунку можна отримати такі переваги:

- Менше коду в поєднанні з більшою читабельністю. Витрачайте менше часу на написання коду та працю, щоб зрозуміти код інших.
- Зріла мова та середовище. З часу свого створення в 2011 році Котлін постійно розвивався не лише як мова, але й як ціла екосистема із надійними інструментами. Тепер він легко інтегрується в Android Studio і активно використовується багатьма компаніями для розробки програм для Android.
- Підтримка Kotlin в Android Jetpack та інших бібліотеках. Розширення KTX додають до існуючих бібліотек Android функції мови Kotlin, такі як програми, функції розширення, лямбди та іменовані параметри.
- Взаємодія з Java. Ви можете використовувати Kotlin разом із мовою програмування Java у своїх програмах, не потребуючи міграції всього коду до Kotlin.
- Підтримка розробки багатоплатформенних систем. Ви можете використовувати Kotlin для розробки не тільки Android, а й iOS, серверних платформ та веб-додатків. Насолоджуйтесь перевагами спільного використання спільного коду між платформами.

- Безпека коду. Менше коду та краща читабельність призводять до меншої кількості помилок. Компілятор Kotlin виявляє ці залишки помилок, роблячи код безпечним.
- Легке навчання. Kotlin дуже простий у вивченні, особливо для розробників Java.
- Велика спільнота. Котлін має велику підтримку та багато внесків від громади, яка зростає у всьому світі. За даними Google, понад 60% із 1000 найкращих додатків у Play Store використовують Kotlin.

Виходячи з вищенаведених переваг, Koltin став ідеальним засобом розробки мобільного застосунку.

4.3.3 Середовище розробки Android Studio

Android Studio — інтегроване середовище розробки (IDE) для платформи Android, представлене 16 травня 2013 року на конференції Google I/O менеджером по продукції корпорації Google — Еллі Паверс (англ. Ellie Powers). 8 грудня 2014 року компанія Google випустила перший стабільний реліз Android Studio 1.0.

Середовище розробки адаптоване для виконання типових завдань, що вирішуються в процесі розробки застосунків для платформи Android. У тому числі у середовище включені засоби для спрощення тестування програм на сумісність з різними версіями платформи та інструменти для проектування застосунків, що працюють на пристроях з екранами різної роздільності (планшети, смартфони, ноутбуки, годинники, окуляри тощо). Крім можливостей, присутніх в IntelliJ IDEA, в Android Studio реалізовано кілька додаткових функцій, таких як нова уніфікована підсистема складання, тестування і розгортання застосунків, заснована на складальному інструментарії Gradle і підтримуюча використання засобів безперервної інтеграції.

Для прискорення розробки застосунків представлена колекція типових елементів інтерфейсу і візуальний редактор для їхнього компонування, що надає зручний попередній перегляд різних станів інтерфейсу застосунку (наприклад, можна

подивитися як інтерфейс буде виглядати для різних версій Android і для різних розмірів екрану). Для створення нестандартних інтерфейсів присутній майстер створення власних елементів оформлення, що підтримує використання шаблонів.

При побудові програмного комплексу середовище розробки Android Studio використовувалося для побудови мобільного застосунку.

5 РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

Система організації доступу до e-learning ресурсів розроблена для використання як на персональних комп’ютерах за допомогою веб браузера, так і на мобільних пристроях, шо працюють на операційній системі. При цьому вимогою до кожного пристрою є активне підключення до глобальної мережі інтернет.

5.1 Інсталяція програми та системні вимоги

5.1.1 Веб-портал

Веб застосунок буде розгортаний на визначеному хостингу. Тому, щоб користувачу запустити програму, потрібно відкрити будь-який веб браузер та перейти за наданим посилання. Користувач побачить головну сторінку з доступними курсами(Рисунок 5.1). Щоб почату роботу саме як викладач, йому необхідно авторизуватись у системі. Програма сама визначить, що авторизований користувач є саме викладачем і надасть відповідні можливості та особливості, що визначені лише для викладача.

Аби застосунок працював коректно, необхідно використовувати сучасні веб браузери. Серед них: Chrome, Firefox, Safari та інші.

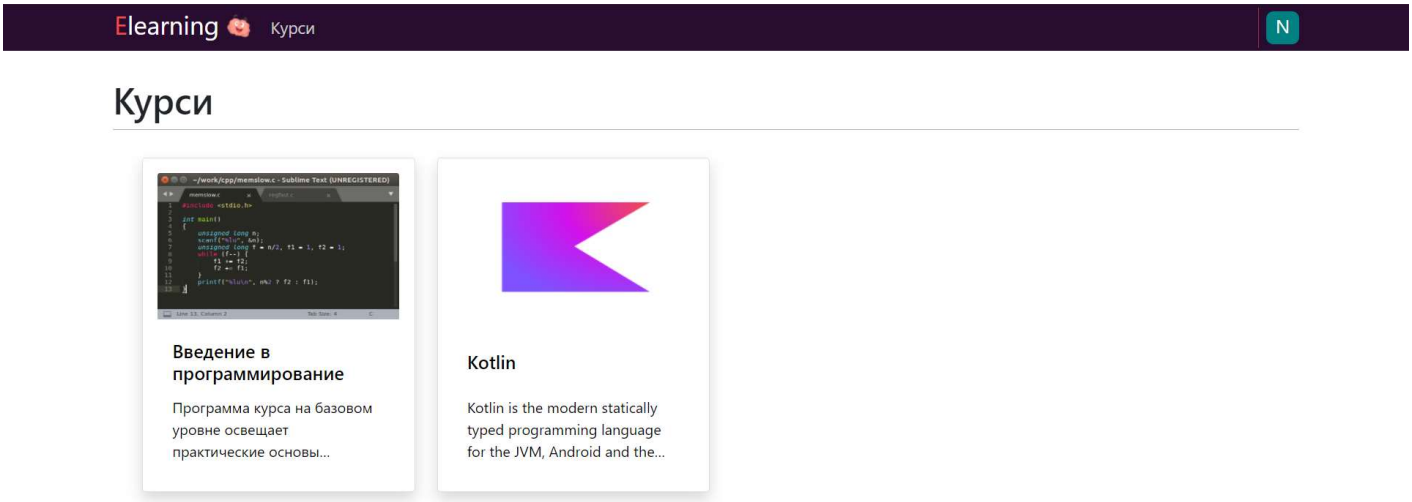


Рисунок 5.1 – Головна сторінка веб порталу “e-learning”

5.1.2 Мобільний застосунок

Щоб запустити мобільний застосунок, потрібно встановити на свій смартфон файл з розширенням «.apk». Мобільний застосунок має навігаційну панель, де користувач може обрати один з екранів «Новини», «Курси» або «Мій акаунт»(Рисунок 5.2).

Застосунок може бути запущений тільки на системі Android. Мінімальна робоча версія системи Android повинна становити не менше ніж Android 6.0 Marshmallow.

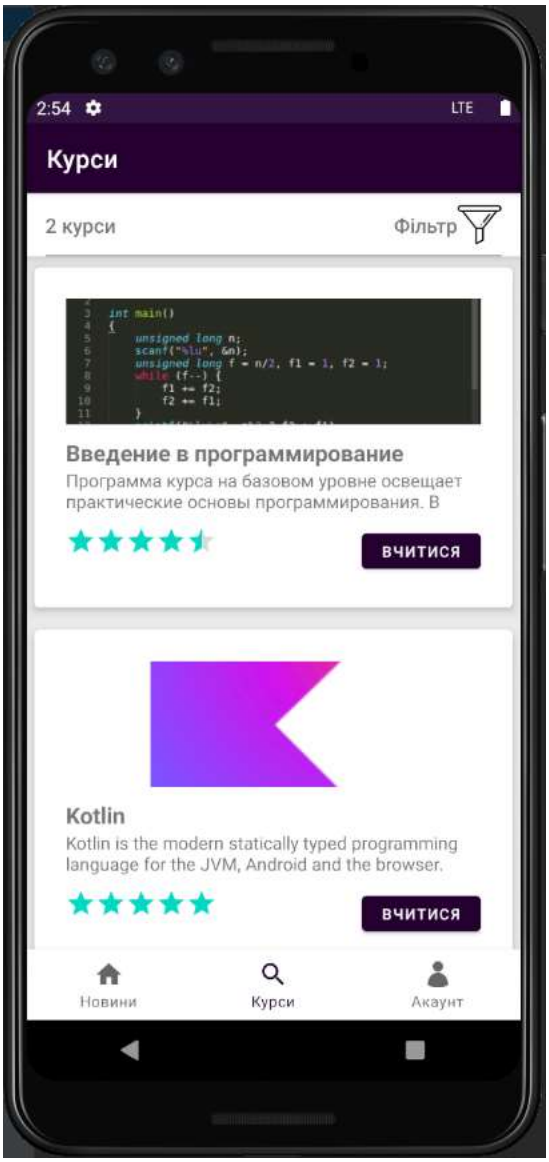


Рисунок 5.2 – Мобільний застосунок e-learning

5.2 Сценарії використання програмного продукту

В загальному в описаній програмній системі викладач може мати такі можливості:

- Авторизація
- Переглядання профілю
- Зміна паролю
- Створення курсу
- Додавання уроків до курсу

- Проставляння балів студентам за домашнє завдання.

5.2.1 Авторизація

Отже, зайшовши на веб-портал, викладачу необхідно авторизуватись. При цьому потрібно ввести власне ім’я користувача у системі і пароль(Рисунок 5.3). Це є однією з найважливіших функцій, адже неавторизованим користувачам доступ до всіх інших функцій викладача будуть недоступні.

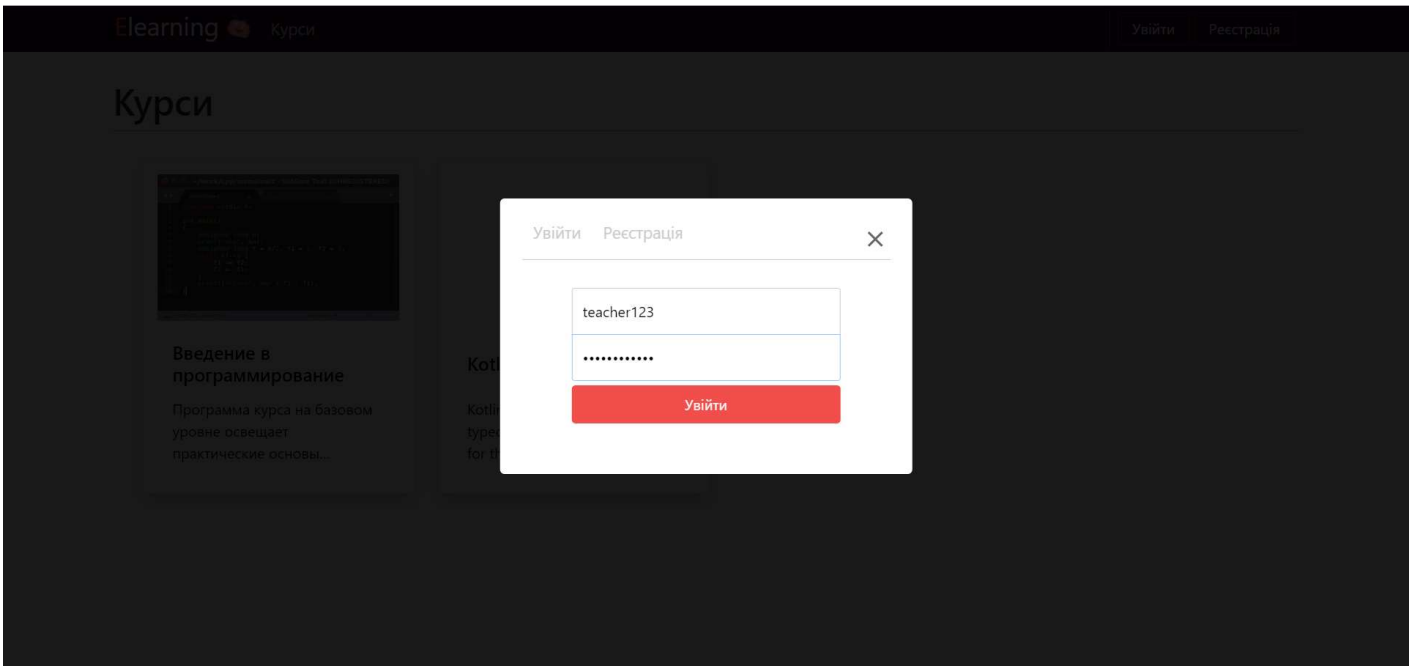


Рисунок 5.3 – Авторизація викладача

Ввівши коректні дані, користувач зможе побачити власний профіль. Інакше, якщо дані не співпадуть з даними в базі, буде показана помилка про невірний введені пароль або ім’я користувача.

5.2.2 Профіль користувача

У разі неуспішної авторизації сторінка «Профіль користувача» не буде відображатися, і користувачі не матимуть доступ до всіх представлених на даній сторінці функцій. Якщо авторизація для викладача пройшла успішно, він зможе доступитися до власного профілю(Рисунок 5.4).

На цій сторінці існує навігаційна панель, що має наступні компоненти:

1) «Опис». Тут викладач може переглядати особисті дані;
2) «Опубліковані курси». На цій сторінці відображаються всі курси, які викладач опублікував;
3) «Неопубліковані курси». Якщо курси були створені, але ще не є опублікованими, вони потрапляють на дану вкладку;
4) «Створити курс». На цій сторінці викладачу надається можливість створити новий курс;
5) «Змінити пароль». Тут викладач може змінити пароль для входу в систему;
6) «Вийти». Клікнувши дану вкладку, система видалить користувача з локального сховища браузера, користувача перекине на головну сторінку і він не матиме доступу до вищенаведених функцій.

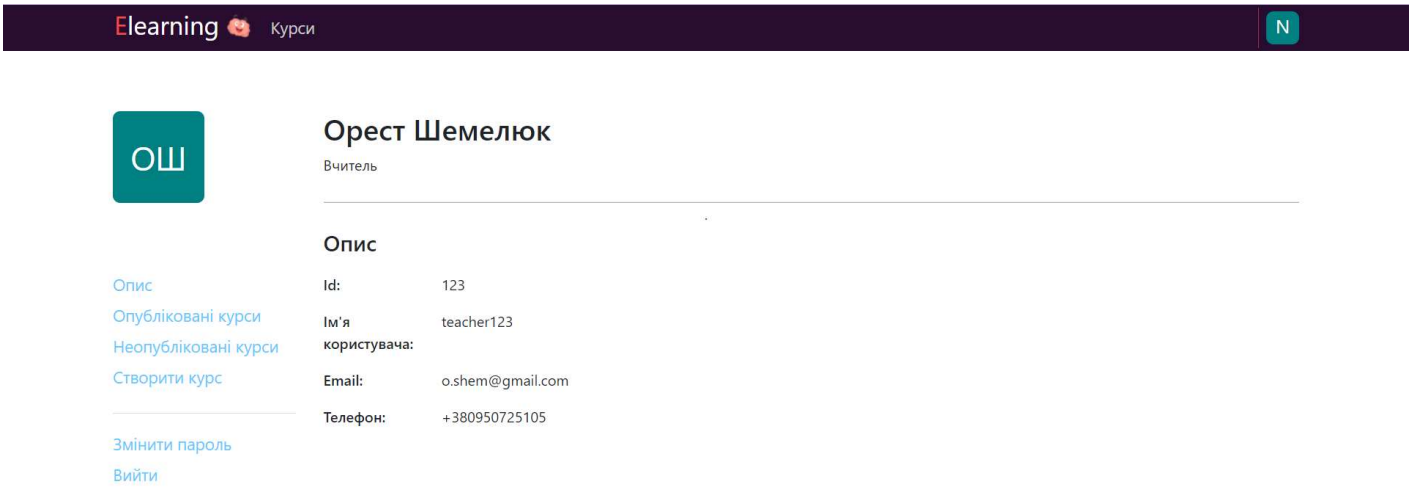


Рисунок 5.4 – профіль користувача

5.2.3 Створення курсу

Перейшовши на вкладку «Створити курс» викладач потрапляє на сторінку, де є форма для введення даних, що будуть використані системою для створення нового курсу(Рисунок 5.5).

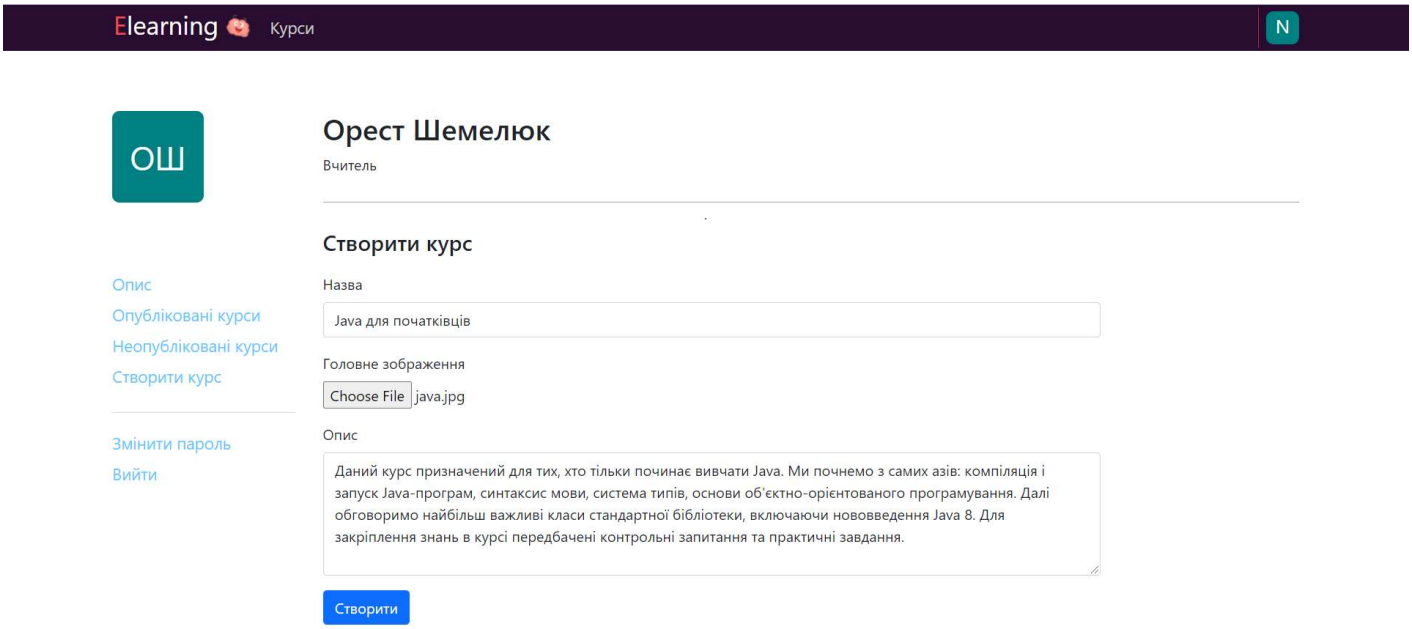


Рисунок 5.5 – Створення курсу

Видно, що для додавання нового курсу у систему потрібно ввести назву, надати картинку для головного зображення курсу, а також короткий опис, що буде відбуватись на курсі.

5.2.4 Додавання уроку до курсу

Щоб додати новий урок до курсу, користувачу потрібно перейти на вкладку «Опубліковані курси» або «Неопубліковані курси» і натиснути на курс, до якого потрібно створити новий урок. В результаті ми потрапимо на сторінку курсу (Рисунок 5.6). Звідси ми можемо натиснути на кнопку «Додати урок». Система покаже форму для створення уроку. Щоб створити урок, необхідно додати такі дані:

- Назву;
- Головне зображення;
- Короткий опис;
- Контент;
- Домашнє завдання;

- Посилання на репозиторій Github, де зберігається проєкт з класами, що потрібно реалізувати, та написаними до них модульними тестами.

Контент може містити різні типи даних, включаючи звичайний текст, який можна форматувати, створювати списки і т.д.; а також мультимедію: відео, аудіо, картинки.

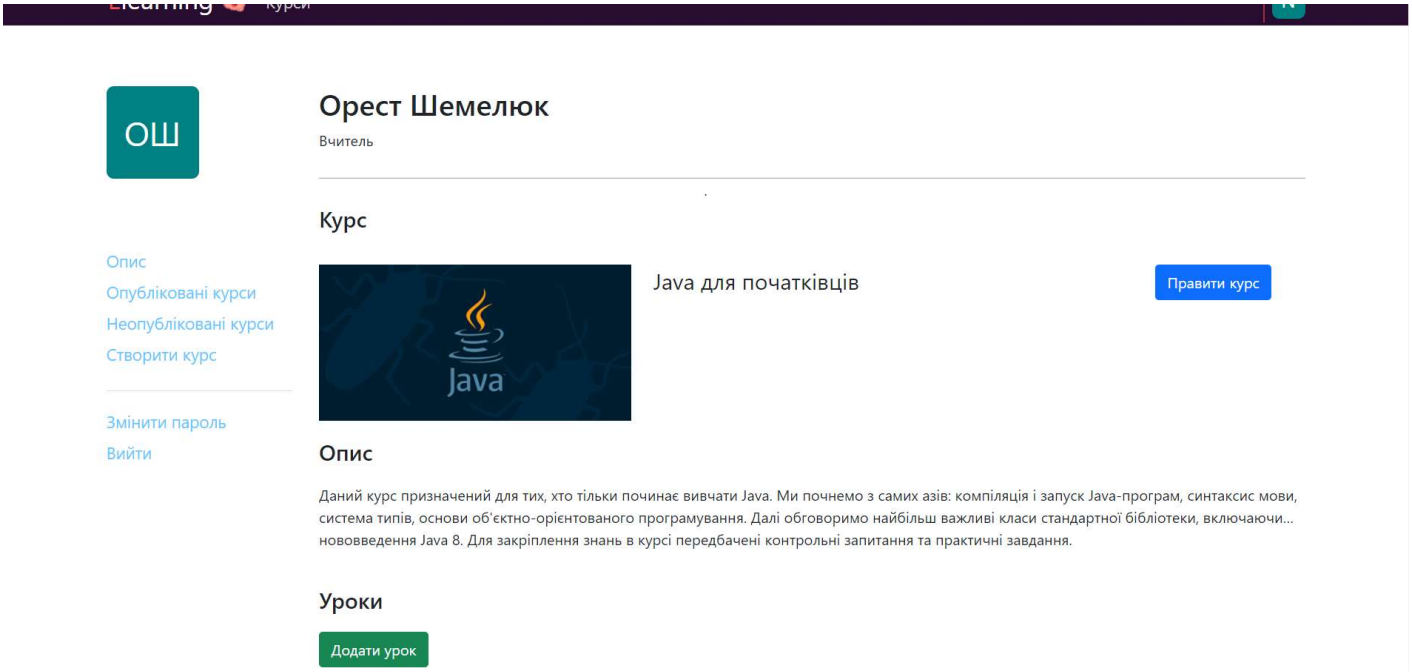


Рисунок 5.6 – Деталі курсу

5.2.5 Публікація курсу

Після того, як викладач вирішив, що курс можна публікувати, він може зробити це натиснувши на кнопку «Опублікувати» для відповідного курсу(Рисунок 5.7). Система змінить статус курсу на «Опублікований», і цей курс зможуть переглядати та брати участь всі студенти.

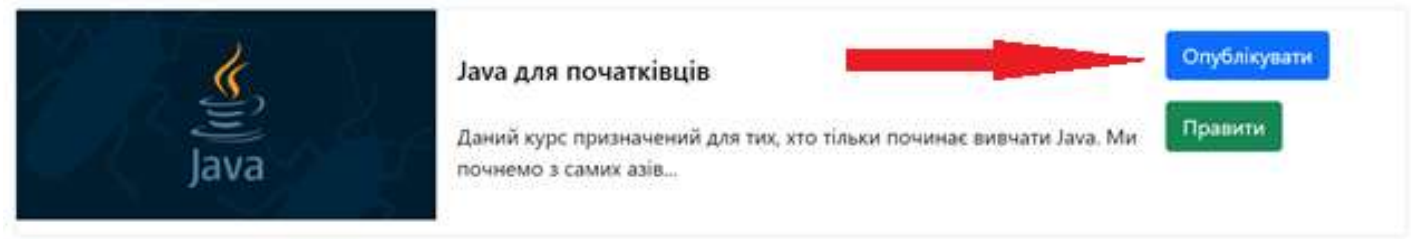


Рисунок 5.7 – Публікація курсу

5.2.6 Виставлення балів за домашнє завдання

Щоб подивитись список студентів, які задіяні в курсі на даний момент, потрібно перейти на вкладку «Опубліковані курси», а потім на деталі даного курсу. Внизу можна буде бачити посилання з текстом «Показати список студентів». Клікнувши на нього, система покаже викладачу список студентів (Рисунок 5.8).

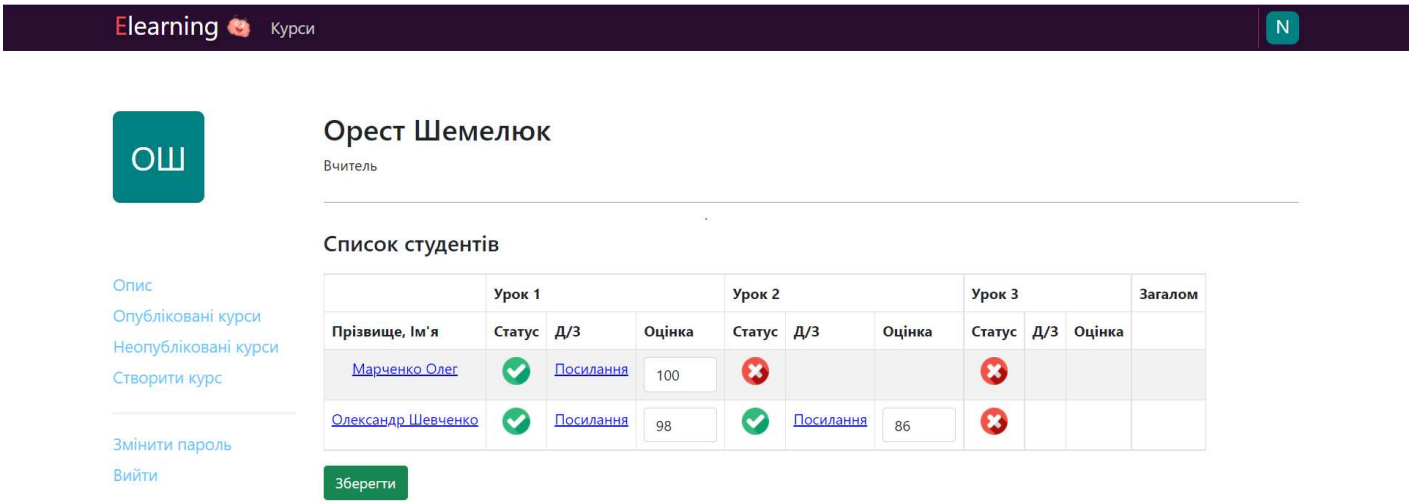


Рисунок 5.8 – Список студентів

На цьому екрані ми можемо бачити, що для кожного студента відображається:

- Статус виконання домашнього завдання до кожного уроку;
- Посилання на виконане домашнє завдання(якщо потрібно);
- Поле для виставлення оцінки за домашнє завдання.

Після завершення процесу виставлення балів, потрібно натиснути «Зберегти», і система відправить запит на сервер, щоб ці дані зберегти у базу.

ВИСНОВКИ

В результаті виконання лабораторної роботи можна зробити наступні висновки:

1. Розроблено програмний продукт для організації системи онлайн курсів та доступу до e-learning ресурсів через них.
2. Організовано доступ до різних функцій системи для різних ролей користувачів: студент та викладач.
3. Розроблена система надає можливість викладачу:

- Створити курс;
- Додати урок;
- Додати матеріали до урок, такі як текст, відео, аудіо тощо;
- Надати домашнє завдання до уроку;
- Перевірити домашнє завдання після його здачі студентом.

4. Система написана з дотриманням правил «чистого» коду та покрита модульними та інтеграційними тестами. Це дозволяє їй бути легко розширюваною, тобто такою, що додавання нового функціоналу та залучення до розробки нових людей не спричинить великих проблем.
5. У якості методу розробки програмного забезпечення був використаний Agile підхід, а саме FDD(Feature Driven Development). Цей підхід дозволив з легкістю регулярно додавати новий функціонал до системи.
6. Вся програма реалізована на клієнт-серверній архітектурі, тобто були побудовані підпрограми для серверної та клієнтської частин. Серверна частина написана на мові програмування Java з використанням сімейства фреймворків Spring. В свою чергу, клієнтська частина була реалізована одразу двома різними застосунками: веб та мобільний. Веб застосунок був реалізований за допомогою мови програмування JavaScript та фреймворку React.js. Мобільний застосунок розроблявся під операційну систему Android на мові програмування Kotlin.

7. У якості сховища даних була використана реляційна база даних MySQL.
8. В ході роботи засвоєні абсолютно нові технології, такі як JavaScript та React.js. Також поглиблено навички з об’єктно-орієнтовного програмування, побудови UML діаграм, розробки моделі бази даних. Отримано неоціненний досвід розробки програмного продукту від етапу створення ідея і до самої її реалізації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бейнон-Девіс П. Системи бази даних / Пол Бейнон-Девіс. – Palgrave Macmillan – 2003(3).
2. Кон М. Успіх з Agile. Розробка програмного забезпечення за допомогою Scrum / Кон Майк – 2009.
3. Крейг Л. Agile та ітеративна розробка ПЗ: Посібник менеджера / Ларман Крейг. – Addison-Wesley – 2004
4. Палмер С. Р., Фелсинг Д. М. Практичний посібник з Feature-Driven Development / – Prentice Hall – 2002.
5. Буч Г. Посібник з уніфікованої мови моделювання / Буч Греді. – Addison-Wesley – 2005.
6. Амман П., Оффут Д. Вступ до тестування програмного забезпечення / – Кембриджська університетська преса – 2016
7. Еккель Б. Філософія Java / Брюс Еккель. – 2015 (4)
8. Рогачев С. Узагальнений Model-View-Controller / Сергей Рогачев. – 2007
9. Сибельшатц А., Сударшан С. Концепції системи баз даних / – New York: McGraw-Hill – 2011

ДОДАТОК 1

Web-портал для організації доступу до e-learning ресурсів (Організація доступу для вчителя)

Специфікація

Аркушів 2

Київ – 2021

Позначення	Найменування	Примітки
Документація		
	Бортнічук_НО_ТІ-72_Записка.docx	Пояснювальна записка
Компоненти		
	UserService.js	Модуль сервісів клієнтської частини
	Courses.js	Модуль відображення сервісної частини

ДОДАТОК 2

Web-портал для організації доступу до e-learning ресурсів (Організація доступу для вчителя)

Текст програми

Аркушів 9

Київ – 2021

```
import apiUrl from '../config/config';
import { authHeader } from '../auth/helpers';

export const userService = {
    login,
    register,
    update,
    logout,
    getByUsername
};

function login(username, password) {
    const requestOptions = {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ username, password })
    }

    return fetch(`${apiUrl}/auth/login`, requestOptions)
        .then(handleResponse)
        .then(user => {
            localStorage.setItem('user', JSON.stringify(user))
        });
}

function register(user) {
    const requestOptions = {
```

```
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify(user)
    };

    return fetch(`${apiUrl}/auth/register`, requestOptions)
        .then(handleResponse)
        .then(user => {
            localStorage.setItem('user', JSON.stringify(user))
            console.log(user)
        })
}

function update(user) {
    const requestOptions = {
        method: 'PUT',
        headers: { ...authHeader(), 'Content-Type': 'application/json' },
        body: JSON.stringify(user)
    };

    return fetch(`${apiUrl}/users/${user.id}`, requestOptions).then(handleResponse);
}

function logout() {
    localStorage.removeItem('user');
}
```

```
function getByUsername(username) {
    const requestOptions = {
        method: 'GET',
        mode: 'no-cors',
        headers: { ...authHeader(), 'Content-Type': 'application/json'}
    };
    console.log(username);
    return fetch(`${apiUrl}/users/user-data`, requestOptions).then(handleResponse);
}

function handleResponse(response) {
    return response.text().then(text => {
        console.log(text)
        const data = text && JSON.parse(text);
        if (!response.ok) {
            if (response.status === 401) {
                logout();
            }
            const error = (data && data.message) || response.statusText;
            return Promise.reject(error);
        }
        return data;
    }
    )
}
import { Component } from 'react'
import { Card, Row } from 'react-bootstrap';
```

```
import { connect } from 'react-redux';
import { courseActions } from '../model/actions/course/CourseActions';
import { withRouter } from "react-router-dom";
import './Courses.css'

class CoursesPage extends Component {

   componentDidMount() {
      this.props.getAll()
   }

   render() {
      const { courses } = this.props
      return (
         <Row>
            <div className="courses container">
               <h1 className="courses-title">Курси</h1>
               <div className="divider"></div>
               {courses.items &&
                  <ul className="courses-list row">
                     {courses.items.map((course, index) =>
                        <li className="col-sm-3" key={index}>
                           <a href={`course/${course.id}/promo`}>
                              <Card className="shadow p-3 mb-5 bg-white rounded">
                                 <Card.Img          className="course-image"          variant="top"
src={course.photoPreview} />
                                 <Card.Body className="course-card-body">
```

```
                                    <Card.Title                                    className="course-
name">{course.name}</Card.Title>
                                    <Card.Text                                     className="course-
description">{course.description}</Card.Text>
                                        </Card.Body>
                                    </Card>
                                </a>
                            </li>
                        )}
                    </ul>
                }
            </div>
        </Row>
    );
  }
}
function mapState(state) {
  const { courses } = state;
  return { courses };
}
const actionCreators = {
  getAll: courseActions.getAll
}
const Courses = connect(mapState, actionCreators)(CoursesPage);
export default withRouter(Courses)
import './Auth.css'
```

```
import {
   BrowserRouter as Router,
} from "react-router-dom";
import { Component } from 'react';
import Login from './login/Login';
import Register from './register/Register';

const Authlinks = () => {
   return (
      <ul className="nav-links">
         <li      className="nav-item"><a      className="link      navbar-link      active"
href="?auth=login">Увійти</a></li>
         <li      className="nav-item"><a      className="link      navbar-link      active"
href="?auth=register">Реєстрація</a></li>
      </ul>
   )
}

class Auth extends Component {
   constructor(props) {
      super();
      this.state = {
         isVisible: true,
         authQuery: props.auth
      };
      this.onClose = this.onClose.bind(this);
   }
```

```
onClose() {
   this.setState({ isVisible: !this.state.isVisible });
}

render() {
   const isVisible = this.state.isVisible
   const authQuery = this.state.authQuery

   const Authblock = () => {
      return (
         <Router>
            <div className="auth-out" onClick={this.onClose}></div>
            <div className="auth-in">
               <header className="auth-header">
                  <Authlinks />
                  <div className="vertical-divider"></div>
               </header>
               {authQuery === 'login' && <Login />}
               {authQuery === 'register' && <Register />}
               <div className="auth-close" onClick={this.onClose}>
                  <svg id="modal-button-close" viewBox="0 0 16 16">
                     <g fill="none" stroke="currentColor" stroke-width="2">
                        <path d="M2 14L14 2M2 2l12 12"></path>
                     </g>
                  </svg>
               </div>
```

```
                </div>
            </Router>
        )
    }
    return (
        <div>
            { isVisible && <Authblock />}
        </div>
    )
  }
}

export default Auth;
```

ДОДАТОК 3

Web-портал для організації доступу до e-learning ресурсів (Організація доступу для вчителя)

Опис програмного модуля

Аркушів 8

Київ – 2021

АНОТАЦІЯ

Розроблений застосунок реалізує веб портал навчальних курсів, де викладач створює курси, а студенти навчаються за ними. Система дозволяє організувати зручний навчальний процес онлайн.

Розроблено серверний застосунок з використанням фреймворку Spring та мови програмування Java.

Розроблено два клієнтські застосунки: веб та мобільний. Тобто користувач отримує одну і ту ж інформацію на різних пристроях. Для побудови веб застосунку використовувався фреймворк React.js, а для мобільного застосунок розроблений під систему Android та з використаннях мови програмування Kotlin.

ЗМІСТ

ЗАГАЛЬНІ ВІДОМОСТІ

Програма є сервісом для створення і проходження навчальних онлайн курсів.

Програма завантажена на хостинг, отже вона не потребує завантаження та встановлення будь-яких компонентів. Але це означає, що для коректної роботи вона потребує доступу в інтернет.

Серверна частина застосунку написана на мові програмування Java. Веб застосунок написаний на мові програмування JavaScript, а мобільний застосунок розроблений за допомогою мови Kotlin.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Програмний продукт надає викладачу наступні можливості:

- Створення курсу;
- Додавання уроків;
- Додавання матеріалів до урок, такі як текст, відео, аудіо тощо;
- Надання домашнього завдання до уроку;
- Перевірка домашнього завдання після його здачі студентом.

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Загальний алгоритм системи може бути представлений у вигляді послідовного списку подій, що відбуваються:

1) Викладач вводить дані;
2) Відбувається перевірка правильності вводу на стороні клієнта;
    1. Якщо дані некоректні – виводиться помилка;
    2. Якщо дані коректні – відправляється запит з даними на сервер;
3) Сервер оновлює дані в базі;
    1. Дані оновилися успішно – на клієнт відправляється повідомлення про успішне оновлення даних;
    2. Виникла помилка – на клієнт відправляється відповідна помилка;
4) На клієнті відображається відповідне повідомлення

ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ

Програмна система розроблялася на ноутбуці HP Elitebook.

- Процесор: Intel core i7 10^{th} gen
- Пам’ять комп’ютера: SSD 512 gb
- Оперативна пам’ять: 32 gb

Також для тестування мобільного застосунку використовувався телефон OnePlus 5T.

ВХІДНІ ТА ВИХІДНІ ДАНІ

У програмній системі, що була розроблена, вхідні та вихідні дані обробляються в текстовому форматі. В деяких випадках вхідними даними є посилання на відео, яке згодом відображається як вбудований відео програвач.