Галицький фаховий коледж імені В’ячеслава Чорновола

відділення комп’ютерних та видавничих технологій

циклова комісія інформатики та комп’ютерних дисциплін

ДОПУСТИТИ ДО ЗАХИСТУ

Завідувач відділенням комп’ютерних
та видавничихтехнологій
Чубей О.О. /_____/

підпис

«___» _____2022 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проекту

освітньо-кваліфікаційного рівня «молодший спеціаліст»

зі спеціальності 122 «Комп’ютерні науки»

на тему:

“Інформаційна система підтримки освітнього процесу студентів Галицького коледжу імені В’ячеслава Чорновола”

Студент групи КН-41	Захарків В.В.	_____ (підпис)
Студент групи КН-41	Захарків М.В.	_____ (підпис)
Керівник проєкту	Павлюс В.П.	_____ (підпис)
Консультанти:		
З техніко-економічного обґрунтування	Меленчук Л.І	_____ (підпис)
нормоконтролер	Сиротюк Н.С.	_____ (підпис)

Тернопіль – 2022

Галицький фаховий коледж імені В’ячеслава Чорновола

відділення комп’ютерних та видавничих технологій

циклова комісія інформатики та комп’ютерних дисциплін

ЗАТВЕРДЖУЮ

Завідувач відділенням

комп’ютерних та видавничих

технологій

Чубей О.О. /_____/

підпис

«__»_____2021 р.

ЗАВДАННЯ

на дипломне проєктування

на здобуття освітньо-кваліфікаційного рівня «молодший спеціаліст»

студентам Захарківу Віталію Володимировичу та

Захарківу Максиму Володимировичу

(прізвище, ім’я та по-батькові студента)

1. Тема проєкту «Інформаційна система підтримки освітнього процесу студентів Галицького коледжу імені В’ячеслава Чорновола» затверджено наказом по коледжу від “_____”____202_ р., №_
2. Термін здачі студентом завершеного проєкту “___”_____202_ р.
3. Вихідні дані до проєкту_____

_____

_____

4. Перелік питань, які повинні бути розроблені в проєкті:

а) основна частина_____

_____

б) техніко-економічне обґрунтування_____

_____

_____

_____

5. Перелік графічного матеріалу_____

_____

6. Консультанти проєкту: _____

Розділ	Консультанти	Підпис, дата	
		Завдання видано	Завдання прийнято
з техніко-економічного обґрунтування	Меленчук Л.І. (вчена ступінь, звання, П.І.Б. консультанта)		

КАЛЕНДАРНИЙ ПЛАН
дипломного проєктування

№ п/п	Найменування етапу	Терміни	
		початку	завершення
1.	Вибір теми, ознайомлення з вимогами до дипломного проєктування	20.09.21	01.10.21
2.	Огляд типових рішень та написання відповідного розділу ПЗ	02.12.21	28.01.22
3.	Дослідження технологій реалізації та написання відповідного розділу ПЗ	28.01.22	17.02.22
4.	Розробка функціональних вимог до проєкту та робота над структурою програмного продукту. Написання відповідного розділу ПЗ	17.02.22	04.03.22
5.	Встановлення та налаштування середовища реалізації та написання відповідного розділу ПЗ	04.03.22	16.03.22
6.	Проєктування програмного засобу (функціоналу, інтерфейсу, бази даних продукту) та написання відповідного розділу ПЗ	21.03.22	15.04.22
7.	Реалізація та налаштування програмного засобу та написання відповідного розділу ПЗ	18.04.22	04.05.22
8.	Доопрацювання модулів	03.05.22	17.05.22
9.	Опрацювання економічного розділу дипломного проєкту та оформлення спеціального розділу	05.05.22	18.05.22
10.	Тестування та налагодження програмного продукту та написання відповідного розділу ПЗ	18.05.22	04.06.22
11.	Робота над оформленням пояснювальної записки	05.06.22	12.06.22
12.	Попередній захист дипломного проєкту, доопрацювання	15.06.22	15.06.22
13.	Підготовка до захисту дипломного проєкту	16.06.22	24.06.22
14.	Захист дипломного проєкту	25.06.22	25.06.22

7. Дата видачі "___"_____2022р. Керівник_____/

Завдання прийняв до виконання

Реферат

Інформаційна система підтримки освітнього процесу студентів Галицького коледжу імені В’ячеслава Чорновола. Дипломний проєкт. Захарків Віталій, Захарків Максим. Галицький фаховий коледж імені В’ячеслава Чорновола, відділення комп’ютерних та видавничих технологій. Спеціальність 122 «Комп’ютерних технологій». ГК, 2022. Сторінок – 85, рисунків – 57, додатків – 3.

Об’єкт дослідження – система підтримки освітнього процесу.

Метою проєкту є пошук та аналіз існуючих у відкритому доступі систем, визначення їх переваг та недолік.

Завдання проєкту є розробка система підтримки освітнього процесу студентів Галицького фахового коледжу імені В’ячеслава Чорновола.

Для реалізації даної системи було використано велику кількість різних інструментів, бібліотек, шаблонів, доступних у мовах програмування Python та JavaScript. Середовищем для реалізації системи проєкту було обранно PyCharm та WebStorm.

СИСТЕМА ПІДТРИМКИ ОСВІТНЬОГО ПРОЦЕСУ, PYTHON, JAVASCRIPT, DJANGO, DJANGO REST FRAMEWORK, VUEJS, VUETIFY, PYCHARM, WEBSTORM.

Abstract

Information system to support the educational process of students of the Vyacheslav Chornovil Galytskyy College. Diploma project. Zakharkiv Vitaliy, Zakharkiv Maxym. Vyacheslav Chornovil Galytskyy professional College, Department of Computer and Publishing Technologies. Specialty 122 "Computer Technology". GK, 2022. Pages – 82, figures – 57, applications – 3.

The object of research is the system of support of the educational process.

The aim of the project is to search and analyze existing publicly available systems, identify their advantages and disadvantages.

The task of the project is to develop a system to support the educational process of students of the Vyacheslav Chornovil Galytskyy professional College.

A large number of different tools, libraries, templates available in Python and JavaScript programming languages were used to implement this system. PyCharm and WebStorm were chosen as the environment for the project system implementation.

EDUCATIONAL PROCESS SUPPORT SYSTEM, PYTHON, JAVASCRIPT, DJANGO, DJANGO REST FRAMEWORK, VUEJS, VUETIFY, PYCHARM, WEBSTORM.

ЗМІСТ

СКОРОЧЕННЯ І УМОВНІ ПОЗНАКИ

JS – JavaScript;

БД – база даних;

HTML – HyperText Markup Language;

UML – Unified Modeling Language;

CSS - Cascading Style Sheets;

SQL - Structured Query Language;

API - Application Programming Interface;

СУБД – система управління бази даних;

GIL – Global Interpreter Lock;

HTTP – HyperText Transfer Protocol;

TCP – Transmission Control Protocol;

ПЗ – програмне забезпечення;

ОС – операційна система;

FTP – File Transfer Protocol;

SPA – Single Page Application;

DOM – Document Object Model;

JWT – JSON Web Token;

CRUD – Create, Read, Update, Delete;

DRF – Django Rest Framework;

REST - Representational State Transfer.

ВСТУП

У теперішніх умовах робоче місце, облаштоване автоматизованими пристроями, уже є буденною реальністю. Сучасні автоматизовані засоби дали свіжі можливості для праці та перепочинку, дали можливість спростити діяльність людини, а обрії вдосконалення обрахункових пристроїв тепер важко уявити.

Спільнота сповнена та пронизана хвилями даних, що вимагають вчасної та якісної обробки. Звідси без сучасних технологій, як і без хімічних, транспортних, енергетичних розробок, вона не зможе коректно працювати. Тому, нові технології, які визначають явища в усіх галузях людської практики, вимагають працівників із високими ступенем інформаційної обізнаності.

Інформаційно-комунікаційні розробки доторкаються до усіх галузей діяльності людини, звідси вони містять немалий вплив на освітній процес, тому що відчиняються перспективи внесення цілком нових методів подання інформації. Використання комп'ютерів в навчанні привело до появлення нових генерацій інформаційних технологій, що допомагають підняти якість освіти, утворити свіжі прийоми впливу, краще діяти викладачам із учнями. Застосування сучасних інформаційних технологій в освіті – це не лише технологічні можливості але і нові методи та форми подання інформації, нові підходи до процесу освіти. Це схиляє викладачів до застосування нових методів навчання та використання цих технологій у освітній хід.

На розвиток і формування людини впливає оточення, у якому вона існує. Звідси актуальність проблеми для будь-якого навчального закладу є потреба використання нинішніх інноваційних технологій для утворення інформативного освітньо-наукового середовища з контактним потенціалом.

У багатьох навчальних закладах в наявності є інформаційні онлайл-системи, з огляду своїх вимог і можливостей. Тому, для нашого навчального закладу буде розроблено інформаційну систему підтримки освітнього процесу студентів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАВДАННЯ ПРОЄКТУВАННЯ

1.1 Обґрунтування доцільності створення системи

Записна книжка являється головним освітнім документом, у якій зберігаються підсумкові атестації студентів за навчальною програмою та підбиває підсумки результату за всі етапи навчання.

Звичайна записна книжка студента понесла відчутні зміни з стандартного паперового вигляду вона змінилась на електронний вигляд. Це сталося у зв'язку із не комфортною роботою із традиційним варіантом. З електронною варіацією сам методист буде мати в своєму розпорядженні всіх, без виключення студентів, а тому в свою чергу не потребуватиме з’являтись у закладі задля того, аби дізнатись свої бали.

Студентська записна книжка допомагають викладачам на власні очі прослідкувати дані про успіхи студентів у навчальному процесі. Це спеціальний знаряддя, яке допомагає здійснювати будь-які дії під час організації освітнього процесу. Онлайн записна книжка допомагає студентам слідкувати за своїми досягненнями у навчанні.

Теперішні студенти мають можливість звіряти свій статус навчання онлайн. Онлайн залікова книжка дає змогу в онлайн режимі онлайн бачити дані про свою результативність, оцінку за будь-який іспит, предмет, залік. Окрім цього викладачі мають змогу слідкувати, як розвивається студент під час навчання, подивитись на індивідуальний прогрес кожної людини.

Даний проєкт вирішує проблему заміни паперової залікової книжки на її електронний аналог.

Негативні сторони паперової залікової книжки:

− Паперові записні книжки є важкими. Зараз викладачі, маючи вільний час, фізично не зможуть постійно носити з собою велику кількість залікових книжок.

− Паперові записні книжки займають доволі багато місця, а місце їх зберігання є обмеженим.

− Виготовлення паперової записної книжки є не екологічним варіантом, тому що для її виготовлення потрібна деревина.

− Паперова записна книжка є платною.

− Якщо її загубити, знайти або відновити її буде доволі складно.

Позитивні сторони електронної книжки:

− Для її зберігання не потрібно мати велике приміщення, а дані можна зберігати на носіях невеликих розмірів.

− Онлайн записна книжка, на відміну від паперового аналога, є більш екологічним варіантом.

− При перегляді немає необхідності в підсвічуванні книжки, тому що онлайн книжка має своє підсвічування.

− Може бути створена можливість додавання до онлайн книжки медіа-контенту.

− Наявність змінювати розмір і шрифт букв, для власних зручностей.

− При втраті книжки її можна швидко знайти або відновити.

− Дає можливість використовувати кращі варіанти для закладок або позначок, на відміну від паперової книжки.

1.2 Огляди існуючих рішень

Широке поширення в мережі Інтернет отримали інформаційні системи, які створенні для потреб навчальних закладів. Дані проєкти дають змогу легко вести дані про успішність студентів. Інформація стає більш доступною як для студента так і для викладача.

Обстеження та розбір наявних онлайн систем показав, що є немало різних за оформлення та функціональними можливостями інформаційних систем. Для формулювання вимог до функціональними можливостями інформаційних систем.

Для формулювання вимог до функціональних можливостей до розробки даної теми розглянемо кілька діючих систем, які існують у мережі інтернет. Першим прикладом подібної системи може слугувати розробка Національного університету харчових технологій у Києві (рисунок 1.1). В ній реалізований простий, але в той же час зручний для користувача інтерфейс та функції які вона повинна виконувати.

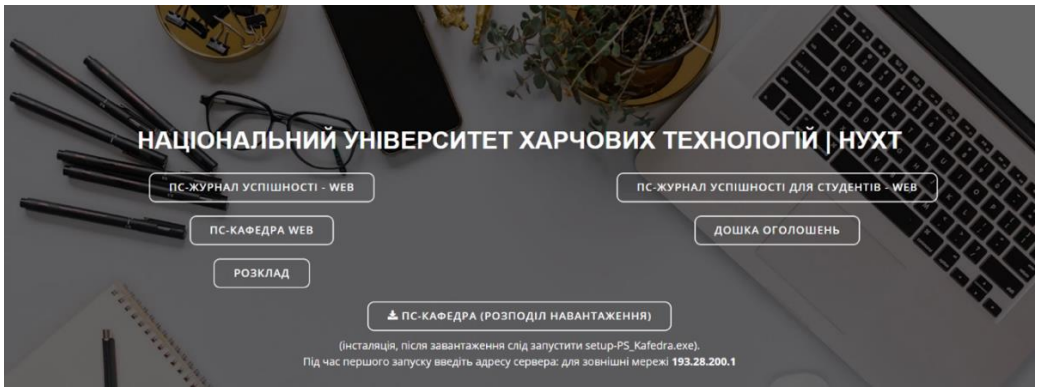

Рисунок 1.1 – Головна сторінка електронної залікової книжки НУХТ

Цей інформаційний ресурс дає змогу здійснювати:

- Вхід у систему.
- Дії над предметами.
- Дії над студентами.
- Перевірка відвідуваності.
- Можливість вивантаження даних.
- Введення успішності.

Наступна система є у вільному доступі, якою можна користуватись пройшовши легку систему реєстрації (рисунок 1.2).

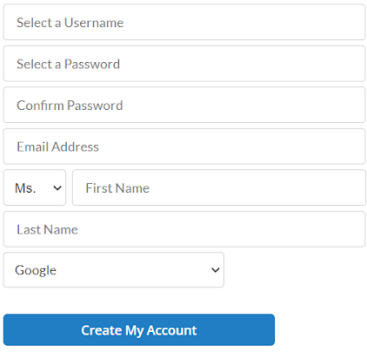

Рисунок 1.2 – Сторінка реєстрації

Далі попадаємо на головну сторінку (рисунок 1.3).

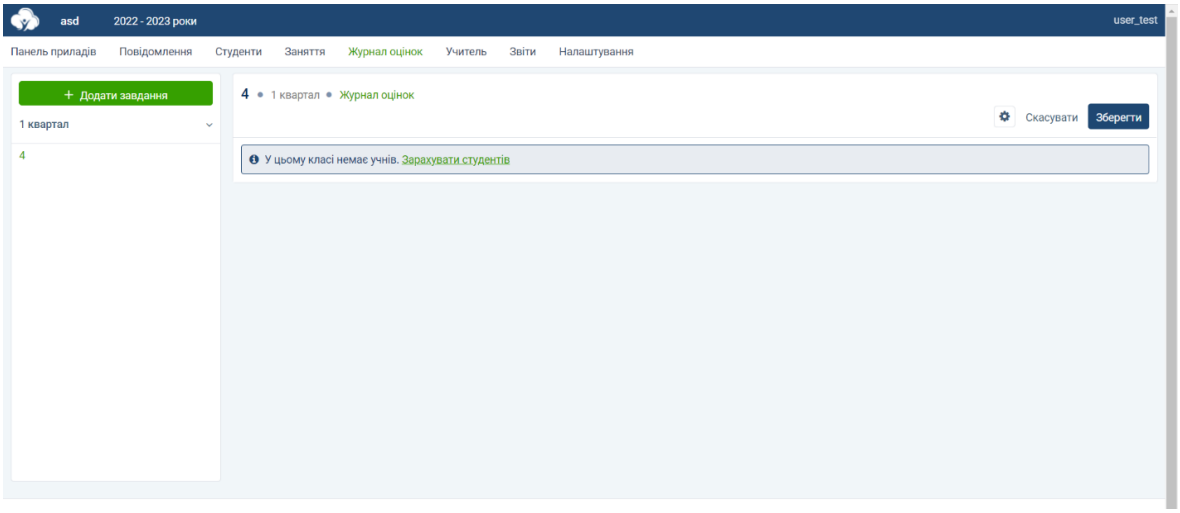

Рисунок 1.3 – Головна сторінка

Цей інформаційний ресурс дає змогу здійснювати:

- Вхід у систему.
- Дії над предметами.
- Дії над студентами.
- Постановка завдань.
- Введення діалогів в реальному часі.
- Введення успішності.

Останній ресурс у вільному доступі, якою можна користуватись пройшовши легку систему реєстрації (рисунок 1.4).

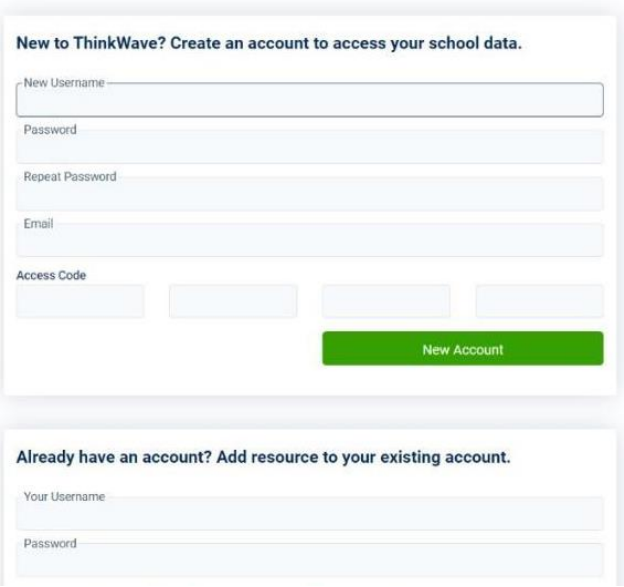

Рисунок 1.4 – Сторінка реєстрації

Далі попадаємо на головну сторінку (рисунок 1.5).

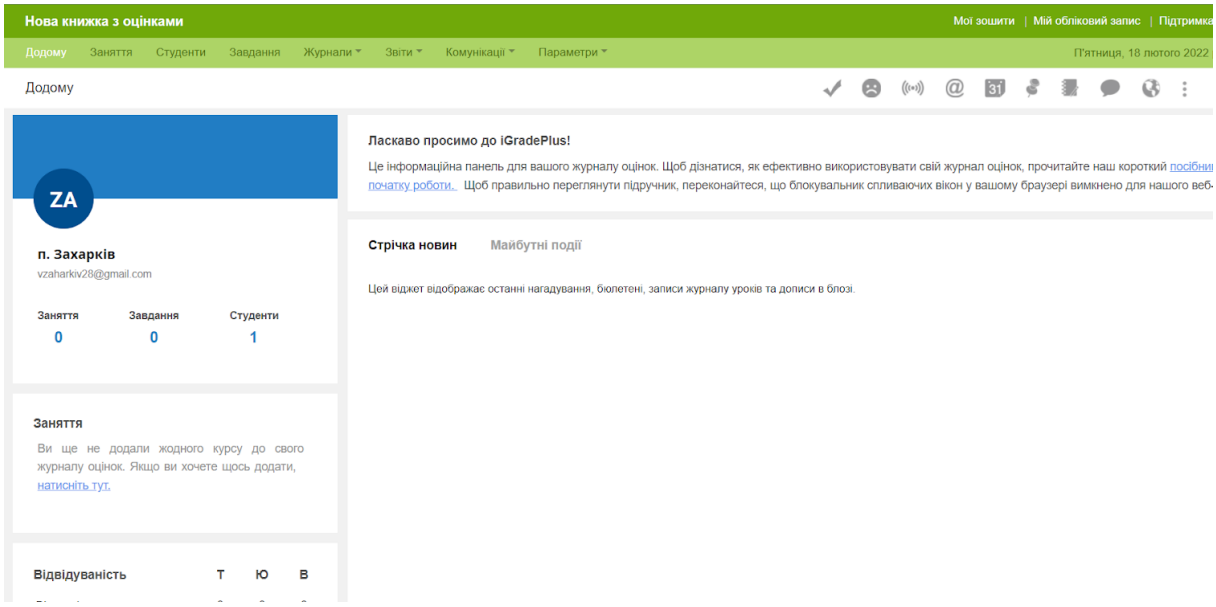

Рисунок 1.5 – Головна сторінка

Цей інформаційний ресурс дає змогу здійснювати:

- Вхід у систему.
- Дії над предметами.
- Дії над студентами.
- Постановка завдань.
- Контроль присутності.
- Введення діалогів в реальному часі.
- Введення успішності.

Опісля наведення прикладу наявних онлайн систем у відкритому досту пі, проведено аналіз за декількома обраними характеристиками. Зіставленні я кості переглянутих онлйн систем показано в таблиця 1.1.

У колонках таблиці розміщенні назви наявних онлайн систем, а в рядках – характеристики. Позначкою «+» продемонстровано присутність переглянутої характеристики, а позначкою «-» – його недостачу даного критерію або відсутність інформації про нього.

Таблиця 1.1 – Аналіз існуючих систем

Назви / Критерії	Система НУХТ	thinkwave	igradeplus
Авторизація в системі	+	+	+
Розмежування прав доступу до інформації в системі	-	+	+
Динамічне формування сторінок	+	+	+
Дизайн та подання інформації, Інформаційне наповнення.	+/-	+/-	+
Введення успішності	+	+	+
Постановка завдань	-	+	+
Дії над предметами (додавання, видалення, редагування, пошук)	+	+	+
Дії над студентами (додавання, видалення, редагування, пошук)	+	+	+
Введення діалогів в реальному часі	-	+	+
Розробник	-	-	-

Розгляд інформації таблиця 1.1 дає можливість говорити, що немає з наявних систем, які можуть забезпечити користувачів всіма функціональними можливостями.

У розглянутих онлайн системах є наступні негативні сторони:

- Обмеженні в функціоналі.
- Нечітке розподілення ролей.
- Відсутність рейтингового списку.
- Відсутність можливості виставлення додаткового балу.

1.3 Постановка завдання

У зв’язку з цим, необхідно визначити вимоги до розробки інформаційно-довідкової Інтернет-системи навчального закладу, які допоможуть ліквідувати зазначені недоліки.

Для здійснення подальшого проектування інформаційно-довідкової Інтернет-системи навчального закладу необхідно визначити категорії користувачів системи.

Автоматизована система зобов’язана забезпечити авторизацію для різних категорій користувачів:

- Адміністратор.
- Студент.
- Методист.
- Викладач.
- Адміністрація закладу.

Данний ресурс забезпечує наступні функціональні можливості:

- Зручний інтерфейс системи.
- Зручні засоби навігації.
- Добавлення та розподілення користувачів в системі(адміністратор).
- Дії з предметами та студентами(методист).
- Перегляд успішності(студент).

Варто надати можливість одержання доступу до веб-застосунку з абиякого пристроя клієнта із інстальованим браузером без потреби установлення додаткового ПЗ.

Ціллю зазначеного дипломного проєкту – розробка інформаційно-довідкової додатка, котра повинна застосовувати будь-який навчальний заклад для керування свого інтернет-ресурсу.

Схему організаційної структури (рисунок 1.6).

Рисунок 1.6 – Схема організаційної структури

Найвищим в ієрархії користувачів є адміністратор системи. Від нього залежить всі процеси, які виконують інші групи користувачів: створення, редагування та видалення користувачів. Залежними від адміністратора є наступні не пов’язані між собою категорії користувачів:

- Методист.
- Викладач.
- Адміністрація.

Найнижчою ланкою у системі є категорія студентів.

Навчальна система здійснює грядущий функціонал:

- Авторизація у систему.

- Дані про рейтингові списки студентів.
- Онлайн записна книжка.
- Записи даних.
- Збереження даних.
- Редагування даних.
- Знищення даних.
- Налагодження облікових записів користувачів.
- Занесення даних в онлайн журнал.

Зазначена інформаційна система має реалізувати щоденну підтримку навчального процесу.

На основі перегляду і розбору матеріальної сфери та правил онлайн-система має забезпечувати додатки користувачів: методиста, студента, адміністратора, адміністрації закладу (рисунок 1.7).

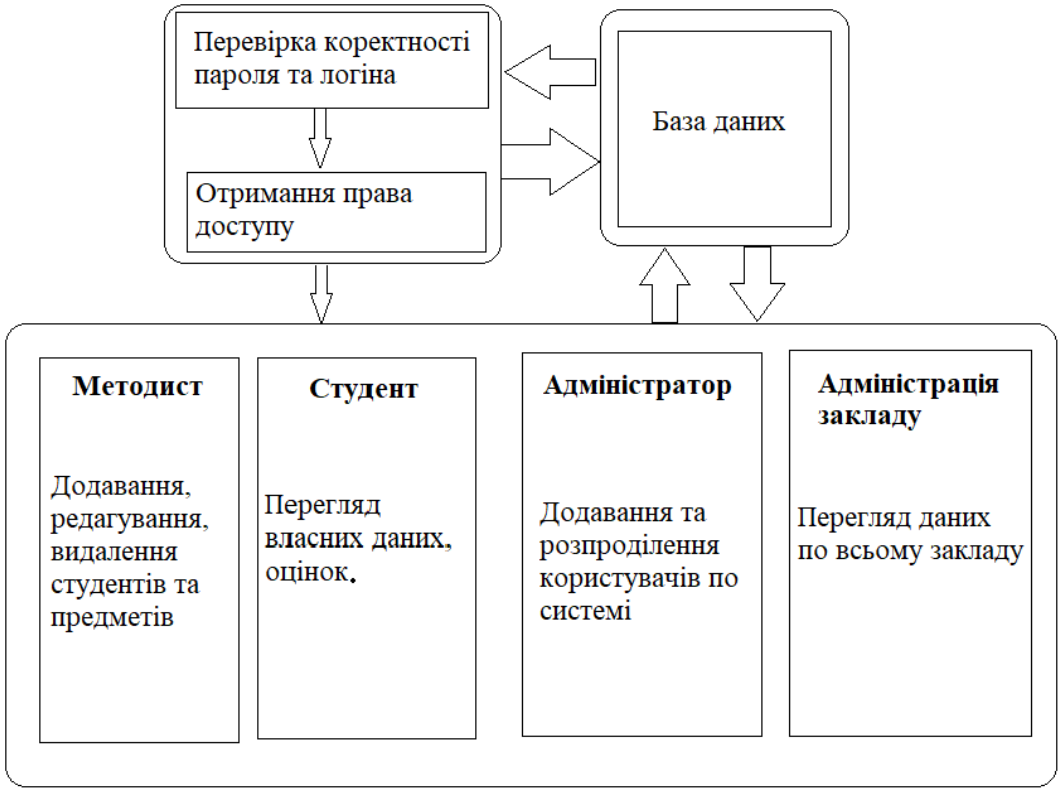

Рисунок 1.7 – Схема управління процесами

Онлайн-додаток методиста має надати наступні функціональні можливості: додавання, редагування, сортування, фільтрація та видалення

даних про студентів. Також подібний функціонал реалізовано для предметів. Окрім цього надати можливість переглядати інформацію про кожного студента, його оцінки. Також створення, видалення та редагування додаткових балів. Формування рейтингових списків.

Онлайн-додаток студента має надати наступні функціональні можливості: перегляд дослідження індивідуального прогресу навчання.

Онлайн-додаток адміністратора має надати наступні функціональні можливості: додавання та розподілення користувачів у системі, перегляд всієї інформації про сутності бази даних.

Онлайн-додаток адміністрації закладу має надати наступні функціональні можливості: перегляд всіх даних по навчальному процесу.

Отже, у першому розділі проаналізовано предметну область, визначенні основні задачі проектування системи. Також було проведено обгрунтування теми дипломного проєкту та аналіз уже існуючих систем, виявлені їх функціональні можливості, а також істотні недоліки, які слугують основною відмінністю теми дипломного проекту від її аналогів. Розподіл прав і функціональних можливостей між різними групами користувачів. Визначення головних завдань та ролей всій категорій користувачів у веб-застосунку.

2 ПРОЄКТУВАННЯ СИСТЕМИ

Піднесення мережі Інтернет привело до появлення нових програмних засобів – веб-застосунків. Вони включають в себе комплект веб-сторінок, планів та різних інших засобів, які містяться на одному або більше комп'ютерах і з'єднанні для здійснення практичних завдань. Теперішні інформаційні системи, збудовані на суті веб-застосунків, що містять в собі бази даних, грунтуються на багатошаровій клієнт-серверній архітектурі та на правилах роботи Інтернету. Веб-застосунок здійснюються на стороні веб-сервера, який містить на веб-вузлах Інтернету. Веб-сервер виконує обробку запитів браузера на одержання веб-сторінок і надсилає йому потрібну інформацію у вигляді веб-документів. Обмін інформацією в Інтернеті виконується на базі контактуючого протоколу TCP / IP і протоколу більш вищого ступеня HTTP[1].

2.1 Проектування системи

Інформаційна система підтримки освітнього процесу студентів – це сервіс, котрий призначений загалом для методистів та студентів, аби полегшити введення успішності. Тому, реалізація має бути ретельно продумана. Це стосується як сервісної так і клієнтської частини.

Перед початком реалізації серверної частини, слід обрати тип архітектури, визначити слої і патерни, які буде використовувати система.

За типом архітектури backend можна поділити на:

- Монолітна.
- Мікросервісна.
- Модульна архітектура.

Кожна із них має свої плюси і мінуси, але зазвичай веб-застосунки розпочинають як моноліт, проте коли проєкт зростає хорошою практикою рахується поділити його на кільки сервісів, котрі працюють незалежно.

Програмне забезпечення на основі монолітної архітектури (рисунок 2.1) будується як одне ціле, де візуальна частина та реалізація доступу до даних зведені в одну програму.

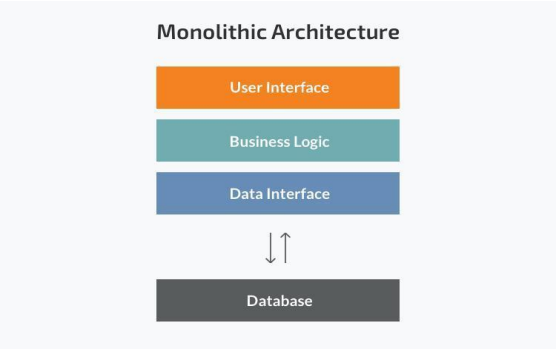

Рисунок 2.1 – Принцип роботи монолітної архітектури

Переваги:

– Простота в розробці. На перших порах, проєкту краще перейти на монолітну архітектуру.

– Простий у розгортанні. Потрібно скопіювати упаковану програму на сервер.

– Проста інтеграція систем моніторингу.

Недоліки:

– Розмір програми сповільнить час запуску.

– Нероздільні програми є складними для масштабування, коли різноманітні блоки мають розбіжні вимоги до джерел.

Після того як було обрана архітектура, слід визначити слої на котрих буде будуватися система. За замовчуванням в Django виділені такі слої, як моделі, контролери, шаблони[2]. Моделі відповідають за взаємодію із базою даних, шаблони для виводу контенту, а контролери зв’язують між собою дані і відповідні їм сторінки.

І тут можна зробити висновок, що контролер є головним слоєм і всю логіку потрібно реалізувати безпосередньо у ньому. Про це рахується поганою практикою, оскільки його задача це виклик відповідних для нього функцій та виводу результату. Після цього виникає питання де у системі реалізовувати

логіку і на рахунок цього існує багато думок. Одна із, помістити логіку у моделі та зробити їх товстими. Цей варіант кращий, аніж зберігання у контролері, проте існують багато мінусів такі як, порушення принципу відповідальності та використання логіки у різних куточках програми. І тут є очевидним добавити свій новий слой для логіки.

Тепер потрібно розібратись, як поділяється логіка.

Поділ логіки:

- Бізнес логіка – код реалізації бізнес-правил.
- Код інфраструктури – робота із БД.

Для кожного типу потрібно створити свій слой, і тоді система буде мати такий вигляд: моделі, репозиторії, сервіси, контролери.

Аналогічно з backend частиною, frontend частина побудована за подібним типом архітектури. За замовчування у VueJs реалізована зрозумілі та лаконічні слої відповідальності: мережевий – створення запитів, глобальне сховище – робота з даними, компонент – вивід даних.

2.2 Логічна та фізична моделі бази даних

Проектування БД — подання дизайну даних як втіленого чи призначеного для виконання у системі управління базами даних. БД системи містить наступні сутності: Користувачі, Студенти, Предмети, Відділення, Права, Групи, Освітні програми, Оцінки, Додаткові бали.

Виходячи з аналізу предметної області, для функціонування інформаційної система підтримки освітнього процесу студентів слід спроектувати базу даних, у котрій мають бути наступні дані:

- Про користувачів.
- Про групи.
- Про предмети.
- Про оцінки.
- Про додаткові бали.

– Про відділи.

– Про освітні програми.

– Про файли студента.

– Про права доступу.

Кожному об'єкту варто надати первинні ключі – виняткові ідентифікатори, які недвозначно визначають кожен екземпляр об'єкта. Для відмітки первинних ключів у добавок потрібно ввести поля ідентифікатори.

Основні вимоги до інформаційної системи:

– Всі користувачі належать до певного відділення за допомогою зв’язку одни-до-багатьох із таблицею Відділення(як вписати назву таблиці).

– Всі користувачі входять до певної групи за допомогою зв’язку один-до-багатьох із таблицею Права доступу.

– Предмети повинні бути унікальні для кожної групи.

– Початковий семестр не має бути більшим за кінцевим.

– Оцінка не має повторюватись по назві предмета, студенту та семестру.

Схема БД (рисунок 2.2).

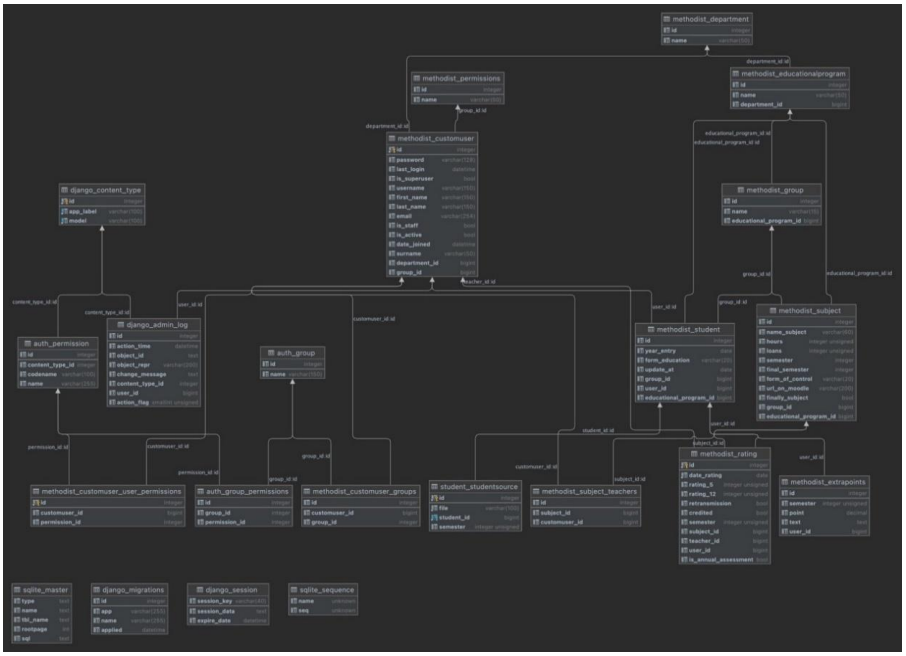

Рисунок 2.2 – Схема бази даних

Виходячи з вимог до системи, був отриманий ряд сутностей реляційної БД, які матимуть наступний структурний вигляд (табл. 2.1).

Таблиця 2.1 - Предмети

Назва стовпця	Тип даних	Null	Unique
Назва	varchar(255)	False	True
Група	int	False	True
Викладачі	int	False	False
Години	int	False	False
Кредити	int	True	False
Семестр	int	False	False
Освіт. програм.	int	True	False
Кінцевий семестр	int	True	False
Форма контролю	varchar(255)	False	False
Фінальний предмет	boolean	False	False
Ідентифікатор	int	False	True
Оцінки			
Студент	Int	False	True
Предмет	int	False	True
Дата	date	False	False
5-бальний бал	int	True	False
12-бальний бал	int	True	False
Перездача	boolean	True	False
Зараховано	boolean	True	False
Семестр	int	False	True

Продовження таблиці 2.1

Річна	boolean	True	True
Викладач	int	False	False
Ідентифікатор	int	False	True
Додаткові бали			
Ідентифікатор	int	False	True
Студент	int	False	False
Семестр	int	False	False
Опис	text	False	False
Бал	real	False	False
Користувачі			
Ідентифікатор	Int	False	True
Логін	varchar(50)	False	True
Пароль	varchar(32)	False	False
Ім’я	varchar(50)	True	False
Прізвище	varchar(50)	True	False
Відділення	int	False	False
Права доступу	int	False	False
По батькові	varchar(50)	True	False
Персонал ресурсу	boolean	True	False
Студенти			
Ідентифікатор	int	False	True
Користувач	int	False	True
Рік вступу	date	False	False
Група	int	False	False

Продовження таблиці 2.1

Форма навчання	varchar(50)	False	False
Освітні програми			
Ідентифікатор	int	False	True
Відділення	int	False	False
Назва	varchar(50)	False	False
Групи			
Ідентифікатор	int	False	True
Освіт. програм.	int	False	False
Назва	varchar(15)	False	False
Права доступу			
Ідентифікатор	int	False	True
Назва	varchar(15)	False	False
Відділення			
Ідентифікатор	int	False	True
Назва	varchar(50)	False	False
Файли студентів			
Ідентифікатор	int	False	True
Студент	int	False	False
Семестр	int	False	False
Файл	varchar(100)	False	False

Логічна модель даних — модель даних певної предметної області, проявлена вільно від визначеного продукту керування БД.

UML — стандартизована мова створення, застосовується у архетипу ООП. Є невіддільною одиницею у одноманітного розвитку створення програмних засобів. UML є мовою розтяжного спрямування, що застосовує графічні мітки заради утворення загальної моделі системи[3].

UML діаграми класифікуються на структурні та поведінкові діаграми.

Структурні UML діаграми:

- Діаграма пакетів.
- Діаграма класів.
- Діаграма компонентів.
- Діаграма об’єктів.

Поведінкові UML діаграми:

- Діаграма систематичності.
- Діаграма станів.
- Діаграма прецедентів.
- Діаграма діяльності.

Діаграма класів – представлення організації моделі. Показує статичні частини: типи даних, зміст, відношення, класи.

Діаграма класів системи (рисунок 2.3).

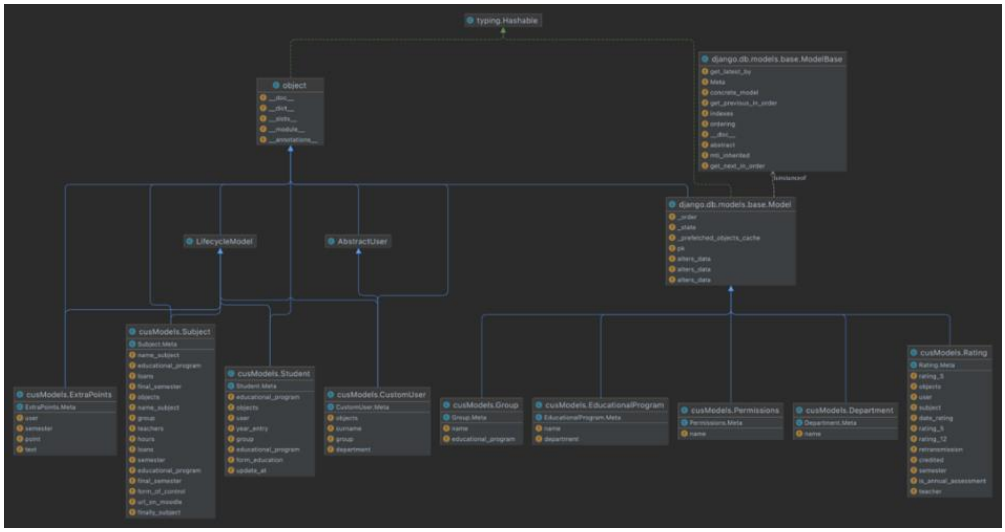

Рисунок 2.3 – Діаграма класів

Діаграма станів – діаграма, яка показує зміни станів компонента під час використання системи. Основними елементами є: коло початку, округленні прямокутники, яка розкриває різні стани, стрілка переходу з одного стану в інший, лінія об’єднання переходів. У системі реалізовано наступні діаграми стану.

Діаграма стану Адміністратора системи (рисунок 2.4).

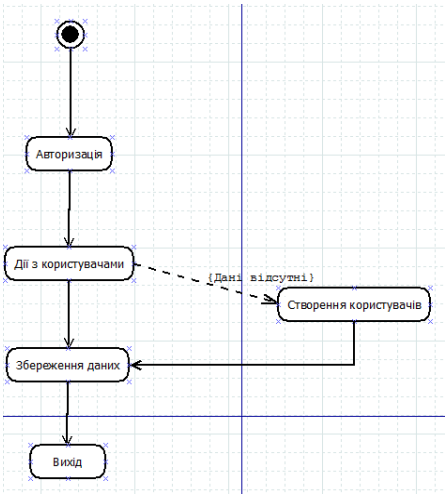

Рисунок 2.4 – Діаграма станів Адміністратора

Діаграма стану Студента (рисунок 2.5).

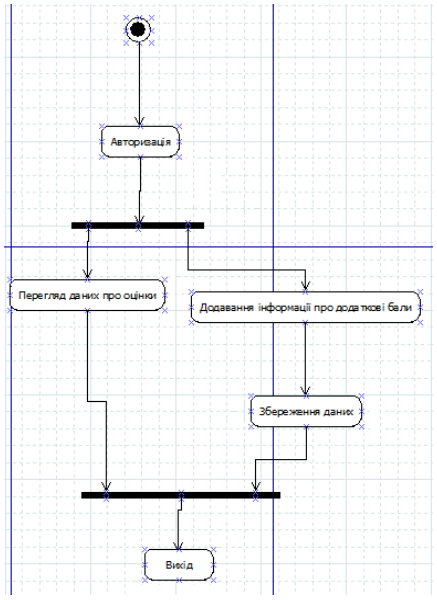

Рисунок 2.5 – Діаграма станів Студента

Діаграма стану Методиста (рисунок 2.6).

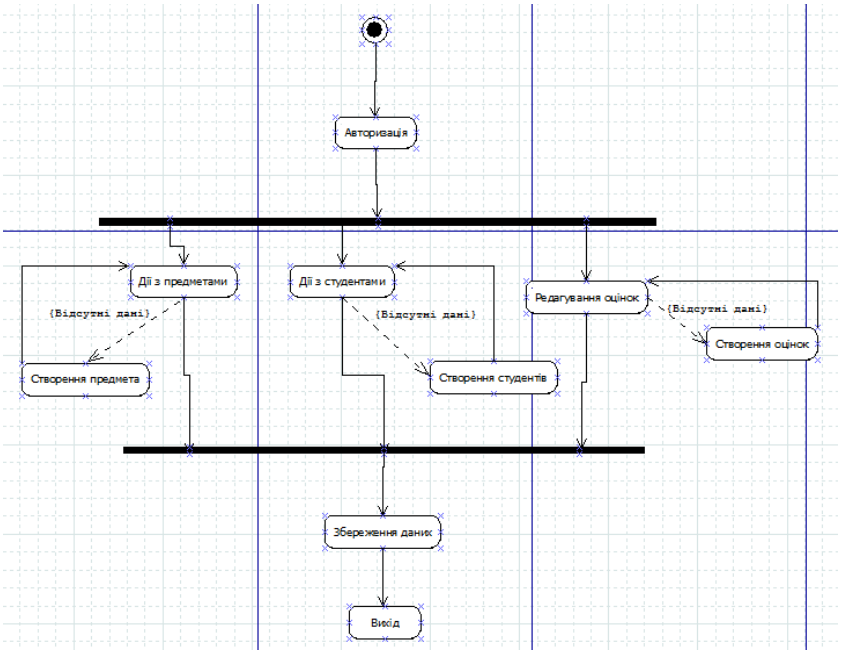

Рисунок 2.6 – Діаграма станів Методист

Діаграма прецедентів включає в себе з частин прецедентів(способи використання методів), акторів(категорій користувачів) та асоціацій між ними. Основна суть даного типу діаграм є в тому, що система спроектована у вигляді множин акторів і методів. Акторами даної системи є: Адміністрація, Адміністратор, Методист, Студент. Методи кожного з акторів описано нижче (рисунок 2.7).

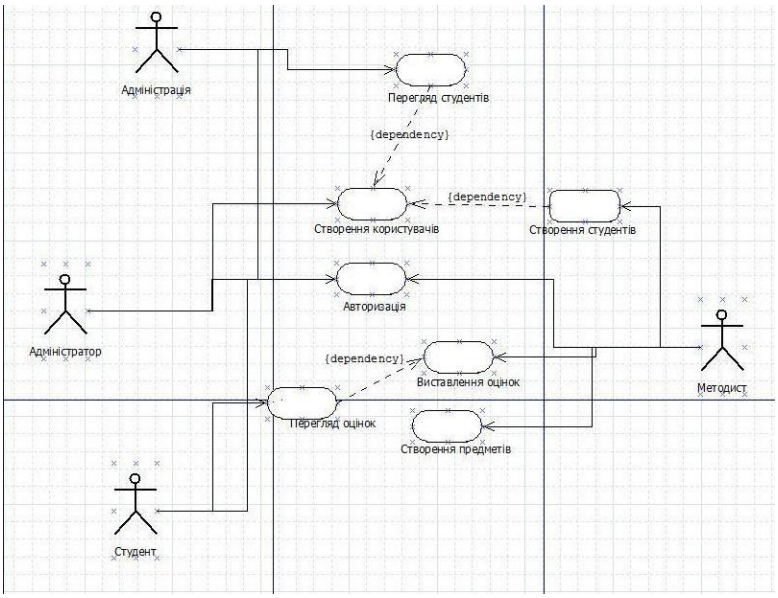

Рисунок 2.7 - Діаграма прецедентів системи

2.3 Представлення основного функціоналу та бізнес-логіки програмної реалізації проєкту

При розробці інформаційної системи визначено кілька категорій користувачів з певною різними потребами, відповідальності та можливостей, а також правами доступу до інформації.

За результатом проектування інформаційної системи, буде створено веб-застосунок з різними функціональними можливостями та користувацьким інтерфейсом для кожної із груп користувачів. Список дії кожної групи може відрізнятися, а також в чомусь бути подібним.

В даній інформаційній системі було розроблено додатки для наступних типів користувачів (рисунок 2.8).

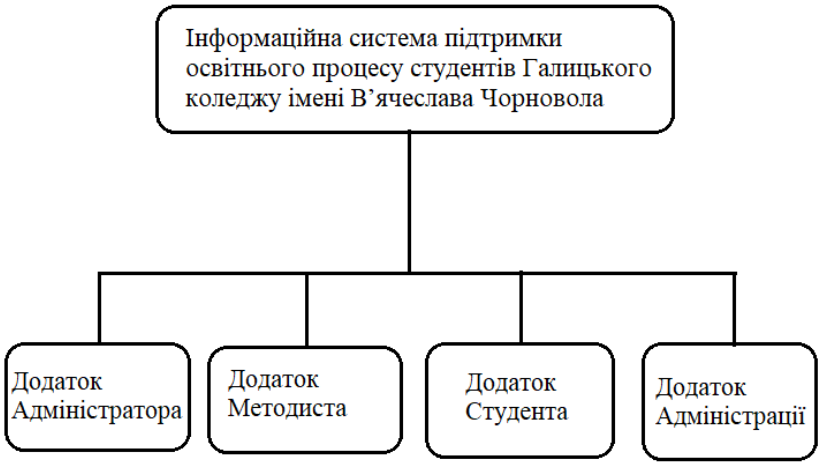

Рисунок 2.8 – Користувачі інформаційної системи

Відповідно з обраними видами користувачів у додатку, виокремимо наступні перспективи при роботі для кожного.

Користувач Адміністратор має можливість авторизації в системі, після чого отримує відповідні можливості. Основною функцією адміністратора є створення користувачів у системі. Окрім цього, він має змогу переглядати дані про предмети, студентів, викладачів тощо. Діаграма функціональних можливостей користувача адміністратора (рисунок 2.9).

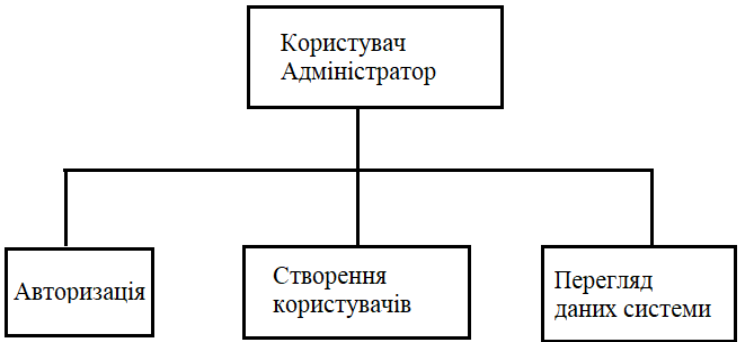

Рисунок 2.9 – Функціональні можливості Адміністратора

Користувач Методист має можливість авторизації в системі, після чого отримує відповідні можливості. Основними функціями методистів є створення студентів і розподілення їх по групах, дії з предметів, виставлення оцінок тощо. Окрім цього має можливість переглядати дані про успішність студентів, створення рейтингових списків. Діаграма функціональних можливостей користувача методиста (рисунок 2.10).

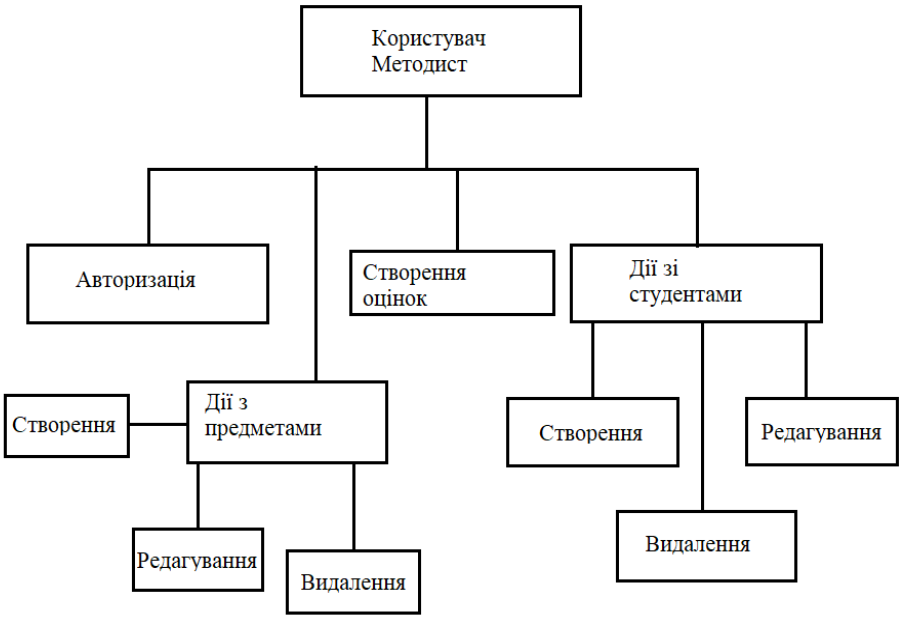

Рисунок 2.10 – Функціональні можливості Методист

Користувач Студент має можливість авторизації в системі, після чого отримує відповідні можливості. Основною функцією студентів є перегляд інформації про успішність навчання. Окрім цього має можливість надавати

нагороди за позанавчальний час для додаткових балів. Діаграма функціональних можливостей користувача студента (рисунок 2.11).

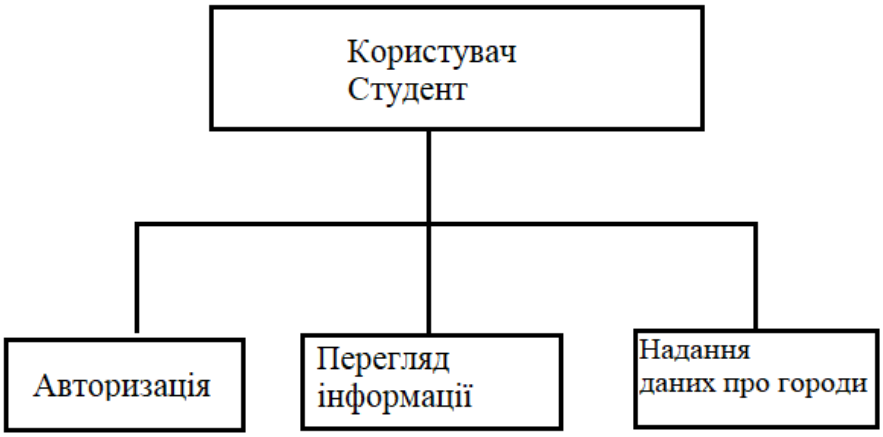

Рисунок 2.11 – Функціональні можливості Студента

Користувач Адміністрація має можливість авторизації в системі, після чого отримує відповідні можливості. Основною функцією адміністрації є перегляд всієї інформації про студентів закладу. Діаграма функціональних можливостей користувача адміністрації (рисунок 2.12).

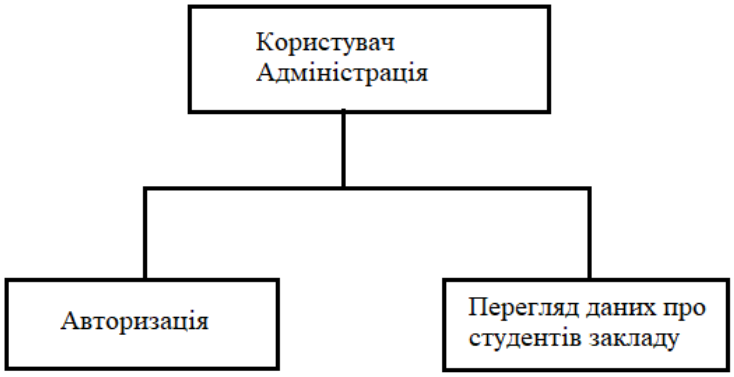

Рисунок 2.12 – Функціональні можливості Адміністрації

2.4 Проєктування інтерфейсу

Навігація веб-застосунку – це та можливість, яка допомагає клієнту швидко віднаходити необхідні дані. Вона спирається на логічну організацію додатка і пособляє оперативно рухатися по ній. Для веб-застосунку передбачено наступні види користувачів: адміністратор, методист, студент,

адміністрація закладу. Кожен з них має спільні та відмінні сторінки. У клієнтів в наявності реалізована навігації по сторінках системи.

Якщо не основною, то однією з основних частин будь-якого додатка є створений для клієнта інтерфейс.

Веб-застосунок дозволяє здійснювати:

- Оперативний пошук потрібних даних.
- Можливість авторизації.
- Можливість операцій над предметами та студентами тощо.

Також як для всіх додатків, для веб-додатків зовнішний вигляд грає важливу роль. Він дає розробникам впізнаваність та обличчя їхнього продукту. Звідси можна відокремити основні частки для створення стилю:

- Шрифт тексту – повинен мати відповідні три характеристики(колір, розмір, накреслення).
- Колірна палітра веб-сторінок – обрання трьох кольорів для сторінки, які мають застосовуватися для показу стандартного тексту, посилань і провіданих посилань. Вона має бути однаковою для всіх сторінок ресурсу, це покажу користувачу ілюзію єдності системи.
- Графічне опрацювання додатку – повинен підходити загальній колірній.
- Схемі системи.
- Стиль всіх веб-сторінок системи має бути розроблений в однаковому стилі.

Для цього потрібно спочатку виготовити макет сторінки. Варіант розробки макету зручний тим, що спочатку створюється макет однієї сторінки, а після цього решта сторінок просто наслідуються від неї. Нижче показано процес проектування макетів системи.

Елементами, які будуть повторюватися на кожній сторінці, будуть:

- Назва.
- Логотип навчального закладу.

– Навігаційне меню.

– Контент (вміст веб-сторінки).

Основною задачею створення веб-системи – це надання клієнтам простий, котрий має багато функцій інтерфейсу системи для роботи з системою.

Головними чинниками, якими можна оцінювати зручність користування додатків, є:

– Рівнозначність розробленого інтерфейсу. Рівнозначність розробленого для клієнта зовнішнього вигляду додатку – це його розроблена належним чином тим задачам, котрі клієнт має розв'язувати за їх допомогою.

– Зрозумілість. Додаток має бути простим та зрозумілим для того, щоб клієнт системи, котрий раніше ніколи її не бачив, добре міг в ній розібратися і правильно застосовувати.

– Ефективність. Додаток не має заважати роботі досвідчених клієнтів, котрі тривалий час користуються нею.

Першою сторінкою авторизація (рисунок 2.13). На присутня форма, з полем для логіну та пароля, яку потрібно заповнити.

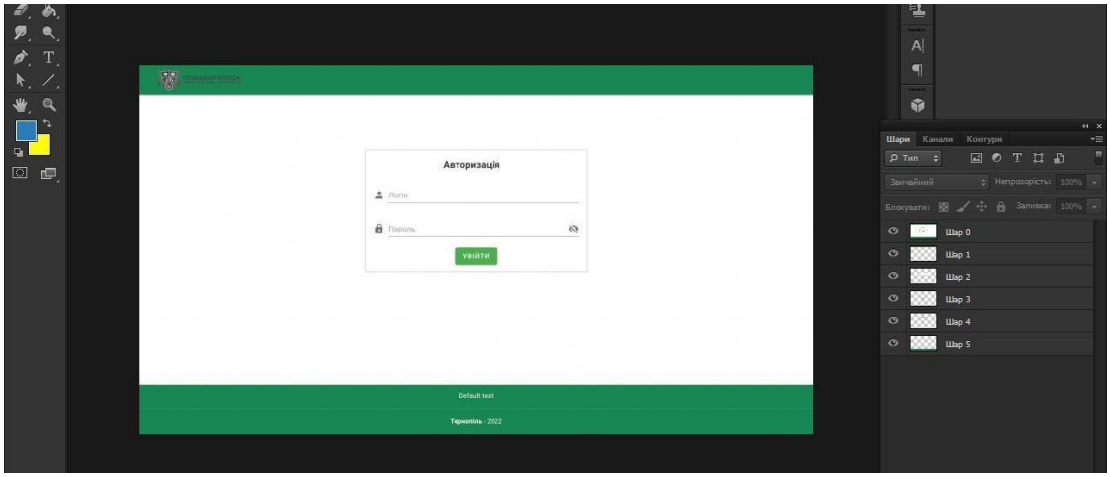

Рисунок 2.13 – Макет сторінки авторизації

Якщо користувач заходить, як адміністратор, то його перенаправляє на головну сторінку Адміністратора сайту. У даній дипломній роботі для

користувача Адміністратор було використано стандартні шаблони Django, які запропоновані за замовчуванням.

Якщо користувач заходить, як методист, то його перенаправляє на головну сторінку Методиста сайту (рисунок 2.14). На сторінці присутнє навігаційне меню, таблиця з даними та кнопками, для редагування та видалення. Окрім цього, є кнопка, натискаючи яку відкривається модальне вікно з формою для заповнення. Також є модальні вікна фільтрації даних.

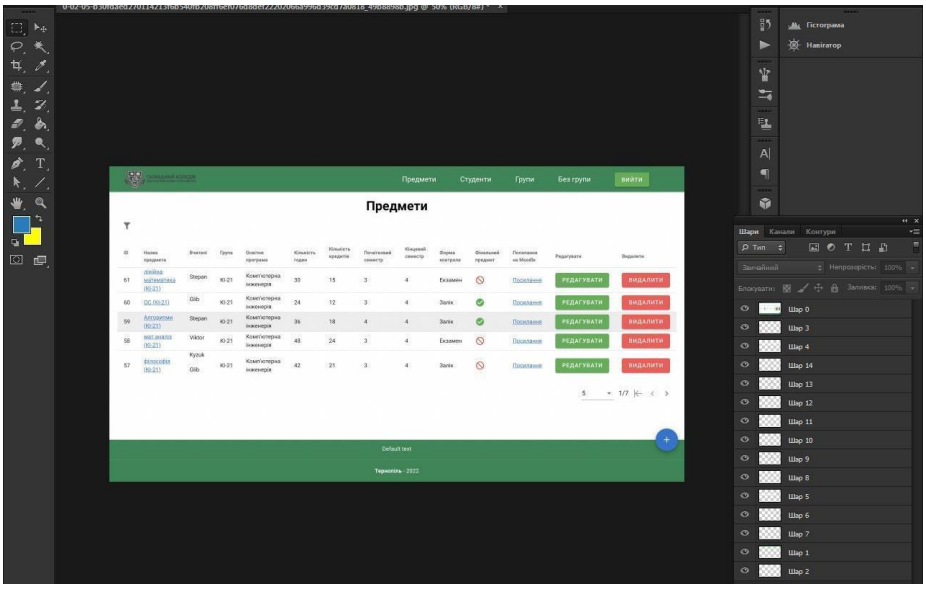

Рисунок 2.14 – Макет сторінки предметів

Також є сторінка студентів з відповідним функціоналом (рисунок 2.15).

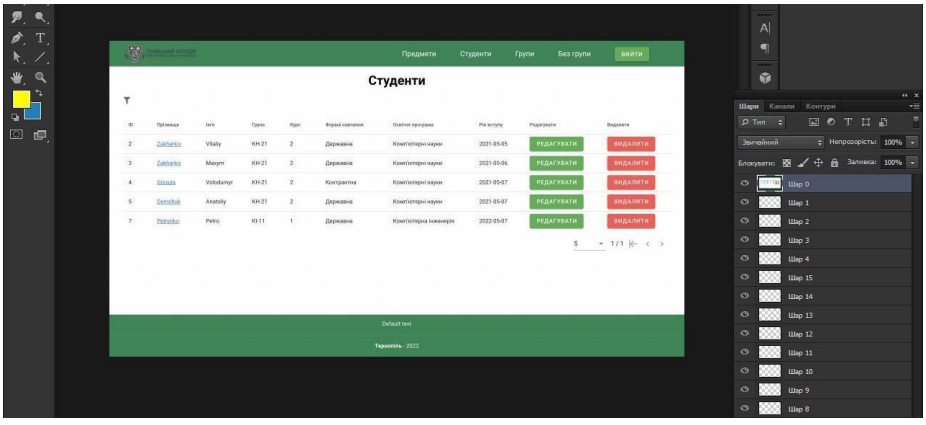

Рисунок 2.15 – Макет сторінки студентів

На сторінці Без групи присутні не розподілені по групах студенти відповідного відділення (рисунок 2.16). Щоб створити студента потрібно

вибрати конкретного студента і натиснути кнопку Додати. Тоді відкривається модальне вікно з формою для заповнення.

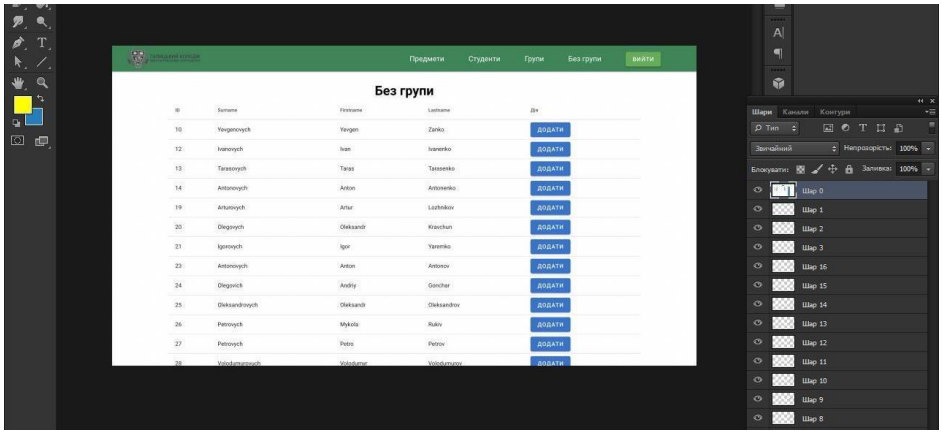

Рисунок 2.16 – Макет сторінки невизначених студентів

На сторінці групи можна вибрати певну групу та певний семестр для перегляду рейтингу і загальних оцінок (рисунок 2.17).

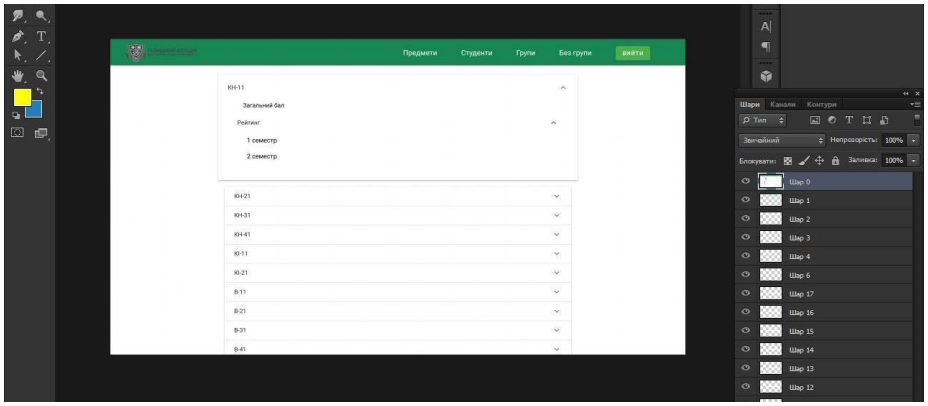

Рисунок 2.17 – Макет сторінки груп

Щоб добавити додатковий бал потрібно на сторінці Групи вибрати групу і потрібний семестр. Тоді відкривається сторінка для добавлення додаткових балів (рисунок 2.18). Щоб додати бал потрібно натиснути кнопку Додати бал. Відкривається модальне вікно для додавання додаткового балу. Окрім цього якщо натиснути на вже існуючий бал його можна редагувати та видалити.

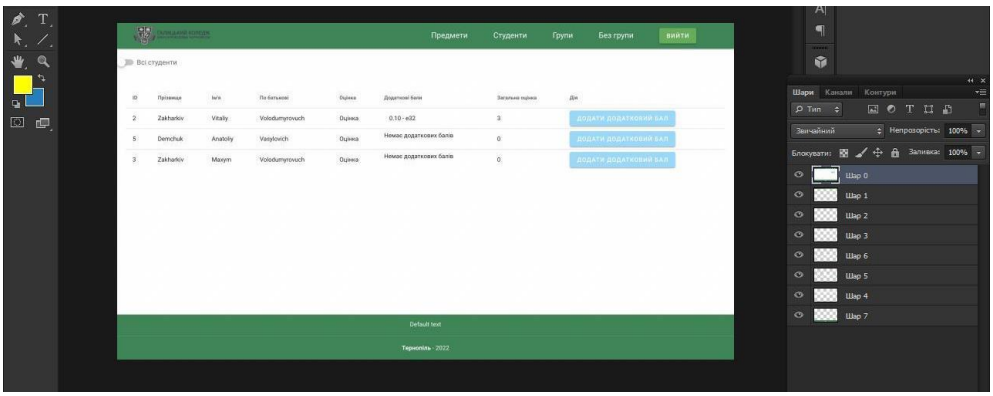

Рисунок 2.18 – Макет сторінки додаткових балів

Щоб поставити оцінку потрібно на сторінці Предмети вибрати будь-який предмет. Після цього відкривається сторінка з семестрами, де вибираємо в якому семестру буде оцінка (рисунок 2.19).

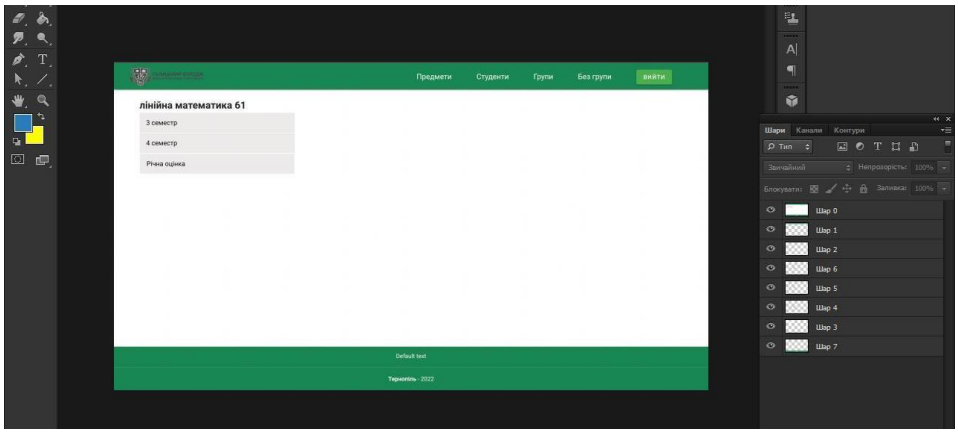

Рисунок 2.19 – Макет сторінки семестрів

Після цього відкривається сторінка з добавленням сторінки. На ній наявна таблиця з студентами групи і поля для виставлення оцінки (рисунок 2.20).

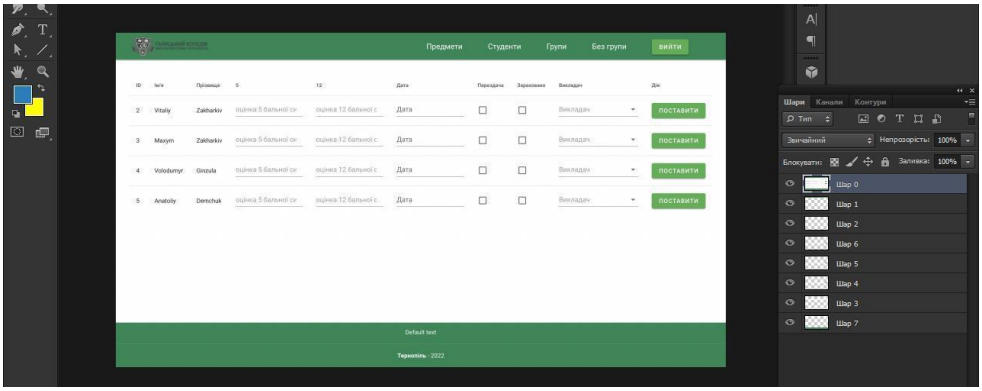

Рисунок 2.20 – Макет сторінки оцінок

Як тільки оцінка була виставлена є можливість зразу оцінку відредагувати.

Якщо користувач заходить, як студент, то його перенаправляє на головну сторінку Студент сайту.

Сторінка спроектована таким чином, що вона поділяється на три частини,

які переключаються спеціальною навігацією. Макет для цих частин є ідентичним (рисунок 2.21):

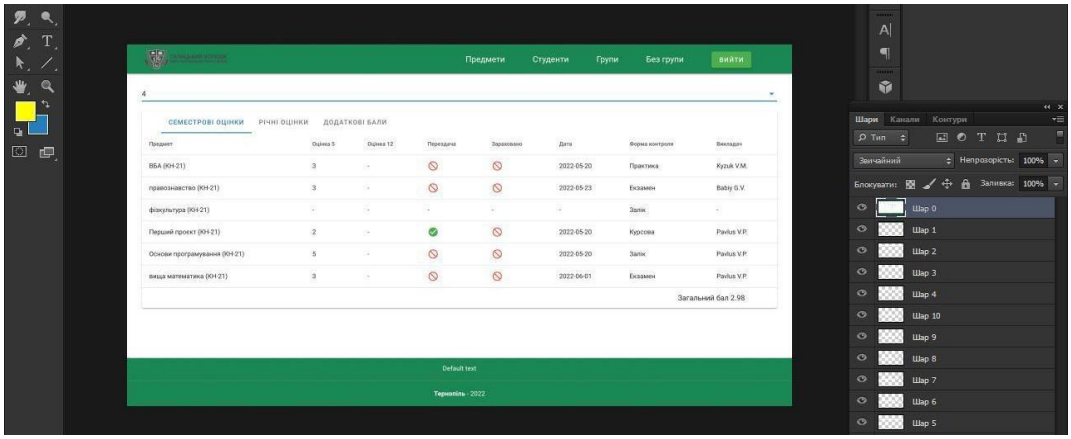

Рисунок 2.21 – Приклад макету

Отже, у другому розділі дипломної роботи було обрано тип архітектури веб-додатку, проектування бази даних та інтерфейсу системи. Розкрито основну інформацію, котру система має зберігати та обробляти, з’ясований основний функціонал, який повинен застосовувати веб-застосунок, для надання потрібної інформації для користувачів різних категорій.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ СИСТЕМИ

3.1 Опис необхідного системного, прикладного та інструментального програмного забезпечення

ПЗ за своєю роллю поділяються на системне, прикладне та інструментальне (рисунок 3.1).

Рисунок 3.1 – Програмне забезпечення

Системне ПЗ - це сукупність програмних засобів, які допомагають керувати резервами комп'ютера, здійснювати утримання дієздатності додатка керування даними, збільшити ефективності її діяльності.

Операційна система - це велика кількість програмних засобів, які здійснюють підтримку в керуванні, взаємодії всіх частин комп'ютера під час роботи і реалізація задач. Здійснює роботу та взаємозв'язок комп'ютерних приладів, як єдиної системи, скоординував взаємини з іншими приладами в мережі.

Серед ОС були використані:

- MacOs.
- Windows 10.

Прикладне ПЗ потрібне для управління інформацією у будь-якій сфері використання, застосовує необхідні для клієнта наявні функціональні можливості[4]. Прикладний застосунок створюється для певної ОС. Наприклад: середовище розробки, бази даних тощо.

Середовище розробки ПЗ – це середовище, яке доводить до автоматизму або розширює додаток, який співпрацює у ланцюгу створення ПЗ. Найпопулярнішими з яких є продукти розроблені JetBrains. Компанія JetBrains – чільний всесвітній виробник професійних способів створення різних програмних розробок.

PyCharm – це об'єднане програмне середовище розробки для мови кодування Python (рисунок 3.2, 3.3)[5].

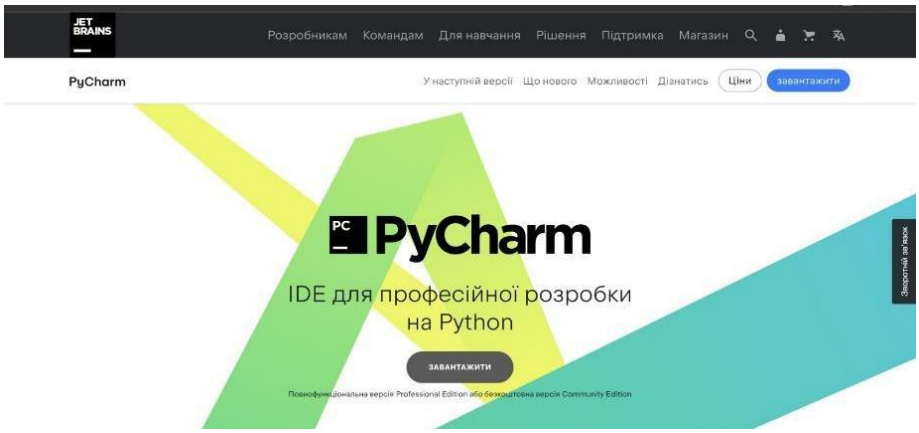

Рисунок 3.2 – Сторінка для завантаження PyCharm

Наявні наступні функціональні можливості:

− Фіксована перевірка програмного коду, виокремлення помилок та синтаксису середовища.

− Редагування коду.

− Інструментальні засоби для створення веб-додатків за допомогою фреймворку Django тощо.

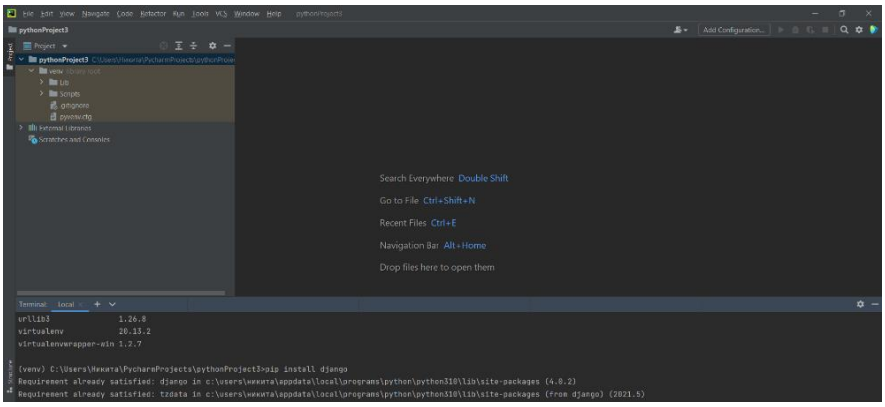

Рисунок 3.3 – Зовнішній вигляд редактора PyCharm

JetBrains WebStorm – це об'єднане середовище розробки для JavaScript, HTML, CSS (рисунок 3.4, 3.5).

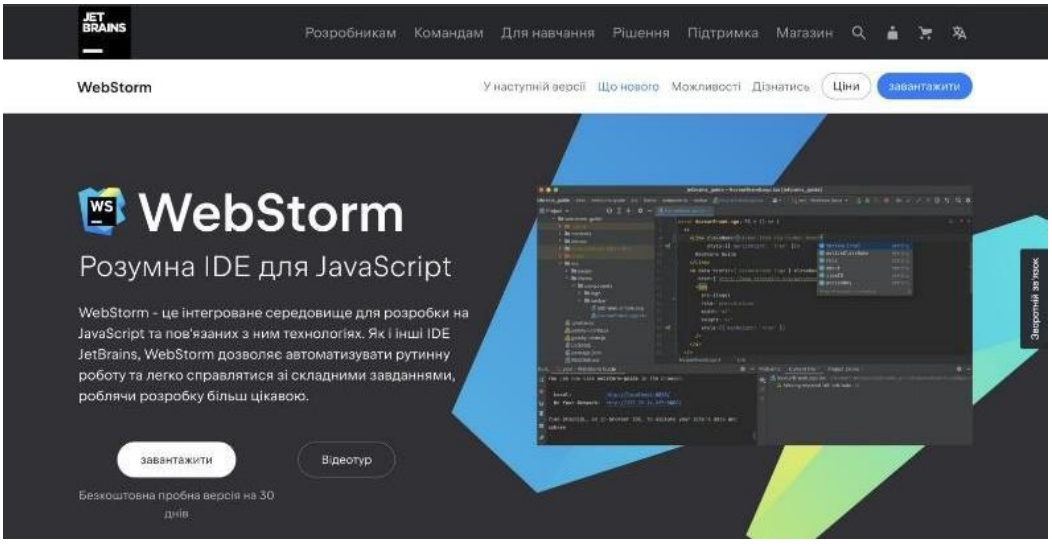

Рисунок 3.4 – Сторінка для завантаження WebStorm

Наявні наступні функціональні можливості:

– Зливання із системними засобами, які допомагають віднаходити помилки під час розробки.

– Видозміна файлів .html, .css, .js із паралельним переглядом підсумків.

– Дальне розвертання по протоколу FTP, чи йому подібними.

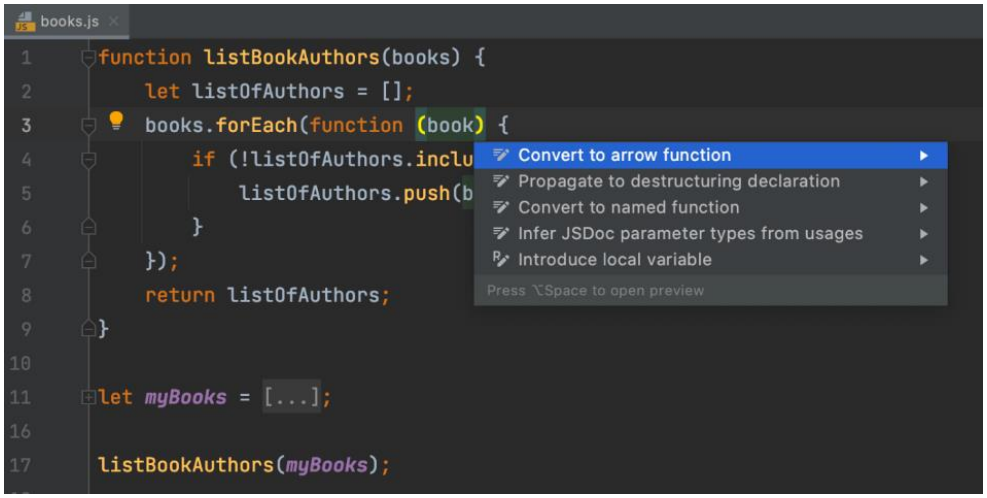

Рисунок 3.5 – Зовнішній вигляд редактора WebStorm

База даних – це спосіб накопичення та упорядкування даних[6]. Для керування будь-котрої БД присутні відповідні менеджери, які, за допомогою графічного інтерфейса, дозволяють легко маніпулювати таблицями.

Для SqLite було обрано SqLiteStudio (рисунок 3.6).

Рисунок 3.6 – Сторінка для завантаження SqLiteStudio

Інструментальне ПЗ – це програмні засоби, які допомагають створювати нові системи та додатки. Використовуються розробниками як засіб на технологічних рівнях ходах проектування, створення, організація, обстеження систем. До нього відносяться мови і системи програмування тощо.

В теперішньому часі для розробки, підтримки будь-якої системи повинна існувати клієнтська та серверні сторони.

Клієнтська сторона(frontend) – це опрацювання зовнішнього вигляду та функціоналу, який виконується на стороні користувача системи[7]. Це все те, що розглянути клієнт системи, розкриваючи веб-сторінку. Базовими технологіями для реалізації клієнтської сторони є:

HTML – це шаблона мова розмітки веб-документів для огляду у браузері. Браузери одержують html-файл або від сервера протоколом HTTP, або від локального диску комп’ютера, далі інтерпретує вміст файлу в зовнішній вигляд, який показується на екрані. Елементи html призначені для побудови html-сторінок. За сприянням html елементів різні побудови можна відобразити на сторінці. Наприклад, форма авторизації, медіа-контент, навігаційне меню тощо. HTML дає можливість для утворення абзаців, цитат,

заголовків, списків тощо. HTML теги створюються за допомогою кутових дужок.

CSS – мова яка призначена, для стилізації html-сторінок. Як правило css застосовують для візуального відображення html або xhtml форматах, проте також css застосовується для інших xml-документів. Кольори чи шрифти тексту, верстку сторінки, кольорове оформлення сторінки, адаптив сторінок – це невеликий перелік можливостей css.

JavaScript – це об’єктно-орієнтована, динамічна мова програмування. Як правило, застосовується для утворення функціоналу веб-сторінок, що дає можливість системі комунікувати з користувачем на боці клієнта, змінювати зовнішній вигляд сторінки, асинхронно отримувати та відправляти інформацію на сервер тощо. JS систематизують з скриптовую, прототипную мовою програмування. Окрім цього, js підримує ще наступні парадигми програмування: функціональну, імперативну парадигми.

Серверна сторона (backend) включає в себе апаратну та програмну частини, кажучи простими словами, все те, що здійснюється під капотом застосунку, його серверна сторона[8]. Основною задачею бекенду – є необхідність створення зв’язку базу даних із клієнтською стороною, яка після чого має показати інформацію у вигідному для клієнта форматі. І навпаки, все що виконується на клієнтській стороні повинно надсилатись у базу даних через серверну сторону. Серверна сторона застосовується у таких мовах програмування: Python, C#, PHP, Ruby, Perl, Java, тощо.

База даних – це поставлена побудова, яка використовується для утримання, редагування та обробки зв’язаних даних, великих обсягів.

Бази даних застосовуються для застосунків з великими об’ємами даних: корпоративний сайт, портал, інтернет-магазин.

Основним плюсом БД є можливість швидкого занесення та застосування необхідних даних. За допомогою умисним алгоритмам, що застосовуються для баз даних, можна просто шукати потрібні дані за мінімальну кількість

часу. Окрім цього, в базі даних є взаємозалежність даних: редагування даних в одному місці зумовлює зміни даних в зовсім іншому місці — це спрощує роботу з даними.

Виокремлюють дві категорії БД – нереляційні та реляційні.

Реляційна БД – це база даних, створена на реляційній моделі інформації. Для користування реляційними бази даними використовують реляційні СУБД. Кажучи іншими словами, реляційна база даних – це БД, що приймається клієнтом як комплект налагоджених відносин відмінних рівнів.

Реляційна БД є комплекс частин інформації, структурованих у форматі комплекту офіційно відображених таблиць, в яких інформація може бути прийнятним або знову зібраних багатьма варіантами.

До популярних реляційних СУБД відносять: MySQL, Sql Server, SQLite, PostgreSQL.

Позитивні сторони реляційної бази даних:

- Легке та зрозумілих на погляд користувачу. Одною вжитою інформаційною структурою є таблиця.
- Суворі норми проектування.
- Цілковита автономія інформації.
- Для створення запитів і створення практичного програмного забезпечення немає потрібності знати конкретну структуру бази даних.

Негативні сторони реляційної бази даних:

- Не постійно практичне застосування може бути у форматі таблиці.
- Результатом проектування появляється багато таблиць.
- База даних займає не мало зовнішньої пам'яті.
- Невисока швидкість ходу до інформації.

Нереляційні бази даних – це відмітка широкого групи різних систем керування БД, які появилися в кінці 2000-х – початку 2010-х років і значно різняться від стандартних реляційних СУБД із ходом до інформації засобами мови запитів SQL[9]. Використовується у системах, в яких виконується дослід

уладнати питання доступу і масштабування за рахунок повного або часткового відмовлення від правил атомарності та погодження інформації.

Позитивними сторонами нереляційної бази даних є:

− Наявність функціоналу для збереження великих об’ємів неорганізованих даних. У NoSql відсутній правила для типів даних, які зберігаються.

− Нереляційні бази даних краще масштабуються, а ніж реляційні. Вони вимагають набагато менше ресурсів для цього.

− Дозволяють використовувати сховища і хмарні обчислення тощо.

Негативні сторони нереляційної бази даних:

− Недолік спритного переходу з однієї бази в іншу.

− Додаток сильно прив’язаний до певної СУБД.

− Немає в наявності достатньої кількості функціоналу, звідси частину приходиться створювати самому тощо.

Клієнтська сторона реалізована за допомогою мови кодування Js. На JavaScript реалізовано безліч фреймворків та інших доповнень. Найпопулярніші з яких: Vue, Angular, React, Jquery.

VueJs – це передовий JavaScript фреймворк, який допомагає розробляти фронтенд для веб-додатка[10]. Головною різницею від фреймворків монолітів, VueJs створено відповідним до покрокового застосування. Основним завданням ядра VueJs - це вирішення задач ступеню презентування, що полегшує об'єднання з різними доповненнями, та готовими додатками. Також, VueJs цілком повністю спроможний для розроблення непростих односторонніх додатків (SPA), якщо користуватися разом з новітніми інструментами та іншими доповненнями. Реактивне програмування – це образ кодування, яка піднята на хвилях інформації і поширення її редагування. Простими словами можна сказати, що у мовах кодування повина бути функція, яка може просто виявити фіксовані та нефіксовані потоки інформації, а

розробка моделі реалізації є динамічне поширення перемін через потоки інформації.

VueJS застосовує Virtual DOM. Переміни не заносяться в DOM, а лише створюється дублікат DOM. Якщо необхідно внести зміни, вони заносяться до JS структури. Тоді кінцеві зміни заносяться до існуючого DOM, яке бачить клієнт. Цей спосіб вдалий у плані оптимізації, йому не потрібно великої кількості ресурсів.

Головними завданнями VueJs є:

- Створення непростих динамічних інтерфейсів.
- Показова візуалізація.
- Здійснення графіки за підтримки CSS.
- Поєднання інформації тощо.

Важливою складовою VueJs є компоненти.

Компоненти – це частини коду, які потрібні для багатократного користування, що містить характеристики зовнішнього вигляду елементів системи, і виконання функціональних потреб проекту. Вони сприяють розробникам у розробці модульної програмної основи, яка легко підтримувати.

VueJs відповідає також маленьким проектам, яким потрібно приєднати динамічності, показати форму за підтримки AJAX, підкреслити важливість під час запровадження інформації клієнтам. VueJs просто масштабується та вдало відповідає вимогам великих проектів.

VueJs окрім цього перкрасно збігається для розробки немалим односторінкових застосунків, дякуючи власним важливим частинам, наприклад таки як: Router та Vuex. VueJs застосовують, як масові API для розробки систем, так і створювати серверні-додатки.

Яка різниця Vue.js від його аналогів?

Компоненти - це та частина, на які грунтується робота всіх JS-фреймворків. Але потрібно зазначити, що Vue та React прекрасно

відповідають для зміни "німих компонентів" - це маленькі компоненти, яким не наявні ситуації функцій, що одержують вхідну інформацію та вертають частини як виставлення.

Гнучкість користування – це перспектива працювати з VueJs, добавивши доповнення Javascript у код. Це нереально у варіанті з Angular, через те що він використовується для складніших задач. Якщо говорити про мікропрограми і мікросервіси, то VueJs, на відміну від інших фреймворків, дає сильніший інспекцію над об'ємом системи, даючи право обирати необхідні у ситуаціях елементи.

Швидкість та об'ємністю файлів - фреймворк Angular найбільш по розмірах від інших. У VueJs в наявності є Virtual DOM, що допомагає створювати дублікат об'єктного показу організаційного файла та дає працювати із візуальним дублікатом. Цей спосіб дає можливість збільшити ефективність фреймворка, звідси система працюватиме швидше.

Vuex - це доповнення яке допомагає, управління становищем, навмисно налагоджених для розробки важких, великих систем VueJs[11]. Вона користує загальне, централізоване сховище для частин у системі.

Vuex створений наступним чином: його становище нереально відредагувати з будь-котрого компоненту. Це допомагає в тому, що становище може бути відредагованим тільки очікуваним методом. Звідси, сховище являється одним осередком правди: щоб перешкоджати компонентам системи зашколити становище, всі частини інформації приберегтися тільки один єдиний раз, тільки для читання.

У патерна Vuex є перелік позитивних та негативних сторін:

− Патерн Vuex є динамічним. Як тільки елементи дістають становище з сховища, вони динамічно обновляються тоді, коли становище переобразовується.

− Елементи не мають можливості напряму модифіковувати становище сховища. Одним варіантом модифікувати становище сховища - це використання спеціальних методів які називаються мутацій.

− Патерн Vuex дає зрозумілий обзор, як усе налаштовується і відбивається на систему.

− Легше утримувати та скоординовувати становище між декількома елементами.

− Vuex виконує реальну просту співпадіння поміж перехресними елементами.

Для чого користуються Vuex?

Це одне джерело правди. Vuex – це зосередженне сховище становищ всіх шляхів і елементів, які користуються у системі. Всі елементи будуть питатись інформацію у сховищі, а тільки тоді будуть повернуті у зворотному напрямку вже відредаговані. За допомогою цього зменшують чисельність похибок і збільшують безпеку додатку.

Одни варіант заклику до інформації. Щоб одержати інформацію з сховища, програмісти розробили гетери, окликуються до них окрім гетерів - сетери, за допомогою яких можна повертати змінений становище сховища назад. Все це дає системі щонайбільше автоматизувати діяльність з інформацією, а програмістам спрощує розробку.

Модульне збереження інформації. Vuex може бути не єдиним об'ємним файлом. За допомогою цього одне велике сховище розбивають на декілька менших.

Динамічність. Одним з основних плюсів VueJs – це наявність уявного дерева, поновлювалюються ті елементи зовнішнього вигляду, становище котрих змінився. Vuex також утримує цю динамічність, машинально добавляючи зміни в віртуальному дереві.

Vuetify – це ціла форма зовнішнього вигляду клієнта, збудована на базі VueJs. Намір розробки – дати програмістам елементи, потрібні для

започаткування великих та спокусливий клієнтських перспектив. У порівнянні з різними іншими бібліотеками, Vuetify створено з самого початку таким чином, що ним доволі легко опанувати на навчитись.

Vuetify застосовує чутливе ставлення до зовнішнього вигляду, простими словами кажучи, розробка буде працювати на всіх платформах.

З часу свого релізу в 2014 році VueJs постав та залишився, одним серед загальновідомих фреймворків JS. Причиною його визнаності є поширене користування елементів, які допомагають програмістам розробляти короткі додатки для користування. Бібліотеки зовнішнього вигляду клієнта – це набір цих частин, що реалізовані певні вказівки з приводу дизайну та дають потрібні елементи для розробки об'ємних проектів.

Vuetify створено відповідно до стандартів Material Design, причому всі елементи старанно створенні, щоб бути модульним, адаптивним та ефективним. Vuetify в наявності динамічний ряд опрацювання і редагування протягом тижня.

Axios – це патерн з доступним кодом, який дає виконувати запити HTTP. Він дає методи .get(), .post(), .put() тощо.

Основні характеристики axios:

- Сприяння Promise API.
- Затримує запити.
- Змінює інформацію запиту.
- Відміняє запити.
- Динамічне змінення для JSON інформації.
- Утримання на клієнтській стороні для безпеки від XSRF.

Встановлення технологій на клієнтську сторону:

- VueJs прогресивний фреймфорк JS (рисунок 3.7).

Рисунок 3.7 – Команда для встановлення VueJs

– Vuetify – UI бібліотека VueJs (рисунок 3.8).

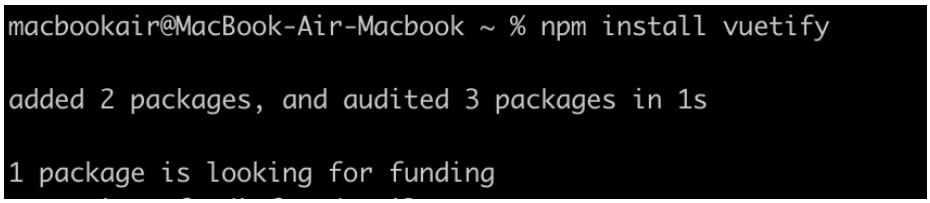

Рисунок 3.8 – Команда для встановлення Vuetify

Серверна сторона реалізована за допомогою мови кодування Python[12]. Python – це інтерпретована об'єктно-орієнтована мова кодування високого рівня. Написана в 1990 р. Гвідо ван Россумом. Організація даних високого ступеню спільно з динамічною семантикою та зв'язуванням зробили її звабливою ради стрімкої створення проєктів. Інтерпретатор та модулі за замовчуванням, присутні і у відібраній, і у кінцевій конфігурації на усіх передових майданчиках. В мові програмування Python придержуються декількома підходами кодування, наприклад: об'єктно-орієнтований, функціональний, процедурний та аспектно-орієнтований.

Відмінностей між Python та інших інструментів програмування достатньо немало, головні з яких:

– Управління пам'яттю - абсолютно автоматичне. Не слід турбуватись з приводу поділу або звільнення пам'яті.

– Типи пов'язані з об'єктами. Це свідчить, що змінній можливо призначити значення будь-якого різновиду, і що список має можливість містити об'єкти будь-яких типів. Сакраментальні мови не мають данної можливості.

– Операції безумовно реалізуються в більш високому щаблі. Це певною мірою результат того, як реалізована мова програмування.

Переваги:

– Гнучкість – провідна перевага мови, внаслідок чого дякуючи своїй гнучкості мова заробила популярність поміж багатьох програмістів.

− Можливість розширення – що сповна пояснює, наскільки мова збагачується. Присутні модулі і фреймворки для будь-який задач. Теж сильною стороною є те, що існує варіант використання С коду разом з Python.

− Легкий синтаксис.

− Інтерпретованість. Інтерпретатор Python наявний для усіх широко відомих платформ та застосовується за замовчування в більшості дистрибутивів Linux.

− PEP – основний стандарт для створення коду на Python. Це робить йогонайдійнішим, підтримуваним і виразним.

− Open Source – це програмне забезпечення із вихідним кодом в наявності. При потребі можна змінити будь-яку частину модуля чи бібліотеки.

− Ком'юніті – довкола Python створилось доволі дружелюбна та приємна спільнота, котра готова допомогти вирішити будь-яке завдання новачку або досвідченому програмісту.

Усі перелічені плюси зробили мову популярною та дозволили рости чи малими кроками.

Недоліки:

− Продуктивність. Левова частка розробників, збігаються на думці, що Python не надто швидкий. Це зумовлено тим, що Python є інтерпретованою мовою програмування. Проте у порівнянні з рештую інтерпретованих мов очевидно, що Python програє в ефективності. Але це легко можна вирівняти дякуючи С реалізацій того чи іншого проблемного шматка коду.

− Динамічна типізація. Велику частину своїх ресурсів Python споживає завдяки динамічній типізації, але це не раз покривається внутрішнім кешуванням.

− GIL. У даний час це є чільною проблемою ефективності в Python, теж цим викликана не зовсім хороша реалізація багатопоточності.

Переважні сфери вжитку:

− Веб-розробка.

– Скрапінг.

– Аналіз та робота з даними, зокрема: машинне навчання, візуалізація і аналіз даних.

У створенні веб-застосунків в Python зустрічається велика кількість фреймворків, які дають змогу сконцентруватися на задачі, а не на реалізації безпечної взаємодії із клієнтами.

Основні із них: Django, FastAPI, Flask, Aiohttp.

Django — високорівневий веб-фреймворк із відкритим кодом для створення веб-систем.

Наявні функції Django:

– ORM, API доступу до бази даних з підтримкою транзакцій.

– вбудована візуальна частина для адміністратора.

– диспетчер для посилань завдяки регулярним виразам.

– Система шаблонів із тегами та спадкуванням.

– система кешування.

– Мультимовність.

– «generic views» - шаблони класів контролерів.

Архітектура Django похожа на «Модель-Уявлення-Контролер» (MVC) (рисунок 3.9). Контролер стандартної моделі MVC орієнтовно збігається із рівнем, котрий Django носить назву представлення, а логіка відображення представлення виконується на рівні шаблонів. Завдяки цьому рівневу архітектуру Django не раз називають "Модель-Шаблон-Представлення".

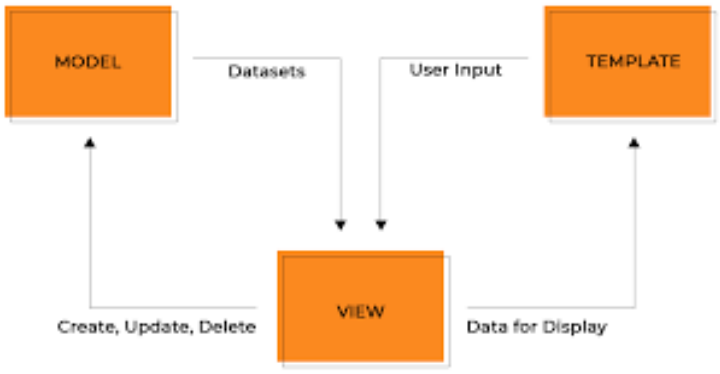

Рисунок 3.9 – Архітектура Django

При використанні js-фреймворка, сервер повинен презентувати дані у певному форматі.

І для додатків написаних на Django є додаткове розширення – Django Rest Framework і Django Ninja, котрі підтримують ідеологію REST .

REST API – це практичний програмний інтерфейс, котрий завдяки HTTP-запити отримає, вилучає та створює дані.

Форматом передачі даних у REST API є JSON (рисунок 3.10).

Рисунок 3.10 – Архітектура Django

Проста заміна об’єктами є провідною та фундаментальною частиною між застосунками.

Канони REST API:

- Один інтерфейс.
- Розділ клієнта та сервера.
- Немає збереження стану.
- Кешування завжди дозволено.
- Багаторівнева система.
- Код надається на запит.

Важливий мінус REST API – неміцна стійкість до злому. Аби захистити ресурси, слід дотримуватись наступних рекомендацій:

- Користуватись HTTPS.
- Використовувати безпечний спосіб автентифікації.
- Застосовувати CORS для лімітувати викликів.
- Звірити усі посилання кінцевих точок та дані запиту.
- Минати надання токенів API у клієнтському JavaScript.
- Заблокувати доступ із незнайомих доменів або IP-адрес.
- Локалізувати швидкість для запитів, що застосовуються один і той самий токен REST API.
- Налаштовувати відповіді згідно з кодом стану HTTP та кешування заголовка.
- Вносити до реєстру запити та спостерігати збої.

Архітектура REST API:

- URL-адреса контролера.
- Метод протокола HTTP.
- Заголовки протокола HTTP.
- Дані запита.

Методи HTTP:

- GET.
- POST.
- PUT.
- DELETE.

Django Ninja – це новітній та результативний фреймворк Python, що добре підходить для розробки RESTful API. Він має можливість опрацьовувати як синхронні, так і асинхронні запити та включають вбудовану підтримку перевірки даних, серіалізації JSON, авторизації, а також OpenAPI-документації.

Визначальні індивідуальності фреймворку:

– Розробники одухотворялися FastAPI, таким чином Django Ninja охоплює легкий мікрофреймворк із підтримкою декораторів, схожим на FastAPI-роути.

– Користується підказками типів у Python для оповіщення аргументів, що дає право виконувати перевірку та створювати OpenAPI документацію.

Pydantic - це небезкорисна бібліотека для розбору та валідації даних. Вона змушує типи запроваджувати до оголошеного типу, нагромаджує усі помилки, і все це так само непогано задокументовано.

Специфікація OpenAPI характеризує класичний інтерфейс до RESTful API, що дає право як комп'ютерам, так і людям знаходити та усвідомлювати функції обслуговування без доступу до вихідного коду. При коректному визначенні користувач має розуміти віддалену службу та співдіяти із щонайменшим об'ємом логіки реалізації.

Swagger – це спосіб, котрий дає змогу документувати REST-послуги (рисунок 3.11).

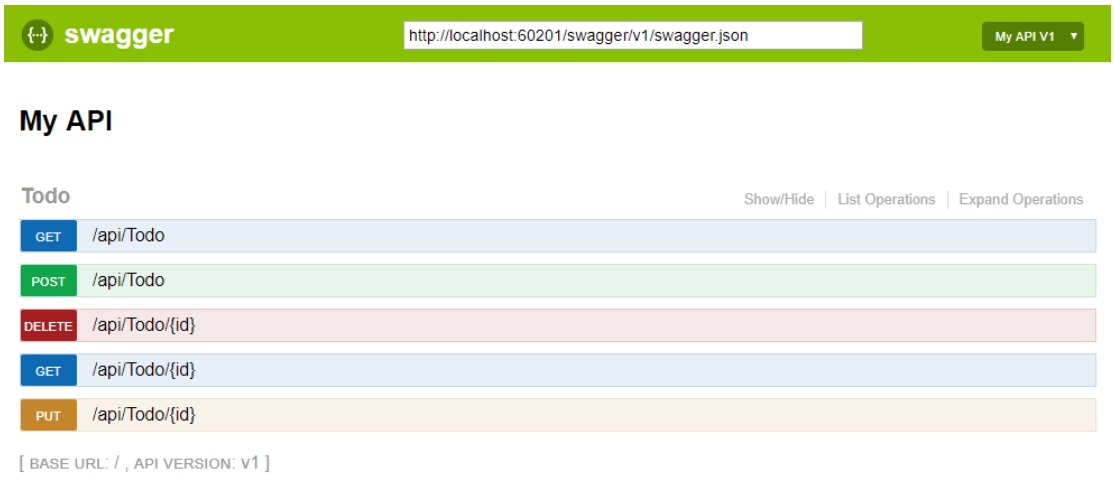

Рисунок 3.11 – Архітектура Django

Swagger має в наявності багато фреймворків і мов програмування. Крім цього, Swagger надає інтерфейс для дослідження документації, зрозумілому для клієнта і комп'ютера. Кардинальний його плюс полягає в додаткових інструментах.

Зберігання та модифікація документації є перший крок. Грядущий – зробити її доступною для усіх програмістів. Завдяки невеликій JavaScript модулю Swagger UI формує HTML деталі для усіх контролерів.

Django REST Framework — це бібліотека Python/Django із доступним вихідним кодом, котра спрямована на розробку складних веб-сервісів. Також він є досить гнучким із повнофункціональним комплектом інструментів з модульною архітектурою, що дає можливість робити розробку як простих кінцевих пунктів API, так і складних структур REST.

Фреймворк Django REST охоплює розлогий набір готових методів і є простим у використанні. Провідна думка зводиться до того, щоб чітко поділити модель, сумарне відображення представлень на основі класів, котрі дозволено сконфігурувати, аби задовольнити певний контролер за допомогою Serializer, котрий характеризує відображення поміж ними.

DRF надає низку плюсів, котрі роблять його кращим в порівнянні з іншими фреймворками для розробки API.

Наприклад:

− Він удосконалює зручність користування для розробників за допомогою реалізованому ним API для дослідження в Інтернеті.

− Програміст може налаштувати будь-які утиліти, як серіалізатори та контролери у фреймворку, і незалежно від цього створювати цілком функціональний API.

− Його документація доволі легка для розуміння, аби допомогти розробникам на кожному етапі створення API.

− Могутній механізм серіалізації, спільний як із осередками даних ORM і без.

− Під’єднується і нескладно налаштовуються валідатори та аутентифікатори.

− Класи контролерів для операцій CRUD.

– Регулювання відповідей HTTP, погодити тип вмісту завдяки заголовків протоколу HTTP Accept.

– Декомпозиція на сторінки полегшує процес повернення даних, таким способом, щоб потім можна поновити довільні типи носіїв.

– Видання метаданих спільно із наборами запитів.

DRF дає гнучко розширювати та регулювати інструменти фреймворка згідно з вимог розробника, що надзвичайно зменшує час розробки.

Poetry - це інструмент для управління залежностями в Python проєктах.

Функціональні можливості Poetry:

– Керування залежностями через toml файл.

– Автоматичне створення окремого віртуального середовища Python.

– Комфортне створення пакетів.

– poetry.lock файл для фіксації версій залежностей.

SQLite - це вбудована кросплатформова БД, котра підтримує досить великий набір команд SQL і доступ до коду. За замовчуванням підтримується Django.

Особливості SQLite:

– Транзакції атомарні, послідовні, ізольовані, і міцні.

– Установлювати за винятком налаштування.

– Реалізує неабияку частку стандарту SQL92.

– БД зберігається в одному файлі на диску.

Встановлення технологій на серверну сторону:

– Мові програмування Python (рисунок 3.12).

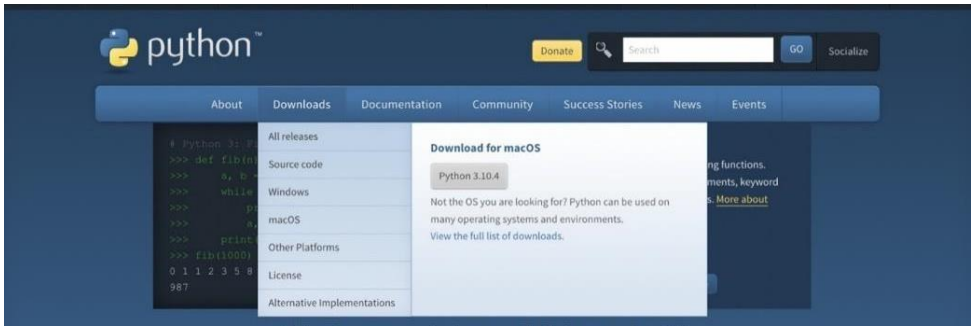

Рисунок 3.12 – Сторінка для завантаження Python

– Веб-фреймворк Django (рисунок 3.13).

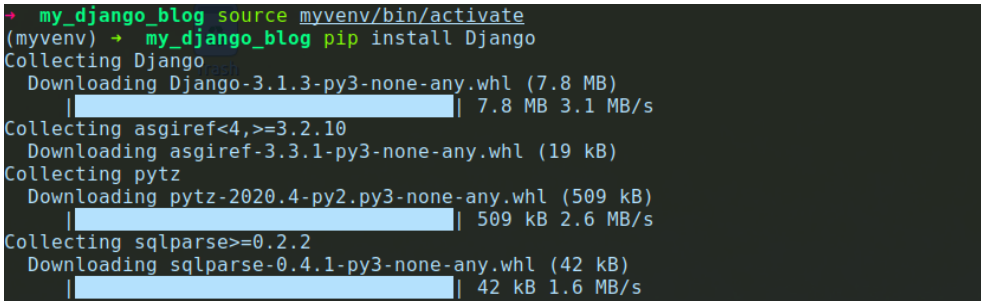

Рисунок 3.13 – Команда для встановлення Django

– Django Rest framework – це API фреймворк для Django (рисунок 3.14).

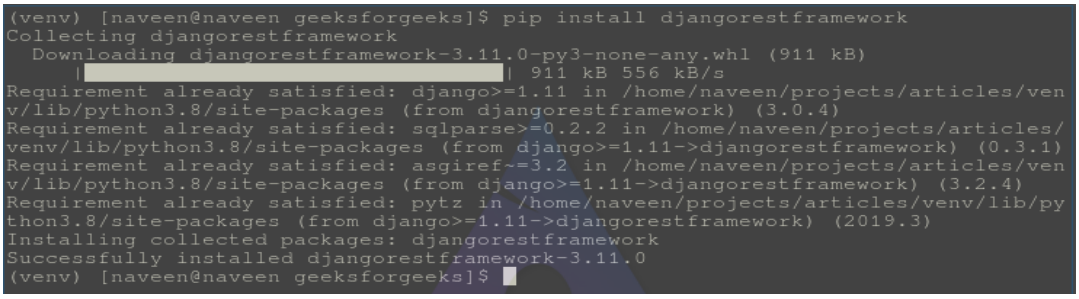

Рисунок 3.14 – Команда для встановлення Django Rest framework

– Django Ninja – це API фреймворк для Django (рисунок 3.15).

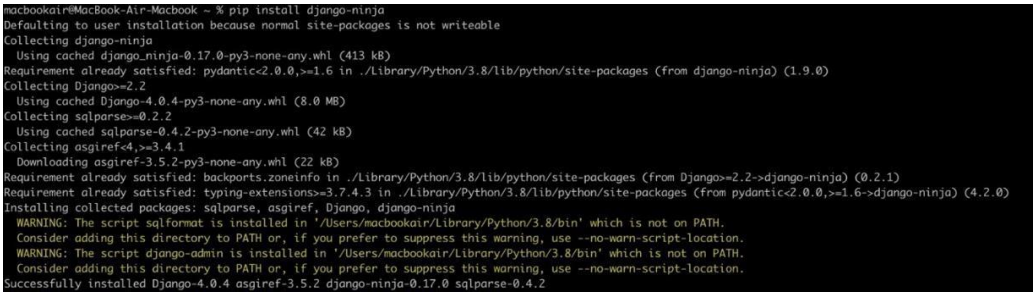

Рисунок 3.15 – Команда для встановлення Django Ninja

– Пакетний менеджер Poetry (рисунок 3.16).

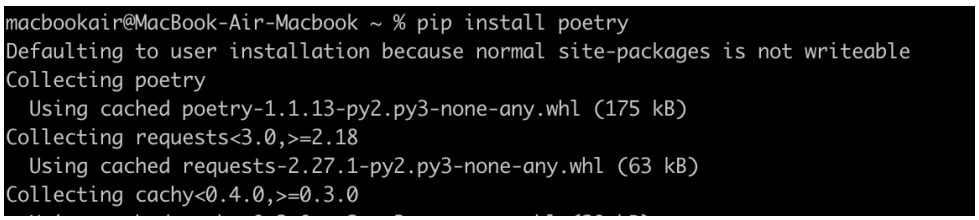

Рисунок 3.16 – Команда для встановлення Poetry

3.2 Реалізація системи

Кожній групі користувачів притаманний свій функціонал. Проте є і спільні функції. Основними з яких є авторизація та вихід з системи.

Авторизація – це ідентифікація користувача у системі та надання можливостей перегляду вмісту сайту. Авторизація системи реалізована за допомогою JWT токенів.

JSON веб-токен – це шаблон токена дозволу на базі JSON.

Після першого входу клієнт повертається згенерованим сервером JWT (рисунок 3.17). При кожному грядущомі запиті користувач має надавати JWT утвердженим методом API. Сервер декодує заголовок та корисне навантаження та, ймовірно, забронювані місця. Тому якщо усе в порядку, зазначеному в заголовку, алгоритм складається підпис. Якщо отримано підпис, переданий авторизованим користувачем.

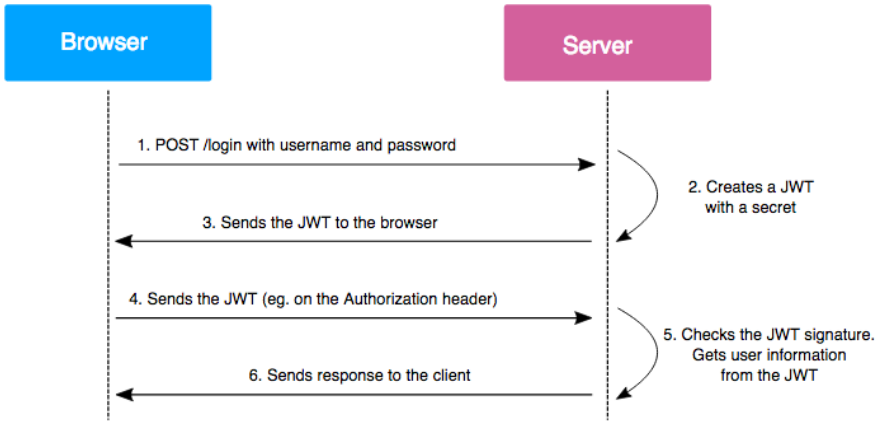

Рисунок 3.17 – Прицип роботи JWT токенів

На сервері авторизація реалізована завдяки бібліотеці Django rest framework simple jwt, яка дає змогу створювати та оновлювати jwt токени. Для того, щоб почати нею користуватись, потрібно встановити за допомогою команди pip install djangorestframework-simple-jwt. Після цього, слід зареєструвати її в файлі налаштувань та внести всі кінцеві точки у модуль із шляхами. Аби отримати токен, клієнт повинен надіслати логін і пароль на конкретний контролер, якщо він коректно пройде процес ідентифікації та

аутентифікації, йому повернуться два маркера. Оскільки токен доступу має певний період існування, для того аби його оновити існує другий підпис. Коли термін дії закінчився клієнт надсилає токен оновлення на певну кінцеву точку та отримує новий маркер доступу.

На клієнтській стороні, після авторизації, отримуємо токени, які зберігаємо локальному сховищі браузера. Аби реалізувати можливість надсилання запитів із маркером, у бібліотеці axios існують кілька варіантів реалізації. Перший варіант - це передавання підпису прямо у запит на сервер. Недоліком є те, що підпис необхідно прописувати у всіх запитах, що є не зовсім зручно при зміні метаданих запиту. Другий варіант – це створення об’єкта класу axios із всією його конфігурацією (лістинг 3.1).

Лістинг 3.1 – Конфігурація axios

```
const http = axios.create({
baseURL:baseUrl,
withCredentials: true,
paramsSerializer: params => Qs.stringify(params, {arrayFormat: 'repeat'})})
```

Цей варіант є більш оптимізованим. Аби реалізувати можливість оновлення токена, у axios є метод interceptors, котрий дозволяє отримати результат запиту і при потребі змінити його (лістинг 3.2).

Лістинг 3.2 – Налаштування Axios interceptor

```
http.interceptors.request.use((config) => {
    if (localStorage.getItem('access')) {
        config.headers.Authorization=`Bearer ${localStorage.getItem('access')}`
    }
    return config;
})
```

Першим типом користувачів у системі є адміністратор. За замовчуванням Django містить панель адміністрування, яку можна

конфігурувати для власних потреб. Першою сутністю з якою зіткнеться адміністратор - це користувачі. Для адміністратора реалізована можливість створення всіх користувачів системи: методистів, викладачів, студентів. Для нього сконфігуровано адміністративну модель: вивід всіх таблиць, показ вибраних колонок, фільтрація, пошук та сортування даних (лістинг 3.3).

Ліснинг 3.3 – Приклад конфігурації сутності у адміністративній панелі

```
@admin.register(Group)
class GroupAdmin(admin.ModelAdmin):
    list_display =
("id","name","educational_program")
    list_display_links=("id","name",
"educational_program")
    list_filter = (
        "educational_program__department",
    )
    @admin.display(description="Відділення")
    def department(self, obj):
        return f'{obj.department.name}'
```

Другим типом користувачів у системі є методист. Більшість функціоналу системи реалізовано для цього типу клієнтів. Перший з яких, це заповнення даних про навчання студентів: рік вступу, група, освітня програма (лістинг А1).

Після цього, студент потрапляє у відповідну таблицю. Там методист, має можливість відредагувати (лістинг В1) або видалити інформацію про студента (лістинг 3.4).

Лістинг 3.4 – Реалізація видалення студента

```
class RemoveData:
    def remove(self, request, *args, **kwargs):
        obj = self.get_obj()
        self.data_remove(obj)
```

Продовження лістингу 3.4

```
res = Response(data=data)
        return res
    def data_remove(self, obj):
  obj.remove()
```

Аналогічно реалізовий функціонал методиста для дій над предметами.

Окрім цих функцій, для методиста реалізовано можливість добавлення оцінок студентам вибраної групи та вибраного семестру (лістинг 3.5).

Лістинг 3.5 – Виставлення оцінок

```
@action(methods=['get'],                detail=False,
url_path='group/(?P<subject_id>\d+)')
    def group(self, *args, **kwargs):
        subject                                    =
models.Subject.objects.detail_subject(subject_id=kw
args.get('subject_id'))
        data=serializers.SubjectSerializer(subject,
context=self.get_serializer_context()).data
res = Response(data=data)
        return res
    def serializer_info(self):
        return {
            "action": self.action,
            "query_params":
self.request.query_params
        }
```

Як тільки оцінка була виставлена, методист має можливість оцінку відредагувати, змінивши потрібні поля на формі.

Після виставлення оцінок, методист має можливість переглянути сформовані системою рейтингові списки (лістинг 3.6). Може подивитись

список як із всією групою, так із тільки із студентами державної форми навчання.

Лістинг 3.6 – Формування рейтингового списку

```
@action(methods=['get'],                detail=True,
url_path='detail/(?P<semester>\d+)')
    def detail_group(self, *args, **kwargs):
bool_value=True  if
self.request.query_params.get('is_all_students') ==
"true" else False
        students=
models.Student.objects.rating_list(
            group_id=kwargs['pk'],
            semester=kwargs['semester'],
            is_all_students=bool_value
        )
dt=serializers.StudentInfoWithExtraPointsSerializer
(students, many=True).data
      return Response(data=dt)
```

Також методист може покращити, становище будь-якого студента у рейтинговому списку, добавивши йому додаткові бали за участь у позанавчальних заходах, олімпіадах, змаганнях тощо (лістинг 3.7). Для цього реалізована окрема сторінка із списком студентів та їх додатковими балами. Для того, щоб поставити додаткову оцінку потрібно заповнити поле з самою оцінкою та поле з описом оцінки.

Лістинг 3.7 – Створення додаткового балу

```
@action(methods=['post'], detail=False)
    def create_extra_points(self, *args, **kwargs):
        serializer=
self.get_serializer().do(data=self.request.data)
```

Продовження лістингу 3.7

```
            self.perform_create(serializer=serializer)
            return
    Response(data=self.get_serializer(serializer.instan
    ce).data)
```

Як і для звичайних оцінок, у додаткових балах передбачено редагування балу, а також видалення.

Третім типом користувачів у системі є студенти. Для них реалізований перегляд оцінок за всі семестри, всі річні оцінки та всі додаткові бали (лістинг 3.8).

Лістинг 3.8 – Вивід оцінок конкретних студентів

```
    @router(url_path='detail',
response=StudentRatingsResponseSchema)
    def ratings_student_by_semester(
            request, semester: int,
            student_id:int=None,
educational_program_id: int = None
    ):
        ratings = StudentRatingRepository(
            semester=semester,
            student_id=student_id or
request.auth.student.id,educational_program_id=educatio
nal_program_id   or
request.auth.student.group.educational_program.id
        )
        return ratings.build_data()
```

Вище перераховані функції є основними для кожної категорії користувачів.

3.3 Тестування системи

Перевірка веб-застосунку допомагає звірити наскільки функціональні можливості сходяться згідно з технічними задачами, стрімкість та незмінності праці, читабельність інформації, зручність переходів по сайту, захищеність від веб-атак і багато іншого.

Веб-застосунок необхідно звірити у повному об'ємі , насамперед вона буде працювати для клієнтів. Здійснюючи перевірку веб-застосунків, група може пересвідчитись, що веб-додаток виконує свої функції коректно із можливістю подальшого релізу для користувачів. Стиль та наявний функцій зовнішнього вигляду являються основними частинами перевірки веб-застосунка.

Інспектування переліку веб-перевірок:

- Тестування функціональних можливостей системи.
- Візуальне тестування.
- Тестування ефективності.
- Тестування захищеності.

Тестування функціональних можливостей зосереджується на бізнес-правил веб-застосунка. Головне завдання — це пересвідчитись, що повністю весь функціонал програмного забезпечення виконую свою роботу так, як відмічено в технічній задачі. У даній роботі було використано модульне та інтеграційне тестування.

Юніт, або модульна перевірка (рисунок 3.18) - це первинний етап перевірки, котрий переважно застосовується програмістами. Юніт перевірка дає можливість пересвідчитись, що автономні елементи програмного забезпечення працює на щаблі коду і чиниться передбачувано.

```python
@pytest.mark.parametrize(
    ("method", "name", "result"),
    [
        ("isdigit", "КН-31", "31"),
        ("isdigit", "!#-?", ""),
        ("isdigit", 321321, False),
        ("isalpha", "КН-31", "КН")
    ]
)
def test_symbols_by_methods(service, method, name, result):
    assert service._symbols_by_method(method, name) == result
```

Рисунок 3.18 – Приклад модульного тестування в системі

Інтеграційна перевірка (рисунок 3.19) - здійснює тестування, як достовірно виконують свою роботу автономні елементи програмного забезпечення, в час поєднуються. Потрібно виявити, як чинять відокремлені частини системи, в час роботи один з одним.

```python
def test_update_student(api, students, group, methodist):
    def convert_str_to_date(_date: str) -> date:
        return date(*map(int, _date.split('-')))

    student = students[0]
    data = {
        "year_entry": "2022-04-03",
        "group": group.id,
        "form_education": "Контрактна"
    }
    assert students[0].year_entry == convert_str_to_date(_date="2021-08-15")
    request = api.put(StudentUrl.STUDENT_BASE.value, data=data)
    request.user = methodist
    response = StudentApi.as_view({"put": "update"})(request, pk=student.id)
    student.refresh_from_db()
    assert response.status_code == 200
    assert StudentListSchema(**response.data)
    assert students[0].year_entry == convert_str_to_date(_date=data['year_entry'])
```

Рисунок 3.19 – Приклад інтеграційного тестування в системі

Візуальне тестування - це дія, за підтримки якої зміряються риси взаємозв'язку людини з автоматичним пристроєм і від находять неміцні сторони заради поправок.

Головними характеристиками є:

– Стрімкість загрузки веб-застосунку та його сторінок.

– Чутливість самих сторінок.

– Комфортний якість та шрифт змісту сторінки.

– Провідні опції легкі та сприйнятні та їх не треба віднаходити.

– Монотонність зовнішнього вигляду системи.

Завдання наведення візуальної перевірки:

– Покращити візитів на застосунок.

– Клієнт має можливість оперативно здійснювати потрібну роботу.

– Розробка поза свідомо доступа, клієнту надана можливість використовувати навички праці із іншими системами у його застосуванні.

Звідси було проведено юзабіліті тестування та опис оцінок для кожного з типів користувачів. Для користувача Адміністратор було проінспектовано сторінки виведення (рисунок 3.20) та детальний перегляд (рисунок 3.21) користувачів.

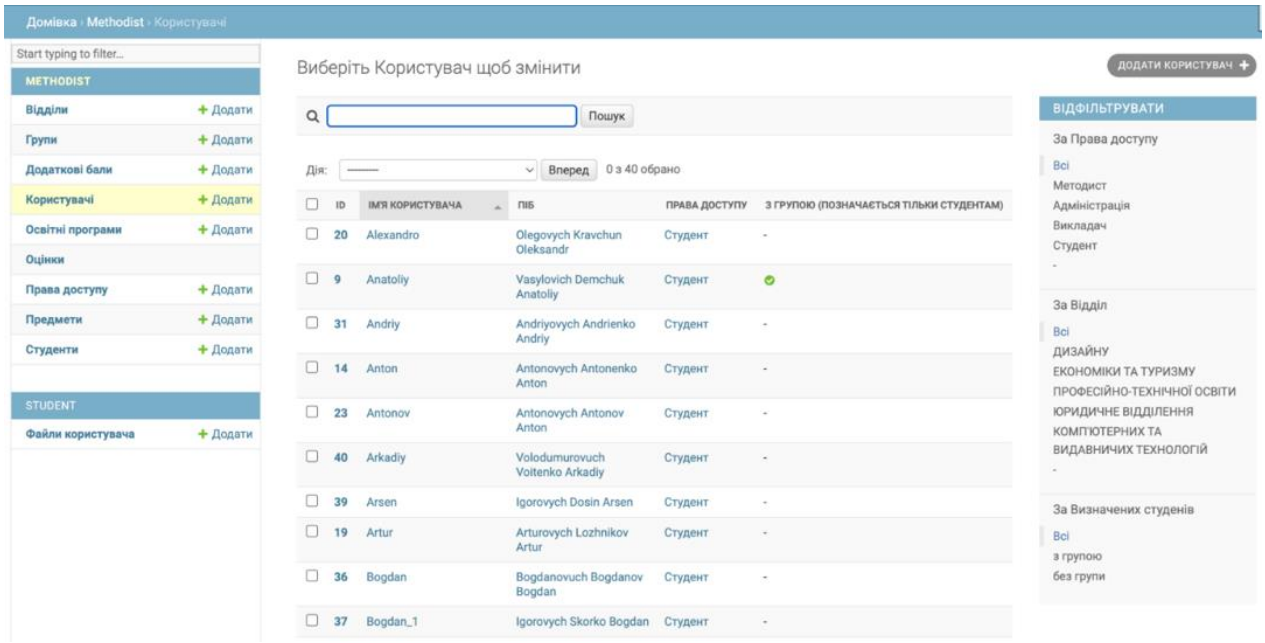

Рисунок 3.20 – Виведення користувачів

По центрі сторінки відображено сторінка з користувачами із можливістю пошуку, сортування, видалення. Для зручності було реалізовано ліву панель виведення всіх сутностей бази даних із можливістю їх фільтрацією. Права панель відповідає за фільтрацію даних у таблиці.

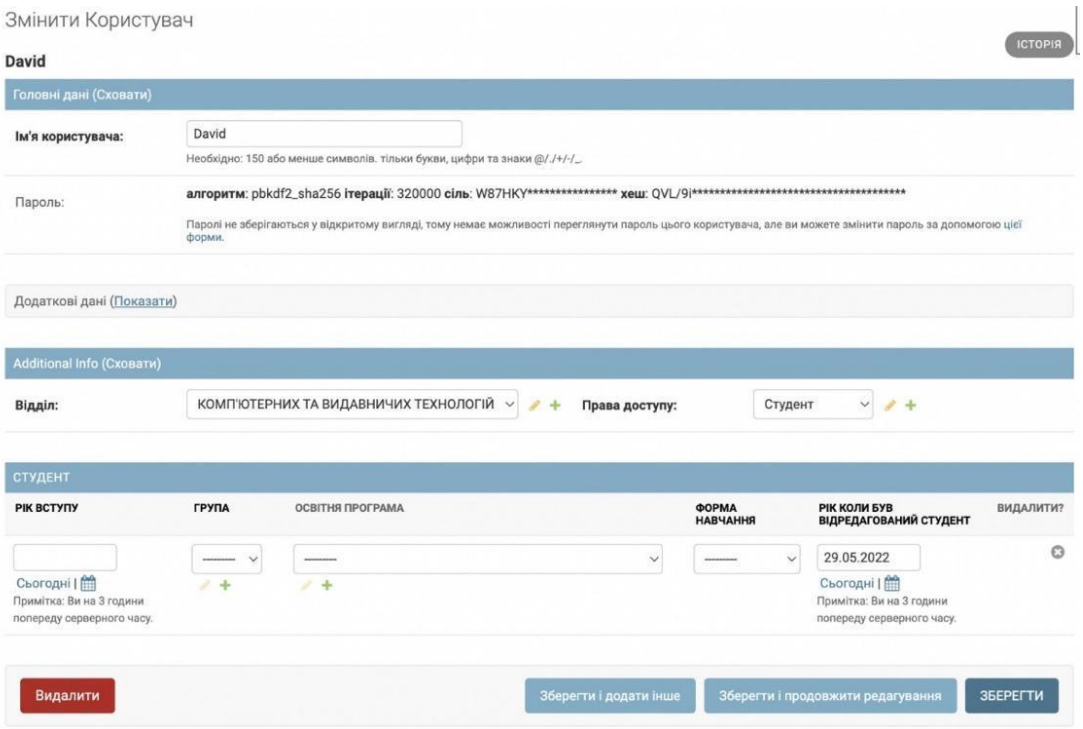

Рисунок 3.21 – Приклад детального перегляду користувача

При детальному перегляді користувача, адміністратор має можливість редагувати, переглядати та видаляти дані.

Для Методиста було досліджено сторінки предметів, студентів, оцінок, рейтингових списків.

Було створено меню з логотипом закладу та навігаційною панеллю. Для початку роботи, методист повинен створити студентів з користувачів, що знаходяться на сторінці Без групи (рисунок 3.22).

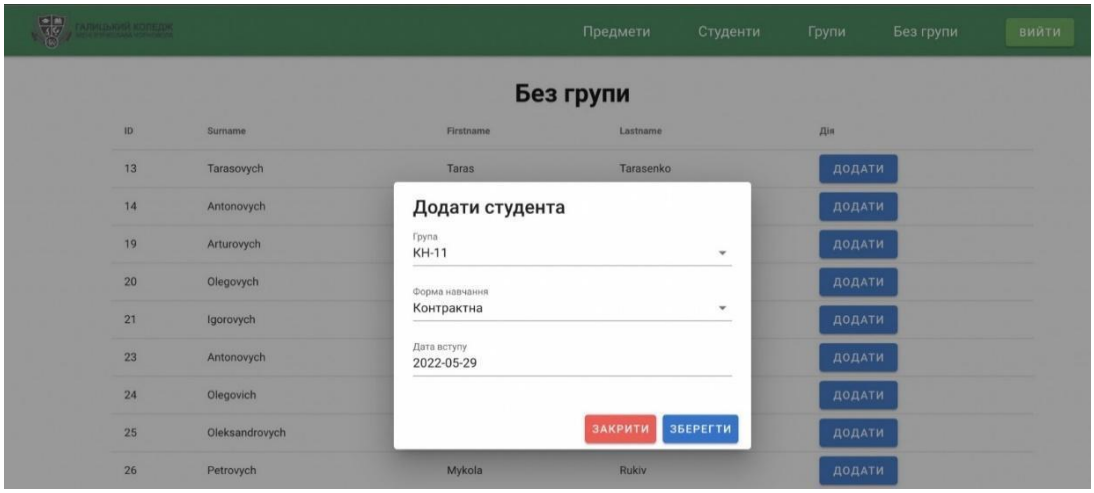

Рисунок 3.22 – Приклад створення студента

При переході на сторінку, відображається таблиця з користувачами. Щоб створити студента потрібно натиснути на кнопку Додати. Після чого відкривається модальне вікно з формою для заповнення. Поки не будуть заповнені всі поля, кнопка Зберегти буде неактивна.

Після створення студента, можна перейти на сторінку Студенти для подальших маніпуляції (рисунок 3.23).

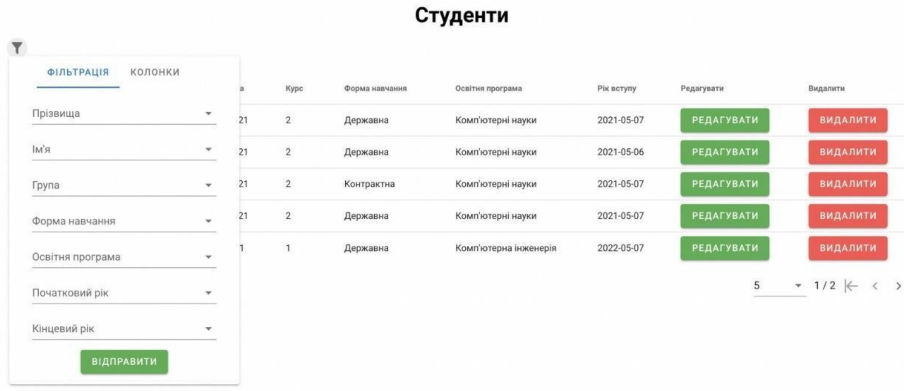

Рисунок 3.23 – Приклад створення студента

На сторінці відображено таблицю з створеними студентами та кнопка для фільтрації даних. Якщо потрібно відредагувати студента, потрібно натиснути на кнопку Редагувати, після чого відкривається модальне вікно з формою для редагування (рисунок 3.24). Валідація аналогічна попередній формі.

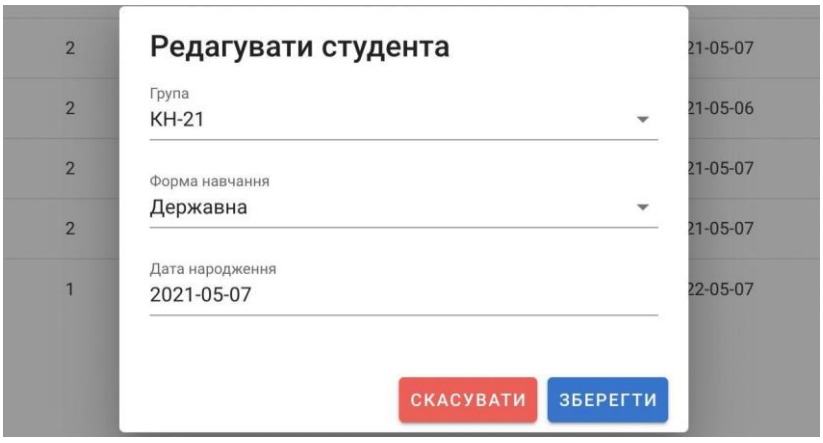

Рисунок 3.24 – Приклад редагування

Якщо потрібно видалити студента, потрібно натиснути на кнопку Видалити, після чого відкриється модальне вікно для підтвердження (рисунок 3.25).

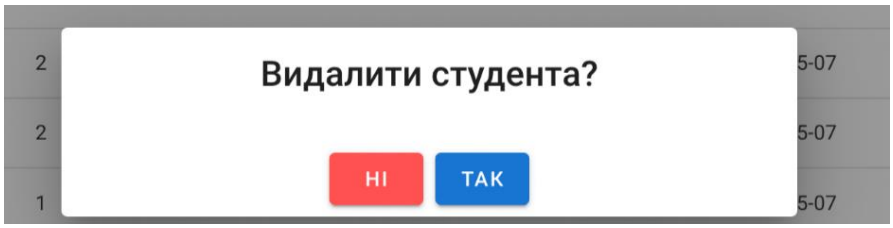

Рисунок 3.25 – Приклад видалення

Натиснувши на іконку фільтрації відкривається модальне вікно з двома вкладками: Фільтрація та Колонки. На вкладці Фільтрація є форма з полями для пошуку та фільтрації даних. На вкладці Колонки є поле з випадаючим списком, елементами якого є назви колонок таблиці, за допомогою якої можна відображати на сторінці тільки ті колонки, яку потрібні.

Аналогічний функціонал реалізовано із предметами.

Перейшовши на сторінки для виставлення оцінок, можна обрати семестр (рисунок 3.26).

Рисунок 3.26 – Обрання потрібного семестру для виставлення оцінок

Наступний кроком потрапляємо на сторінку виставлення оцінок студентам (рисунок 3.27).

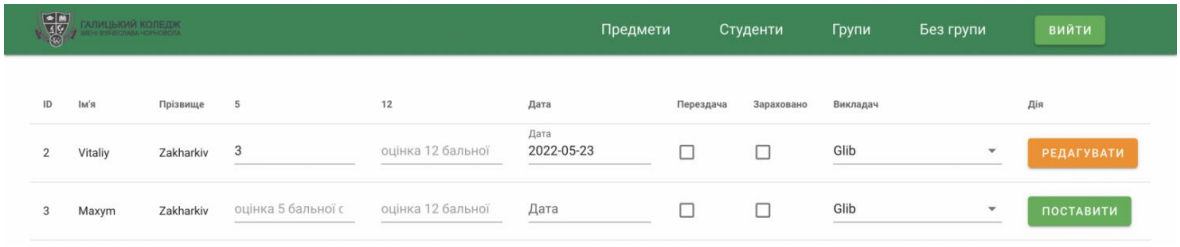

Рисунок 3.27 – Сторінка для виставлення оцінок

На ній відображена таблиця із студентами, де існує можливість заповнити дані та створити або відредагувати запис. Для зручності, після кожної дії будуть появлятися повідомлення про статус виконання операції.

Після того як в студента появились оцінки можна переглянути рейтинговий список на поточний момент (рисунок 3.28).

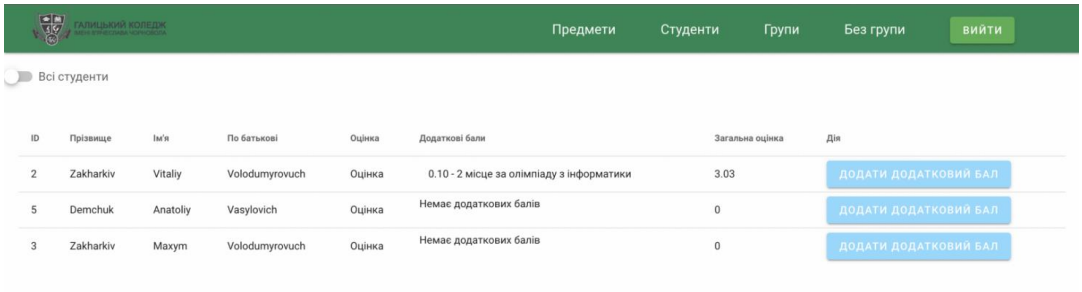

Рисунок 3.28 – Рейтинговий список

На сторінці присутня відсортована таблиця із студентами. За замовчування на ній відображається студенти державної форми навчання, але якщо активувати перемикач виведуться всі студенти.

Також є можливість добавити додатковий бал для студентів. Це можна зробити натиснувши на кнопку Додати Додатковий Бал, після чого відкривається модальне вікно з полем для вводу оцінки та полем для опису додаткового балу (рисунок 3.29).

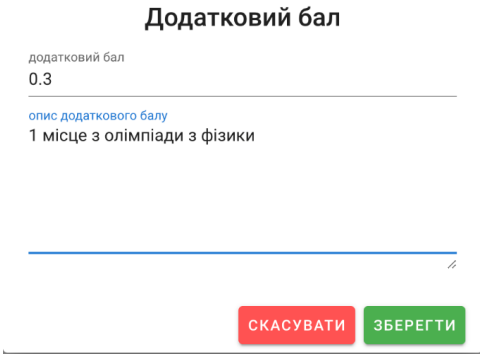

Рисунок 3.29 – Приклад виставлення додаткового балу

Окрім цього можна відредагувати або видалити додатковий бал (рисунок 3.30). Для цього потрібно натиснути на будь-який бал і вибрати потрібну дію.

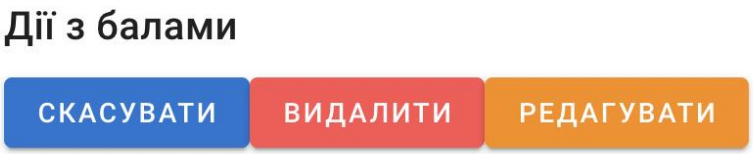

Рисунок 3.30 – Дії з додатковими балами

Тестування безпеки – ключ до надійності сайту.

Небезпека інформаційної загрози — це комплекс чинників, які утворюють загрозу.

Користувачі є найважливіші у всіх системах, тому заради них слід створювати комфортні та найголовніше безпечні системи. Першочерговим завданням є надійний захист облікових записів користувачів на веб-ресурсі. Тому для кожного користувача створюється особистий обліковий запис. Для ідентифікації якого повинні надаватися логін і пароль. Введений користувачем пароль небезпечно зберігати у відкритому вигляді в БД, тому що при взломі бази даних зловмисник отримує всі логіни та паролі користувачів. Для запобігання цього Django за замовчування, використовує хешування паролів (рисунок 3.31).

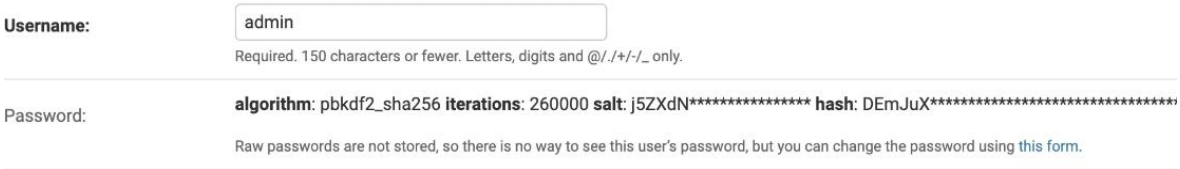

Рисунок 3.31 – Хешування паролів в Django

Крім цього потрібно перевірити чи має студент мати доступ до даних, які призначені методисту (лістинг 3.9). Для цього в Django існує модуль із класами котрі перевіряють права на ресурс в користувача.

Лістинг 3.9 – Перевірка права доступу

```
class BasePermission(BP):
    permission_name = None
    def has_permission(self, request, view=None):
```

Продовження лістингу 3.9

```
        req = getattr(request, 'auth')
            if not request.path.startswith('/student'):
                req = getattr(request, 'user')
            return bool(
                req and req.is_authenticated and
req.group.name == self.permission_name
            )
```

Наступною загрозою є ddos атаки. Зазвичай на сервері існує механізм throttling, така можливість є і у Django. За допомогою нього можна вказати кількість запитів на сервер, які він зможе прийняти.

Отже, у даному розділі було створено та проінспектовано серверну та клієнтську частини системи обліку студентів, створено модель бази даних, графічний інтерфейс веб-застосунка. Проаналізовано набір використаних технологій та продемонстровано їх встановлення.

4 ТЕХНІКО-ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ

4.1 Аналіз ринку

У теперішній час, поширеність веб-застосунків набуло великого значення у світовій мережі Інтернет. Прикладами таких застосунків є: інтернет-магазин сайт з новинами, інтернет банки, застосунки для обліку товарів та бухгалтерії тощо. Звідси, багато користувачів кожного дня користуються веб-застосунками.

З часом, розвиток веб-застосунків збільшується великими темпами і від звичайних новинних сайтів до автоматизованих систем обліку, які раніше велися виключно на папері. Це дає значне спрощення та підвищення ефективності під час роботи. Тому зараз неможливо уявити навчальні заклади, фірми, підприємства без подібних систем.

Розроблений веб-застосунок повинен здійснювати підтримку освітнього процесу студентів Галицького коледжу імені В’ячеслава Чорновола. Головними рисами створеного веб-застосунка є:

− Щоб використовувати систему необхідно будь-який технічний пристрій з будь-якою операційною системою.

− Зареєстрований, адміністратором системи, користувач після авторизації отримує перелік даних та функцій в залежності від свого типу.

− Простота системи у використанні, звідси не малий попит на неї та її аналогів.

− Створений простий та зрозумілий зовнішній вигляд системи тощо.

Паралельно розробці, було проведено аналіз ринку подібний систем автоматизації навчального процесу. З’ясовано, що для названого типу веб-додатків є безкоштовні аналоги. На них реалізований доволі примітивний функціонал, як для систем даного виду.

Окрім безкоштовних, є і приватні системи, котрі не доступні пересічним користувачам. Ці системи розробляються індивідуально під потреби та можливості навчального закладу. І застосовуються виключно у ньому.

Створений веб-додаток відноситься до другого варіанти і буде використовуватися виключно у Галицькому коледжі.

4.2 Розрахунок витрат на проєктування

Важливою структурою створеної системи, як і інших систем, є матеріальна частина. Це витрати на систему, матеріальну винагороду розробникам тощо. Об’єм матеріальної винагороди залежить від ефективності роботи, від складності виконання, від наслідків зробленої діяльності, можливостей фірми.

Матеріальна винагорода складається з основної та неосновної частини. Основна частина виплачується за реалізацію основної роботи, за визначеними правилами та часу. Вона фіксується для розробника при наймі на роботу або на проєкт. Неосновна виплачується за виконання певних додаткових завдань, які не стояли на першому порядку.

Грошова оплата повинна систематично оплачуватись за встановленими нормами закону.

Мінімальний оклад з 1 січня 2022 року в Україні становить 6500 гривень. Звідси, матеріальна винагорода розробників не може становити менше вище зазначеної суми.

Створення веб-застосунку розробили наступні працівники: проектувальник, бекенд-розробник, фронтенд-розробник, веб-дизайнер, тестувальник. Розмір окладу залежить від терміну, кількості, досвіду, ефективності роботи.

Проектувальник отримав за виконану роботу за місяць отримав 6500 гривень.

- Обчислення податку на прибутки фізособи: 6500 * 18% = 1170 гривень.
- Військовий податок: 6500 * 1,5% = 97,50 гривень.
- Єдиний внесок: 6500 * 22% = 1430 гривень.

- Утримання: 1170 + 97,50 = 1267,5 гривень.
- Розрахунок заробітньої платні: 6500 – 1267,5 = 5232,5 гривень.

Бекенд-розробник отримав за виконану роботу за місяць отримав 16000 гривень.

- Обчислення податку на прибутки фізособи: 16000 * 18% = 2880 гривень.
- Військовий податок: 16000 * 1,5% = 240 гривень.
- Єдиний внесок: 16000 * 22% = 3520 гривень.
- Утримання: 2880 + 240 = 3120 гривень.
- Розрахунок заробітньої платні: 16000 – 3120 = 12880 гривень.

Фронтенд-розробник отримав за виконану роботу за місяць отримав 14000 гривень.

- Обчислення податку на прибутки фізособи: 14000 * 18% = 2520 гривень.
- Військовий податок: 14000 * 1,5% = 210 гривень.
- Єдиний внесок: 14000 * 22% = 3080 гривень.
- Утримання: 2520 + 210 = 2730 гривень.
- Розрахунок заробітньої платні: 14000 – 2730 = 11270 гривень.

Тестувальник отримав за виконану роботу за місяць отримав 8000 гривень.

- Обчислення податку на прибутки фізособи: 8000 * 18% = 1440 гривень.
- Військовий податок: 8000 * 1,5% = 120 гривень.
- Єдиний внесок: 8000 * 22% = 1760 гривень.
- Утримання: 1440 + 120 = 1560 гривень.
- Розрахунок заробітньої платні: 8000 – 1560 = 6440 гривень.

Веб-дизайнер отримав за виконану роботу за місяць отримав 6500 гривень.

− Обчислення податку на прибутки фізособи: 6500 * 18% = 1170 гривень.

− Військовий податок: 6500 * 1,5% = 97,50 гривень.

− Єдиний внесок: 6500 * 22% = 1430 гривень.

− Утримання: 1170 + 97,50 = 1267,5 гривень.

− Розрахунок заробітньої платні: 6500 – 1267,5 = 5232,5 гривень.

Матеріальне винагорода розробників система зведена у таблиці 4.1.

Таблиця 4.1 – Обрахунок матеріальної винагороди

№	Посада	Оклад, Грн./міс	Відрахування, Грн./міс	Кількість		Сума, грн
1	Проектувальник	6500	1170	1 чол.	1 міс.	5330
2	Бекенд-розробник	16000	2880	1 чол.	5 міс.	65600
3	Фронтенд-розробник	14000	2520	1 чол.	5 міс.	57400
4	Веб-дизайнер	6500	1170	1 чол.	1 міс.	5330
5	Тестувальник	8000	1440	1 чол.	1 міс.	6560
		Усього зарплати:				140220

Вираховування на суспільні питання складає: 1430 + 3520 + 3080 + 1760 + 1430 = 11220 гривень.

Відрядження та контрагентські функції не проводилися, отже і розходів відповідно не було.

Інших прямих розходів також не було.

Взагалом прямих розходів складає 11220 гривень.

Накладні витрати за один місяць дорівнює 40% від загальної суми прямих витрат – 11220 * 40% = 4488 гривень.

Передбачуване зібрання дорівнють 25% від суми прямих та накладних витрат – (11220 + 4488)*25% = 3927 гривень.

Кошторисна ціна додатка: 4488 + 11220 + 3927 = 19635 гривень.

Ціна за договором: 3927 + 19635 = 23562 гривень.

Податок на додаткову ціну: 19635*20% = 3927 гривень.

Кошторис розходів у проектуванні зведено у таблиці 4.2.

Таблиця 4.2 - Кошторис розходів у проектуванні

Назва статей витрат	Сума, грн	Обгрунтування
1. Зарплата розробників	140220	
2. Відрахування на соціальні питання	11220	
3. Контрагентські функції	-	Не відбулися
4. Відрядження	-	Не відбулися
5. Інші прямі витрати	-	
6. Усього прямих витрат	11220	
7. Накладні витрати	4488	
8. Передбачуване зібрання	3927	
9. Загальна вартість проекту	19635	
10. Податок на додаткову вартість	3927	
11. Загалом, ціна за договором	23562	

4.3 Обгрунтування необхідності

Всі проєкти, котрі є у розробці, створюються задля принесення повного результат. Результатом буде вважатися автоматизація певних процесів, які позбавлятимуть від непотрібних рутинних завдань та допомагають зосередитись на більш важливих речах.

Коли навчальний заклад думає про власну систему, зазвичай іде процес серйозного проектування. Здебільшого планування створення даних застосунків іде таким чином, щоб потребувало внесення кардинальних змін у майбутньому.

Визначальним обґрунтуванням для створення подібного сайту є зручність та доступне введення успішності. Усім користувачам буде комфортно користуватись додатком, оскільки він буде отримувати тільки ті дані, котрі хотів бачити. Окрім цього, студент матиме можливість переглядати свої оцінки та приймати участь у формуванні стипендіального рейтингу.

З огляду на вищеперелічені обгрунтування грамотно спроектована та реалізована система разом із мінімальною її підтримкою, принесе користь для навчально закладу та позбавить його від багатьох не вирішених проблем.

ВИСНОВКИ

Під час виконання дипломного проєкту було спроєктовано, реалізовано та протестовано інформаційну систему підтримки освітнього процесу студентів. Реалізація почалась із пошуку та порівняння аналогів, виявлення плюсів та мінусів існуючих рішень. Після чого була поставлена мета та цілі ресурсу. Здійснено дослідження можливостей та способів реалізації програмної системи. Аргументовано вибір утворення додатку, оснований на веб-технологіях, а також збудований завдяки монолітній архітектурі та відомих патернів програмування. Це дає спроможність покращити гнучкість та комфортність системи, як в розробці та підтримці, так і у користуванні.

Під час розробки було спроектовано і сформовано БД, серверну частину та візуальну частину. Для створення системи було використано наступний інструментал: Python, Django, JavaScript, Vue.js, Vuex, Vuetify.

Використані технології продемонстрували, що раціональний вибір інструментів для розробки проєкту може покращити хід розробки і зробити його привабливим.

В підсумку дипломної роботи було одержано працездатну веб-систему, котрою можна користуватись задля підтримки освітнього процесу студентів. Даний сайт дає змогу адміністратору контролювати всі дії в системі, методисту фіксувати дані та інформацію про студентів, а сам він переглядати та досліджувати свій прогрес протягом усього навчання.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. WEB-СЕРВЕРИ: веб-сайт. URL: https://stud.com.ua/97608/info/serveri (дата звернення 10.01.2022).

2. Підручник Django – MVT Архітектура, команди користувача: веб-сайт. URL: https://pythobyte.com/django-tutorial-mvt-architecture/ (дата звернення 10.01.2022).

3. Розділ 2. Основи UML: веб-сайт.URL: https://docs.org/trunk5/uml (дата звернення 13.01.2022).

4. Програмне забезпечення та його класифікація: веб-сайт. URL: https://kppk.com.ua/ELLIB/Gorbenko/IKT/ (дата звернення 20.02.2022).

5. Посібник Pycharm : веб-сайт. URL: https://www.jetbrains.com/ua/ (дата звернення 24.02.2022).

6. Що таке база даних: веб-сайт. URL: http://apeps.kpi.ua/shco-basa-danykh (дата звернення 01.03.2022).

7. Що таке frontend-розробка: веб-сайт. URL: https://te.it.org/blog/frontend (дата звернення 06.03.2022).

8. Що таке backend-розробка: веб-сайт. URL: https://te.it.org/blog/backend (дата звернення 07.03.2022).

9. Що таке нереляційна база даних: веб-сайт. URL: ttps://uk.theastrolo.com (дата звернення 10.03.2022).

10. The Progressive JS Framework: веб-сайт. URL: https://vuejs.org/compon/ (дата звернення 10.03.2022).

11. What is a "State Management Pattern": веб-сайт. URL : http://vuex.org/st/ (дата звернення 23.03.2022).

12. Що таке Python: веб-сайт. URL: https://python.org/docs/(дата звернення 27.04.2022).

ДОДАТКИ

Додаток А

Програмний код

Лістинг А1 – Створення студентів

```
saveDate(){
          const dataForm = {
            group: this.formGroup,
            form_education: this.formEduction,
            year_entry: this.date
          }
          this.loading = true

http.post(`/methodist/student/create/${this.id_user}/`,
dataForm)
              .then(response => {
                console.log(response)
                this.listStudent=
this.listStudent.filter((p) => p.id !== this.id_user)
                this.loading = false
                const    info    =    {'text':'Студента
створено', 'color':'green'}
                this.actionOpenSnack(info)
              })
              .catch(error => {
                console.log(error)
                alert('Error')
              })
          this.formEduction = ''
        this.formGroup = ''
```

Продовження лістингу А1

```
        this.date = ''
        this.dialog = false
        },
```

Додаток В

Програмний код

Лістинг В1 – Редагування студентів

```
changeStudent(){
      const dataChange = {
        group: this.activeData.group,
        form_education: this.activeData.form_education,
        year_entry: this.activeData.year_entry
      }
      this.loading = true

http.put(`/methodist/student/${this.activeData.id}/`,
dataChange)
        .then(response => {
          let find = this.listStudent.find(i => {
            return i.id === this.activeData.id
          })
          let index = this.listStudent.indexOf(find)
          this.listStudent[index].id = response.data.id
          this.listStudent[index].group =
response.data.group
          this.listStudent[index].educational_program =
response.data.educational_program
          this.listStudent[index].form_education =
response.data.form_education
```

Продовження лістингу В1

```
  this.listStudent[index].year_entry =
response.data.year_entry
          this.loading = false
  this.dialogChange = false
          const info = {'text':'Студент успішно
відредагований', 'color':'orange'}
          this.actionOpenSnack(info)
        })
        .catch(error => {
          console.log(error)
        })
    },
```

Додаток С

Програмний код

Лістинг С1 – Навігаційне меню

```
<nav>
    <v-toolbar elevation="2" style="background:
#198754">
      <v-container style="max-width: 1460px; display:
flex">
      <v-app-bar-nav-icon>
        <v-img src="../../assets/logoGC.png"
style="margin-left: 150px" width="180px"></v-img>
      </v-app-bar-nav-icon>
      <v-spacer></v-spacer>
      <div v-if="auth === 'Методист'" class="mt-2">
        <router-link to="/subjects"
class="router">Предмети</router-link>
```

Продовження лістингу С1

```
 <router-link to="/students"
class="router">Студенти</router-link>
        <router-link to="/group"
class="router">Групи</router-link>
        <router-link to="/without-group"
class="router">Без групи</router-link>
    <router-link to="/" class="router">
    <v-btn class="success" @click="logout">
    Вийти
          </v-btn>
        </router-link>
      </div>
      <div v-else-if="auth === 'Студент'">
        <router-link to="/asd">
          Студент
        </router-link>
        <router-link to="/" class="router">
          <v-btn class="success" @click="logout">
            Вийти
          </v-btn>
        </router-link>
      </div>
      </v-container>
    </v-toolbar>
  </nav>
```

Лістинг С2 – Форма авторизації

```
<v-form>
    <v-row>
      <v-col cols="12">
```

Продовження лістингу С2

```
<v-text-field
            prepend-icon="mdi-account"
            hide-details
            placeholder="Логін"
            v-model="username"
            color="green"
            class="text--white"
></v-text-field>
      </v-col>
      <v-col cols="12">
        <v-text-field
            prepend-icon="mdi-lock"
            class="input"
            placeholder="Пароль"
            hide-details
            color="green"
            v-model="password"
            :append-icon="show1 ? 'mdi-eye' : 'mdi-eye-
off'"
            @click:append="show1 = !show1"
            :type="show1 ? 'text' : 'password'"
        ></v-text-field>
      </v-col>
      <v-col align="center">
        <v-btn class="success" @click="login">
          Увійти
        </v-btn>
      </v-col>
```