ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ

Навчально-науковий інститут денної освіти
Форма навчання денна
Кафедра комп'ютерних наук та інформаційних технологій

Допускається до захисту
Завідувач кафедри
_____ Олена ОЛЬХОВСЬКА
(підпис)

«___»_____202_ р.

КВАЛІФІКАЦІЙНА РОБОТА

на тему
«Розробка підсистеми Рейтингування викладачів сервісу «Електронний деканат»

зі спеціальності 122 Комп'ютерні науки
освітня програма «Комп'ютерні науки»
ступеня магістра

Виконавець роботи Пісковий Дмитро Русланович
_____«____»_____ 202_ р.
(підпис)

Науковий керівник к.ф.-м.н., доц., Ольховська Олена Володимирівна
_____«____»_____ 202_ р.
(підпис)

Рецензент _____

ПОЛТАВА 2023

ЗАТВЕРДЖУЮ
Завідувач кафедри _____ Олена ОЛЬХОВСЬКА
«___» січня 2023 р.

ЗАВДАННЯ І КАЛЕНДАРНИЙ ГРАФІК
ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

на тему «Розробка підсистеми Рейтингування викладачів сервісу «Електронний деканат»
зі спеціальності 122 Комп'ютерні науки»
освітня програма «Комп'ютерні науки»
ступеня магістр
Прізвище, ім'я, по батькові Пісковий Дмитро Русланович
Затверджена наказом ректора № ___-Н від «__» _____ 202__ р.
Термін подання студентом роботи «__» _____ 202__ р.
Вихідні дані до кваліфікаційної роботи: публікації з теми, підсистема рейтингування викладачів
Зміст пояснювальної записки (перелік питань, які потрібно розробити)

ВСТУП
РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ
РОЗДІЛ 2. ІНФОРМАЦІЙНИЙ ОГЛЯД
2.1 Поняття веб-сторінки
2.2 Поняття інтернет-порталу
2.3 Поняття та історія мережі «Інтернет»
2.4 Середовища та мови програмування, середовища для розробки програмного забезпечення
РОЗДІЛ 3. ТЕОРЕТИЧНА ЧАСТИНА
3.1. Огляд матеріалу з теми
3.2. Алгоритм роботи підсистеми
3.3. Обґрунтування вибору програмних засобів для реалізації завдання роботи
РОЗДІЛ 4. ПРАКТИЧНА ЧАСТИНА
4.1 Опис програмної реалізації
4.2 Інструкція по роботі з програмою
ВИСНОВОК
СПИСОК ЛІТЕРАТУРИ
ДОДАТОК А. ПРОГРАМНИЙ КОД

_____

Консультанти розділів кваліфікаційної роботи

Розділ	ПІП, посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Постановка задачі	Ольховська О.В.		
Інформаційний огляд	Ольховська О.В.		
Теоретична частина	Ольховська О.В.		
Практична частина	Ольховська О.В.		

Календарний графік виконання кваліфікаційної роботи

Зміст роботи	Термін виконання	Фактичне виконання
1. Вступ		
2. Вивчення методичних рекомендацій і стандартів та звіт керівнику		
3. Постановка задачі		
4. Інформаційний огляд джерел бібліотек та інтернету		
5. Теоретична частина		
6. Практична частина		
7. Закінчення оформлення		
8. Доповідь студента на кафедрі		
9. Доопрацювання (за необхідності), рецензування		

Дата видачі завдання «__» _____ 202_р.

Здобувач вищої освіти Дмитро ПІСКОВИЙ

Науковий керівник к. ф.-м. н. _____

Результати захисту кваліфікаційної роботи

Кваліфікаційна робота оцінена на _____

(балів, оцінка за національною шкалою, оцінка за ECTS)

Протокол засідання ЕК №____ від «____»_____ 202_ р.

Секретар ЕК _____ _____

(підпис) *(ініціал та прізвище)*

Затверджую
Зав. кафедрою _____
к.ф.-м.н. Олена Ольховська
«____» _____ 202__ р.

Погоджено
Науковий керівник _____
к.ф.-м.н. Олена ОЛЬХОВСЬКА
«____» _____ 202__ р.

План

кваліфікаційної роботи ступеня магістр
зі спеціальності 122 Комп'ютерні науки
освітня програма 122 Комп'ютерні науки
Пісковий Дмитро Русланович
Прізвище, ім'я, по батькові

на тему «Розробка підсистеми Рейтингування викладачів сервісу «Електронний деканат»

ВСТУП
РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ
РОЗДІЛ 2. ІНФОРМАЦІЙНИЙ ОГЛЯД
2.1 Поняття веб-сторінки
2.2 Поняття інтернет-порталу
2.3 Поняття та історія мережі «Інтернет»
2.4 Середовища та мови програмування, середовища для розробки програмного забезпечення
РОЗДІЛ 3. ТЕОРЕТИЧНА ЧАСТИНА
3.1. Огляд матеріалу з теми
3.2. Алгоритм роботи підсистеми
3.3. Обґрунтування вибору програмних засобів для реалізації завдання роботи
РОЗДІЛ 3. ПРАКТИЧНА ЧАСТИНА
4.1 Опис програмної реалізації
4.2 Інструкція по роботі з програмою
ВИСНОВОК

Здобувач вищої освіти _____ Дмитро ПІСКОВИЙ
«____» _____ 202__ р.

АНОТАЦІЯ

Записка: 46 с., 10 рис., 6 джерел.

Мета роботи – Алгоритмізація та програмування тренажера з теми «Числові послідовності" дистанційного навчального курсу «Математичний аналіз».

Об’єкт розробки – розробка підсистеми Рейтингування викладачів сервісу «Електронний деканат», веб-сайт.

Методи дослідження та інформаційне забезпечення - методи розробки WEB застосунків, середовище розробки PhpStorm та мова програмування PHP.

Результати дослідження. Проведено огляд програмного забезпечення створеного студентами минулих років для керування процесами деканату. Виділено позитивні та негативні аспекти. Розроблено алгоритм підсистеми Рейтингування викладачів. Здійснена програмна реалізація в середовищі PhpStorm за допомогою мови програмування PHP.

Рекомендації щодо використання результатів дослідження. Програмний продукт, що реалізує підсистему Рейтингування викладачів сервісу впроваджено в освітній процес ПУЕТ.

Наукова апробація. За матеріалами роботи опубліковано тези (науково-практичний семінар «Комп’ютерні науки та інформаційні технології» (КНІТ-2023).

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

Умовні позначення, символи, скорочення, терміни	Пояснення умовного позначення, скорочень, символів
ДК	Дистанційний курс.
ДН	Дистанційне навчання.
PHP	Мова програмування.
PhpStorm	Середовище для написання програмного коду.

ВСТУП

Проблему впровадження інформаційних технологій в навчальний процес досліджують у великій кількості робіт. В останні роки створено непогані варіанти програмних рішень широкого кола застосування проте вони не узгоджені один з одним за великою кількістю параметрів, відрізняються операційними системами, функціональністю, їх зміст не дозволяє їх уніфікувати.

Сучасні інформаційні та комп'ютерні технології, в сучасності широко використовуються для організації навчання і суміжних процесів.

Водночас комп'ютерні технології дають змогу не тільки зменшити вартість реалізації низки процесів, але й покращити їх якість. Адже відомо, що інформаційні технології навчання сприяють підвищенню продуктивності, якості наданих послуг а ще й доволі часто їх автоматизації що дозволяє уникати необхідної участі кінцевого користувача.

Вони передбачають (крім передачі інформації і закладених у ній знань):

- інтелектуальний розвиток (конструктивне, алгоритмічне мислення, завдяки особливостям спілкування з комп'ютером;
- креативний розвиток (творче мислення) особистості за рахунок зменшення частки репродуктивної діяльності;
- професійний розвиток (формування уміння приймати оптимальні рішення у складних ситуаціях під час буденного виконання існуючих процесів.

Мета кваліфікаційної роботи – розробка підсистеми Рейтингування викладачів сервісу «Електронний деканат».

Об'єктом розробки є процес програмної реалізації підсистеми Рейтингування викладачів сервісу «Електронний деканат».

Предмет розробки — процес програмної реалізації заданої підсистеми.

РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ

Рішення будь-якої прикладної задачі з використанням електронних обчислювальних машин (ЕОМ) складається з наступних етапів:

- аналіз вимог і формальна постановка задачі;
- вибір або розробка математичної моделі;
- аналіз способів вирішення;
- логічне проектування і розробка алгоритму;
- кодування (написання програми);
- тестування і налагодження програмного забезпечення;
- впровадження, використання і супровід програмного забезпечення.

Постановка завдання розробки програмного забезпечення є найпершим і найбільш важливим етапом при проектуванні програмного забезпечення. Саме тут закладається фундамент майбутньої програми. Якщо на цьому етапі допущені помилки або не передбачені якісь важливі деталі, то це відіб'ється на кінцевому результаті, а вносити зміни і виправлення в такий проект буде вкрай проблематично. Більш того, якщо завдання поставлене невірно, то її подальше рішення не має сенсу. На етапі постановки завдання необхідно відповісти на наступні питання:

- чи існують методи або способи вирішення завдання без використання ЕОМ, чи потрібно вирішувати це завдання на ЕОМ;
- які «дивіденди» принесе використання ЕОМ для вирішення завдання, що буде покращено, прискорене, оптимізовано, зекономлено при використанні ЕОМ, яка основна мета застосування ЕОМ;
- що необхідно з обладнання, крім універсальної ЕОМ, який тип ЕОМ необхідний для вирішення завдання, яка платформа та ЕОМ може підійти, яка операційна система буде керувати роботою ЕОМ, яке додаткове програмне забезпечення потрібно;

- які бізнес-процеси і документообіг прикладної предметної області, де буде застосовуватися розробляється програмне забезпечення;
- який формат вихідних (вхідних) даних, які дані і в якій формі необхідні для вирішення завдання;
- який формат проміжних і вихідних даних, які дані і в якій формі необхідно отримати;
- який інтерфейс користувача програмного забезпечення потрібно забезпечити, який інтерфейс з додатковим обладнанням необхідний;
- яка «глибина» опрацювання користувальницької, інженерної, програмної і конструкторської документації, експлуатаційних документів, методичних посібників і керівництв по використанню програми, хто і як буде застосовувати результати рішення.

Перший крок у проектуванні програмного забезпечення полягає в точному формулюванні цілей впровадження програми. Необхідно, щоб цей процес був гнучким та організованим і працював протягом тривалого часу, оскільки на будь-якому етапі розробки або впровадження можуть розкриватися раніше непередбачені проблеми, які потребують внесення в проект певних змін. Необхідно також заздалегідь визначити умови внесення несуттєвих змін в проект. Всі домовленості такого плану повинні оформлятися розробником і замовником програми в офіційному порядку у вигляді окремих пунктів у договорі на розробку програмного забезпечення, додаткових протоколів чи актів.

З офіційним висновком договору зазвичай передають:

- з'ясування реальної необхідності такої системи;
- оцінка можливості її розробки і зразкового обсягу витрат;
- визначення очікуваного ефекту від впровадження.

Після завершення етапу попередніх досліджень складають список вимог, що пред'являються до програмного забезпечення. Сюди входять:

- аналіз обстановки (сукупність умов, в яких передбачається експлуатувати програмну систему);
- опис виконуваних програмою системою функцій (чіткий опис того, що повинна робити система, на підставі яких вхідних даних, які дані є вихідними);
- обмеження, які повинні враховуватися в процесі проектування (терміни завершення; ресурси, наявні в наявності).

Головна мета на етапі аналізу формальної постановки задачі і складання вимог до програми полягає в пошуку і поданні таких ситуацій, які можуть привести до збою в роботі програми і у визначенні причин і способів подолання таких ситуацій.

РОЗДІЛ 2. ІНФОРМАЦІЙНИЙ ОГЛЯД

2.1. Поняття веб-сторінки

Веб-сторінка - це набір текстових файлів, які містять теги HTML. Ці файли під час завантаження відвідувачем на його комп'ютер, обробляються браузер тож користувач бачить не вміст файлу, а результат його інтерпретації. Мова HTML дозволяє форматувати текст та розрізняти у ньому функціональні елементи, створювати гіперпосилання, прикріплювати зображення або інші мультимедійні елементи. Відображення сторінки можна змінити додаванням стилів CSS або сценаріїв JavaScript. Сторінки сайтів можуть бути простим статичним набором файлів або містити код, який виконується у спеціальному середовищі на сервері. Воно може бути розробленим на замовлення для окремого сайту, або бути готовим продуктом, розрахованим на певний клас сайтів.

Деякі з них можуть забезпечити власнику сайту можливість гнучкого налаштування структуризації і відображення інформації на веб-сайті. Такі керуючі програми називають системами управління змістом (CMS).

Процес створення HTML-файлу за затвердженим дизайном називають версткою.

Верстка передбачає створення розмітки веб-сторінки, налаштування стилів, підготовку графічних елементів дизайну для інтеграції їх в шаблон та чимало інших завдань.

Тобто, веб-сайт (англ. website, від web – павутина, site – місце) – це сукупність логічно пов'язаної гіпертекстової інформації, оформленої у вигляді окремих сторінок, що утворюють певний інформаційний ресурс (комерційний, бібліотеки, фотогалереї, або всі перелічені одночасно в одному місці, доступний за унікальною адресою (URL))

2.2. Поняття інтернет-порталу

Інтернет-портал (веб-портал, інформаційний портал) — сукупність поєднаних безпосередньо та через мережу інтернет апаратних засобів, що включають комп'ютери та електронні носії інформації з заздалегідь записаною на них інформацією та виконані з можливістю запису та зчитування інформації у вигляді комп'ютерних програм, баз даних. З можливістю обробки зазначеної інформації та команд користувача системи та надання йому Інтернет-сервісів як результатів обробки відповідної інформації і команд.

В залежності від спеціалізації, портали поділяються на горизонтальні (охоплюють велику кількість тем і сервісів), вертикальні (мають вузьку тематичну спрямованість, орієнтовану на повне охоплення інформації з певної окремої галузі) та змішані (поєднують у собі функції електронної торгівлі та довідкові сервіси)

Саме тому вертикальний портал єнайсучаснішою формою структурування освітньої інформації, оскільки портали є багатокомпонентними, їх користувачами є різні категорії споживачів з різним рівнем освітніх запитів, яким портал надає можливість для вільного виходу у відкритий інформаційно-освітній простір за цільовим призначенням.

2.3. Поняття та історія мережі «Інтернет»

Інтернет — всесвітня система сполучених комп'ютерних мереж, що базуються на комплекті Інтернет-протоколів.

Інтернет також називають мережею мереж, бо він складається з мільйонів локальних і глобальних приватних, публічних, академічних, ділових і урядових мереж, пов'язаних між собою з використанням різноманітних дротових, оптичних і бездротових технологій.

Міжмережжя становить фізичну основу для розміщення величезної кількості інформаційних ресурсів і послуг, таких як взаємопов'язані гіпертекстові документи Всесвітнього павутиння (World Wide Web — WWW) та електронна пошта.

Історія інтернету починається з 1962 року.

Саме тоді Джозеф Ліклайдер (1915–1990), керівник Агентства передових оборонних дослідницьких проєктів США висловив ідею Всесвітньої комп'ютерної мережі.

1969 року Міністерство оборони США започаткувало розробку проєкту, що мав на меті створення надійної системи передачі інформації на випадок війни.

Розробка була доручена Каліфорнійському університету Лос-Анджелеса, Стенфордському дослідному центрові, Університету штату Юта і Університету штату Каліфорнія в Санта-Барбарі.

Цю мережу було названо ARPANET (Мережа Агентства передових досліджень). У рамках проєкту мережа об'єднала названі заклади. ARPANET почала активно рости й розвиватись, її дедалі ширше почали використовувати вчені, які працювали у різних галузях науки.

До 1971 року було створено першу програму для надсилання електронної пошти мережею, котра одразу ж стала дуже популярною.

На початку 1970-х років мережа об'єднувала близько 200 вузлів.

1973 року до мережі через трансатлантичний кабель були підключені перші іноземні організації з Великої Британії і Норвегії — мережа стала міжнародною.

Наприкінці 1970-х років почали активно розвиватись протоколи передачі даних, що були стандартизовані у 1982–1983 роках.

1 січня 1983 року мережа ARPANET перейшла з протоколу NCP на протокол TCP/IP, який досі успішно використовується для об'єднання мереж. Саме 1983 року за мережею ARPANET закріпився термін «Інтернет».

Комерціалізація в 1990-х міжнародної мережі привела до її популяризації та впровадження в практично кожен аспект сучасного життя людини.

1984 року була розроблена система доменних назв (DNS). Тоді ж у мережі ARPANET з'явився серйозний суперник — NSFNet – мережа, яка була сформована з дрібніших мереж, і мала значно більшу пропускну здатність, аніж

ARPANET. До цієї мережі за рік під'єдналось близько 10 тисяч комп'ютерів; звання «Інтернет» почало плавно переходити до NSFNet.

1988 року було винайдено протокол Internet Relay Chat (IRC), завдяки якому в Інтернеті стало можливим спілкування в реальному часі (чат).

1989 року в Європі, народилась концепція тенет. Її запропонував знаменитий британський вчений Тім Бернерс-Лі, він же протягом двох років розробляв протокол HTTP, мову гіпертекстової розмітки HTML та ідентифікатори URI.

1990 року мережа ARPANET припинила своє існування, програвши конкуренцію NSFNet.

1992-го року компанія Sprint Corporation вперше почала надавати послугу під'єднання до Internet за допомогою модема і телефонної лінії.

1991 року http-сервери стали доступні в Інтернеті, а 1993 року з'явився знаменитий веб-браузер NCSA Mosaic.

1995 року було сформовано Консорціум всесвітньої павутини (англ. World Wide Web Consortium).

З 1996 року Всесвітнє павутиння майже повністю підмінило собою поняття «Інтернет».

Станом на 1997 рік в Інтернеті нараховувалось близько 10 мільйонів комп'ютерів і було зареєстровано понад мільйон доменних назв. Інтернет став дуже популярним засобом обміну інформацією.

2.4. Середовища та мови програмування, середовища для розробки програмного забезпечення

Laravel — безкоштовний, з відкритим кодом PHP-фреймворк, створений Тейлором Отвелом і призначений для розробки вебдодатків відповідно до шаблону model–view–controller (MVC).

Серед особливостей Laravel можна назвати: модульну систему упакування з виділеним менеджером залежностей Composer, різні способи для

доступу до реляційних баз даних, утиліти, які допомагають в розгортанні додатків і технічного обслуговування, а також його орієнтація на “синтаксичний цукор”.

Станом на березень 2015 року, Laravel вважався одним з найпопулярніших PHP фреймворком, разом з Symfony5, Nette, CodeIgniter, Yii2 й іншими.

Сирцевий код Laravel'a розміщується на GitHub і ліцензований відповідно до умов MIT License.

Тейлор Отвел створив Laravel як спробу забезпечити досконалішу альтернативу фреймворку CodeIgniter, який не забезпечував певні функції, такі як вбудовану підтримку для аутентифікації і авторизації користувачів. Перша бета-версія Laravel була розміщена 9 червня 2011 року, пізніше випустили Laravel 1 в тому ж місяці. Laravel 1 містить вбудовану підтримку для аутентифікації, локалізації, моделі, уявлення, сесій, маршрутизації та інших механізмів, але відсутня підтримка контролерів, що не заважає йому бути справжнім MVC фреймворком.

Laravel 2 був випущений у вересні 2011 року, Laravel 3 - у лютому 2012 року, Laravel 4 - у травні 2013 року, Laravel 5 - у лютому 2015 року, Laravel 6 - у вересні 2019 року, Laravel 7 - у березні 2020 року, Laravel 8 - у вересні 2020 року.

PHP – скриптова мова програмування, була створена для генерації HTML-сторінок на стороні вебсервера. PHP є однією з найпоширеніших мов, що використовуються у сфері веб розробок (разом із Java, .NET, JavaScript, Python, Ruby). PHP підтримується переважною більшістю хостинг-провайдерів. PHP — проєкт відкритого програмного забезпечення.

Всі сценарії оформляються у вигляді блоків коду. Ці блоки можуть бути поміщені в HTML-код, але відділені від нього відповідними обмежувачами. Код PHP в HTML повинен знаходитись між початковим тегом <?php та кінцевим ?> (або між <script language="php"> та </script>) Бажаним варіантом виділення PHP коду є варіант <?php ?>, оскільки саме такі початковий та

кінцевий теги дозволять використовувати PHP код у документах, які відповідають правилам XML. Також можна використовувати скорочений запис <? ?> (інколи потрібно активізувати даний стиль у файлі налаштувань інтерпретатора php.ini: змінна short_open_tag повинна мати значення On) і записом у стилі ASP: <% %> (в php.ini змінна asp_tags повинна мати значення On). Проте, стиль ASP не рекомендується і очікується, що він буде відсутній у PHP 6.

JetBrains PhpStorm — комерційне крос-платформове інтегроване середовище розробки для PHP, яке розробляється російською компанією JetBrains на основі платформи IntelliJ IDEA.

PhpStorm являє собою інтелектуальний редактор для PHP, HTML і JavaScript з можливостями аналізу коду на льоту, запобігання помилок у сирцевому коді і автоматизованими засобами рефакторинга для PHP і JavaScript. Автодоповнення коду в PhpStorm підтримує специфікацію PHP 5.3/5.4/5.5/5.6/7.0/7.1 (сучасні і традиційні проекти), включаючи генератори, співпрограми, простори імен, замикання, типажі і синтаксис коротких масивів. Присутній повноцінний SQL-редактор з можливістю редагування отриманих результатів запитів.

PhpStorm розроблений на основі платформи IntelliJ IDEA, написаної на Java. Користувачі можуть розширити функціональність середовища розробки за рахунок установки плаґінів, розроблених для платформи IntelliJ, або написавши власні плаґіни.

Вся функціональність WebStorm включена в PhpStorm.

РОЗДІЛ 3. ТЕОРЕТИЧНА ЧАСТИНА

3.1. Огляд матеріалу з теми

Ідентифікація — процедура розпізнавання користувача в системі, як правило, за допомогою наперед визначеного імені (ідентифікатора) або іншої апріорної інформації про нього, яка сприймається системою.

Є початковою процедурою надання доступу до системи. Після неї здійснюється автентифікація та авторизація.

Ідентифікація дозволяє суб'єктові (користувачу, процесу, який діє від імені певного користувача) повідомити своє ім'я за допомогою унікального параметра — ідентифікатора (логін, наприклад), який є відомим іншій стороні. Під час ідентифікації здійснюється порівняння заявленого суб'єктом параметра на відповідність відомому іншій стороні. В разі успішної ідентифікації відбувається Автентифікація. Шляхом автентифікації інша сторона переконується що суб'єкт саме той за кого він себе видає (використовується пароль у випадку парольної автентифікації або інший секретний параметр).

Автентифікація — процедура встановлення належності користувачеві інформації в системі пред'явленого ним ідентифікатора.

З позицій інформаційної безпеки автентифікація є частиною процедури надання доступу для роботи в інформаційній системі, наступною після ідентифікації і передує авторизації.

Авторизація — керування рівнями та засобами доступу до певного захищеного ресурсу, як у фізичному розумінні (доступ до кімнати готелю за карткою), так і в галузі цифрових технологій (наприклад, автоматизована система контролю доступу) та ресурсів системи залежно від ідентифікатора і пароля користувача або надання певних повноважень (особі, програмі) на виконання деяких дій у системі обробки даних.

З позицій інформаційної безпеки Авторизація є частиною процедури надання доступу для роботи в інформаційній системі, після ідентифікації і автентифікації.

Керування доступом на основі ролей — розвиток політики вибіркового керування доступом, при якому права доступу суб'єктів системи на об'єкти групуються з урахуванням специфіки їх застосування, утворюючи ролі.

Формування ролей покликане визначити чіткі і зрозумілі для користувачів комп'ютерної системи правила розмежування доступу. Рольове розмежування доступу дозволяє реалізувати гнучкі та динамічно змінні в процесі функціонування комп'ютерної системи правила розмежування доступу.

Таке розмежування доступу є складовою багатьох сучасних комп'ютерних систем. Як правило, даний підхід застосовується в системах захисту СУБД, а окремі елементи реалізуються в мережевих операційних системах. Рольовий підхід часто використовується в системах, для користувачів яких чітко визначено коло їх посадових повноважень і обов'язків.

Незважаючи на те, що Роль є сукупністю прав доступу на об'єкти комп'ютерної системи, рольове керування доступом аж ніяк не є окремим випадком вибіркового управління доступом, так як його правила визначають порядок надання доступу суб'єктам комп'ютерної системи в залежності від наявних (або відсутніх) у нього ролей в кожен момент часу, що є характерним для систем мандатного керування доступом. З іншого боку, правила рольового розмежування доступу є більш гнучкими, ніж при мандатному підході до розмежування.

Так як привілеї не призначаються користувачам безпосередньо і отримуються ними тільки через свою роль (або ролі), управління індивідуальними правами користувача по суті зводиться до призначення йому ролей. Це спрощує такі операції, як додавання користувача або зміна підрозділу користувачем.

Кросплатформність — властивість програмного забезпечення працювати більш ніж на одній програмній (в тому числі — операційній системі) або

апаратній платформі; технології, що дозволяють досягти такої властивості. Кросплатформність дозволяє суттєво скоротити витрати на розробку нового або адаптацію існуючого програмного забезпечення.

Залежно від засобів реалізації поділяється на кросплатформність на рівні мов програмування (а також інструментів таких мов: компіляторів та редакторів зв'язків), середовища виконання, операційної системи та апаратної платформи.

Кросплатформність на рівні мов програмування досягається шляхом забезпечення незалежності програмного коду від платформи. Багатоплатформними є більшість сучасних високорівневих мов програмування, для яких реалізовані транслятори, що можуть виконуватись на різних платформах. Наприклад, C, C++ і Pascal — кросплатформні мови на рівні компіляції, тобто для цих мов є компілятори під різні платформи.

Кросплатформність на рівні редакторів зв'язків досягається реалізацією для різних платформ кросплатформних бібліотек, які реалізують незалежний від платформи інтерфейс, в тому числі — стандартизованих бібліотек. Зокрема, стандартизовані багато бібліотек мови Сі (див. POSIX). Існує також велика кількість нестандартних кросплатформних бібліотек: Qt, GTK+, FLTK, STL, Boost, OpenGL, SDL, OpenAL, OpenCL.

Фреймворк — абстракція, в якій програмне забезпечення, що забезпечує загальну функціональність, може бути вибірково змінено додатковим кодом, написаним користувачем, таким чином забезпечуючи програмне забезпечення, специфічне для програми. Він забезпечує стандартний спосіб створення та розгортання додатків і є універсальним багаторазовим програмним середовищем, яке надає певні функції як частину більшої програмної платформи для полегшення розробки програмних додатків, продуктів і рішень.

Фреймворки програмного забезпечення можуть включати програми підтримки, компілятори, бібліотеки коду, набори інструментів та інтерфейси

прикладного програмування (API), які об’єднують усі різні компоненти для забезпечення розробки проекту чи системи.

Фреймворки мають ключові відмінні риси, які відрізняють їх від звичайних бібліотек:

- інверсія керування: у фреймворку, на відміну від бібліотек або стандартних користувальницьких програм, загальний потік керування програмою диктується не викликачем, а фреймворком.

- поведінка за замовчуванням: це може бути забезпечено інваріантними методами шаблону методу шаблону в абстрактному класі, який надається фреймворком.

- розширюваність: користувач може розширити фреймворк—зазвичай шляхом вибіркового перевизначення - або програмісти можуть додати спеціальний код користувача для надання певних функцій. Зазвичай це досягається за допомогою методу підключення в підкласі, який перевизначає метод шаблону в суперкласі.

- код фреймворка, який не підлягає зміні: код фреймворка, як правило, не повинен бути змінений, але приймає розширення, реалізовані користувачем. Іншими словами, користувачі можуть розширювати структуру, але не можуть змінювати її код.

3.2. Алгоритм роботи підсистеми

На стартовому екрані користувач бачить форму авторизації з наступними полями:

- електронна пошта;
- пароль;
- прапорець для можливості запам’ятати сесію

Якщо вибрати прапорець “*Запам’ятати мене*”, термін життя сесії користувача буде значно збільшений.

Після заповнення усіх необхідних даних і відправки форми, відбувається ідентифікація користувача за його електронною поштою.

Якщо запитуваний користувач присутній в базі даних системи, визначається його роль, і список його дозволів. В системі описано 2 ролі:

Староста. Користувач має право заповнити інформацію щодо присутності студентів групи на відповідній парі, та з інформаційною метою, староста групи має можливість проглядати відвідуванність студентів, але це право обмежено його групою.

Адміністратор. Користувач має необмежені права, в тому числі доступ до відвідуванності усіх наявних в системі груп, за будь-який часовий проміжок.

Покроковий приклад використанння підсистеми старостою:

1 крок. Користувач проходить авторизацію.

2 крок. На панелі навігації натискає: «*Заповнити присутність*»

3 крок. В формі заповнюються необхідні дані: дата, предмет, пара, та відмічаються присутні студенти.

4 крок. Після відправки форми відбувається переспрямування на сторінку з журналом відвідування групи користувача, з заповненими даними

3.3 Обґрунтування вибору програмних засобів для реалізації завдання роботи

Для реалізації підсистеми було обрано мову програмування - PHP. Це рішення було прийняте через те, що наразі переважна кількість людей використовує свій персональний комп'ютер і/або телефон в якості інструменту доступу до мережі Інтернет, а тому «WEB-застосунок» є більш зручним та значною мірою більш звичним для звичайних користувачів. І тому для написання коду для реалізації підсистеми було обрано найбільш підходящу мову програмування - PHP.

Також однією з переваг цієї мови є її мультиплатформність, що дозволяє підсистемі виконуватися на будь-якій операційній системі не залежно від її архітектури та типу.

Ще одним позитивним аспектом цієї мови є велика кількість бібліотек, фрейморків, стандартних рішень що в свою чергу значно спростило процес створення підсистеми і терміни виконання задачі.

В процесі розробки підсистеми було використано найбільш розповсюджене середовище розробки PhpStorm. Головною причиною вибору PhpStorm, як основного середовища розробки, було те, що він є крос-платформеним, як результат з можливістю крос-платформенного запуску, наприклад в операційній системі Linux, а також його зручність. PhpStorm є напрацюванням чешської компанії JetBrains. Зокрема, PhpStorm є надбудовою над IntellijIdea яка є в основі всіх продуктів компанії JetBrains.

Інтегроване середовище розробки містить вбудований завантажувач, інструменти для роботи з системою контролю версій Git, а також інструменти для рефакторингу, навігації по коду, автодоповнення типових мовних конструкцій. Дане інтегроване середовище розробки підтримує в тому числі роботу з базами даних, шляхом використання вбудованого інструменту DataGrip.

Для забезпечення потрібного рівня надійності та зручності розробки було обрано систему контролю версій Git. Git є однією з найвідоміших та найефективніших та високопродуктивних систем керування версіями, що надає цілий інструментарій зручних засобів нелінійної розробки, які базуються на відгалуженні і злитті гілок різних версій проекту. Для забезпечення цілісності історії змін та стійкості до змін заднім числом використовуються низка криптографічних методів, а також можлива прив'язка цифрових відбитків розробників до тегів і комітів. Ця система спроєктована як набір програм, розроблених спеціально з врахуванням використання у скриптах, що дає можливість зручно створювати спеціалізовані системи керування версіями або користувацькі інтерфейси.

РОЗДІЛ 4. ПРАКТИЧНА ЧАСТИНА

4.1. Опис програмної реалізації

Для створення користувацького інтерфейсу додатку була використана така технології як Bootstrap. Серед іншого Bootstrap впроваджує засоби для:

- створення структурованого інтерфейсу користувача;
- створення інтерактивних форм.

Bootstrap — це безкоштовний набір інструментів з відкритим кодом, призначений для створення вебсайтів та вебдодатків, який містить шаблони CSS та HTML для типографіки, форм, кнопок, навігації та інших компонентів інтерфейсу, а також додаткові розширення JavaScript. Він спрощує розробку динамічних вебсайтів і вебдодатків.

Bootstrap — це клієнтський фреймворк, тобто інтерфейс для користувача, на відміну від коду серверної сторони, який знаходиться на сервері. Репозиторій із цим фреймворком є одним із найпопулярніших на GitHub. Серед інших, його використовують NASA і MSNBC

Laravel – фреймворк, що відповідає за архітектуру підсистеми і реалізацію бізнес-логіки. Даний фреймворк спрощує реалізацію авторизації користувача, роботу з базою даних, і імплементування доступу до частин системи на основі ролей користувача, за рахунок використання вже існуючої функціональності в сирцевому коді. Причому фреймворк також має в собі компоненти – набір готових рішень для вирішення типових проблем в процесі розробки. Тобто програміст може значно підвищити швидкість і якість розробки застосунку шляхом використання вже існуючих кваліфікованих рішень.

З одного боку, Laravel розглядається як заміна більш старого фреймворку Codeigniter. З іншого боку, Laravel не просто заміняє Codeigniter, а й надає перелік значно вагоміших інструментів для ефективної та зручної розробки.

Розглянуті обрані інструменті та технології, які були необхідні для створення програмного продукту. Розроблений програмний продукт написано скриптовою мовою програмування PHP з використанням фреймворку Laravel. Розробка проводилась з використанням середовища JetBrains PHP Storm та за допомогою принципів плаского дизайну створення графічних інтерфейсів.

Створюваний додаток розрахований на значне якісне покращення певного набору організаційних процесів деканату.

Створюваний додаток повинен мати такі вимоги:

- простий та інтуїтивно зрозумілий інтерфейс;
- можливість заповнення відвідуванності старостою
- можливість повного доступу адміністратором

Оскільки створюваний додаток не є масштабним, то для початку планується випустити версію з мінімальним базовим функціоналом.

Даний програмний продукт розроблювався за допомогою IDE PHP Storm. Тип застосунку можна описати як веб-портал.

Сторінка авторизації (див. Рисунок 4.1) дозволяє користувачеві пройти процедуру авторизації.

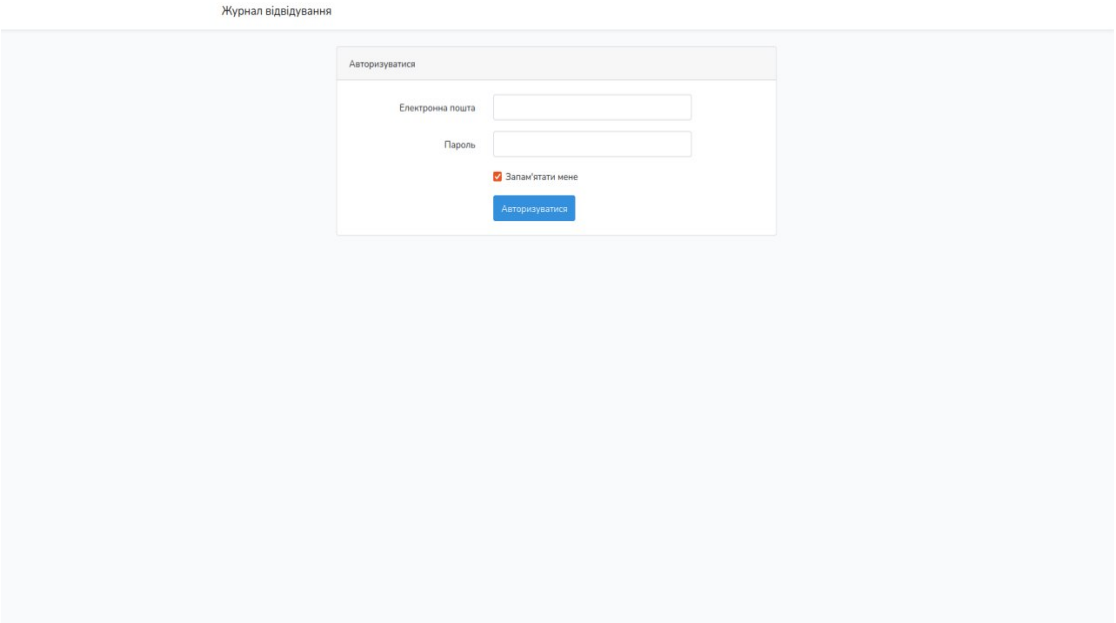


Рисунок 4.1 – Сторінка авторизації

При успішному проходженюю процедури авторизації, можна спостерігати форму для отримання інформації з відвідуванності студентів за потрібний період часу (див. Рисунок 4.2).

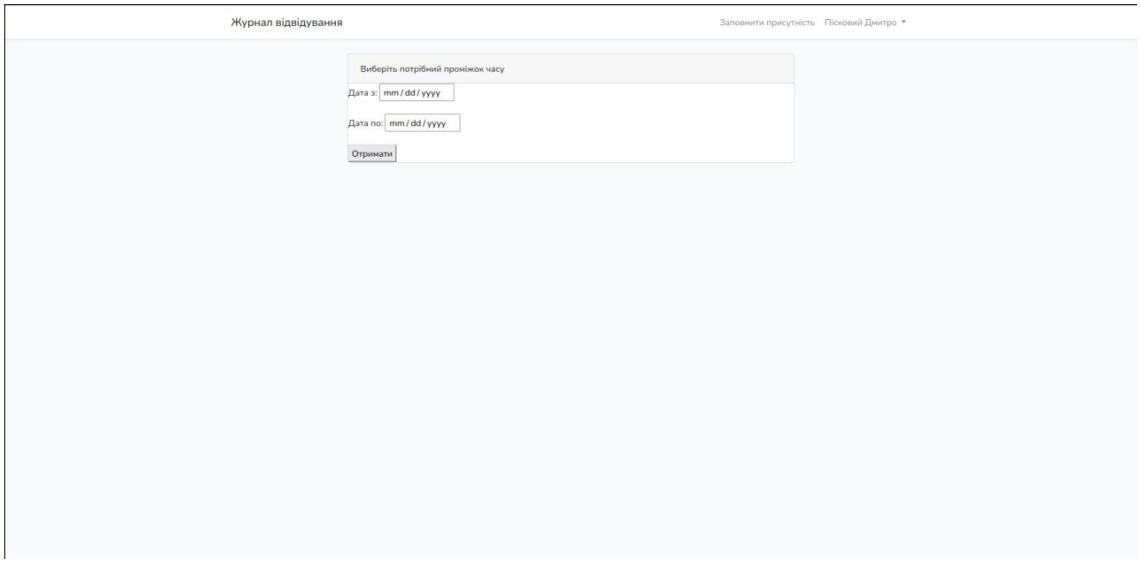


Рисунок 4.2 – Форма для отримання даних

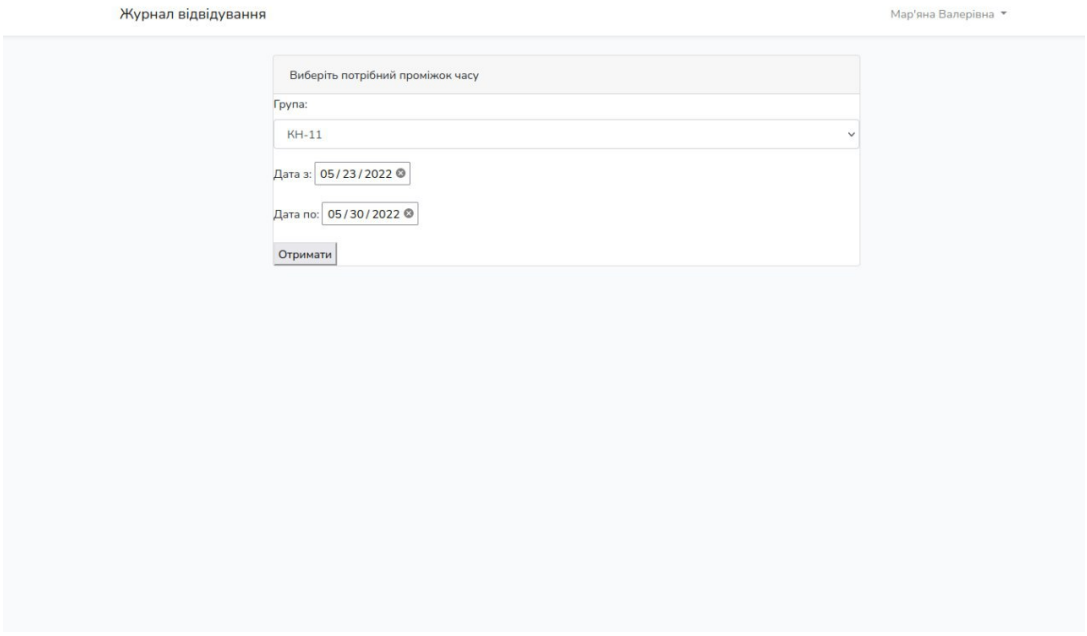


Рисунок 4.3 – Форма для отримання даних представнику адміністрації

Під час взаємодії з порталом староста заповнює дані присутності студентів групи на вказаній парі (див. Рисунок 4.4). Після заповнення відповідної форми відбувається перенаправлення на сторінку з таблицею, де відображена інформація про відвідування пар студентами групи за вибраний період (див. Рисунок 4.5).

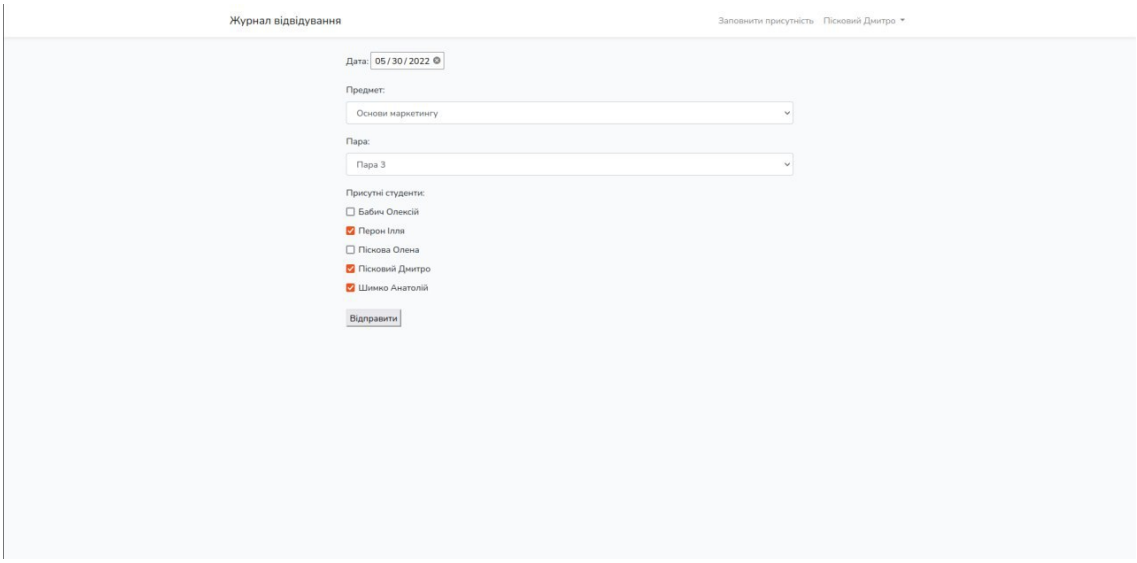


Рисунок 4.4 – Форма для заповнення присутності студентів

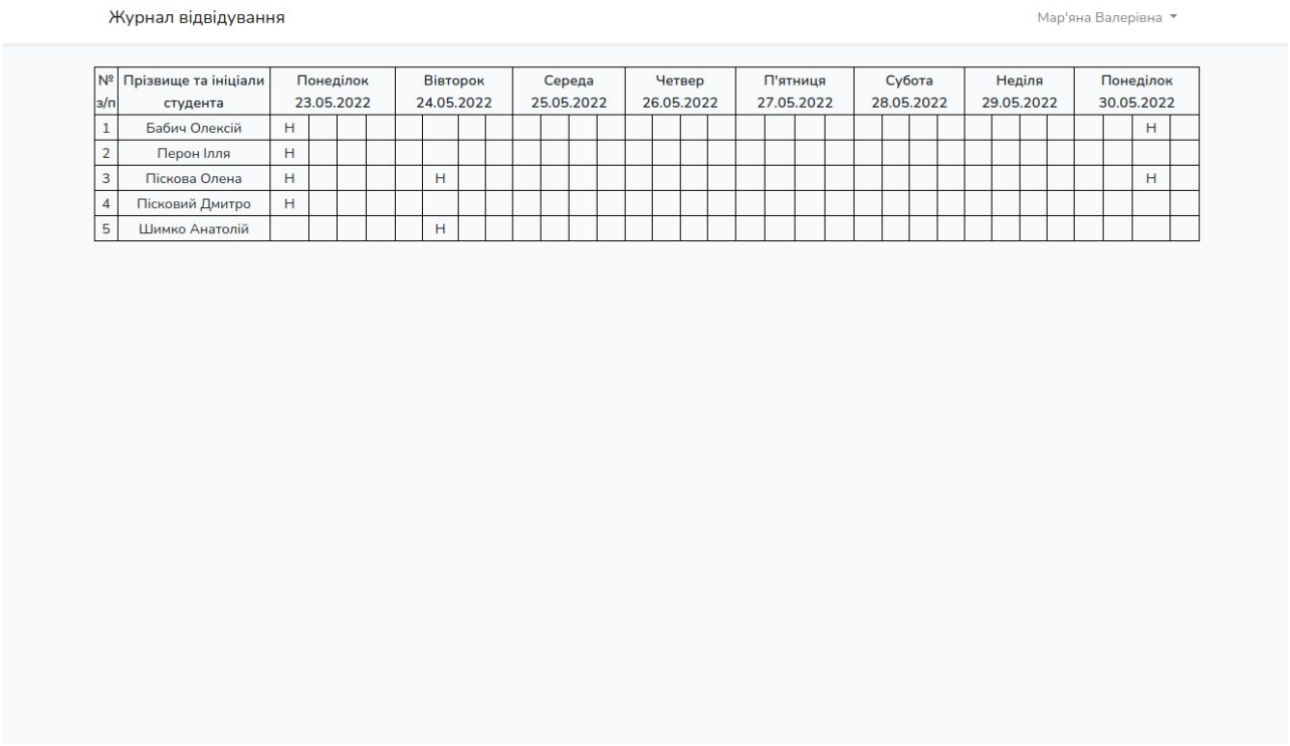


Рисунок 4.5 – Таблиця з відвідуванністю студентів

Для створення користувацького інтерфейсу додатку була використана така технології як Bootstrap. Серед іншого Bootstrap впроваджує засоби для:

- створення структурованого інтерфейсу користувача;
- створення інтерактивних форм.

Bootstrap — це безкоштовний набір інструментів з відкритим кодом, призначений для створення вебсайтів та вебдодатків, який містить шаблони CSS та HTML для типографіки, форм, кнопок, навігації та інших компонентів

інтерфейсу, а також додаткові розширення JavaScript. Він спрощує розробку динамічних вебсайтів і вебдодатків.

Bootstrap — це клієнтський фреймворк, тобто інтерфейс для користувача, на відміну від коду серверної сторони, який знаходиться на сервері. Репозиторій із цим фреймворком є одним із найпопулярніших на GitHub. Серед інших, його використовують NASA і MSNBC

Laravel – фреймворк, що відповідає за архітектуру підсистеми і реалізацію бізнес-логіки. Даний фреймворк спрощує реалізацію авторизації користувача, роботу з базою даних, і імплементування доступу до частин системи на основі ролей користувача, за рахунок використання вже існуючої функціональності в сирцевому коді. Причому фреймворк також має в собі компоненти – набір готових рішень для вирішення типових проблем в процесі розробки. Тобто програміст може значно підвищити швидкість і якість розробки застосунку шляхом використання вже існуючих кваліфікованих рішень.

З одного боку, Laravel розглядається як заміна більш старого фреймворку Codeigniter. З іншого боку, Laravel не просто заміняє Codeigniter, а й надає перелік значно вагоміших інструментів для ефективної та зручної розробки.

Розглянуті обрані інструменті та технології, які були необхідні для створення програмного продукту. Розроблений програмний продукт написано скриптовою мовою програмування PHP з використанням фреймворку Laravel. Розробка проводилась з використанням середовища JetBrains PHP Storm та за допомогою принципів плаского дизайну створення графічних інтерфейсів.

Створюваний додаток розрахований на значне якісне покращення певного набору організаційних процесів деканату.

Створюваний додаток повинен мати такі вимоги:

- простий та інтуїтивно зрозумілий інтерфейс;
- можливість заповнення відвідуванності старостою
- можливість повного доступу адміністратором

Оскільки створюваний додаток не є масштабним, то для початку планується випустити версію з мінімальним базовим функціоналом.

Даний програмний продукт розроблювався за допомогою IDE PHP Storm. Тип застосунку можна описати як веб-портал.

Сторінка авторизації (див. Рисунок 4.1) дозволяє користувачеві пройти процедуру авторизації.

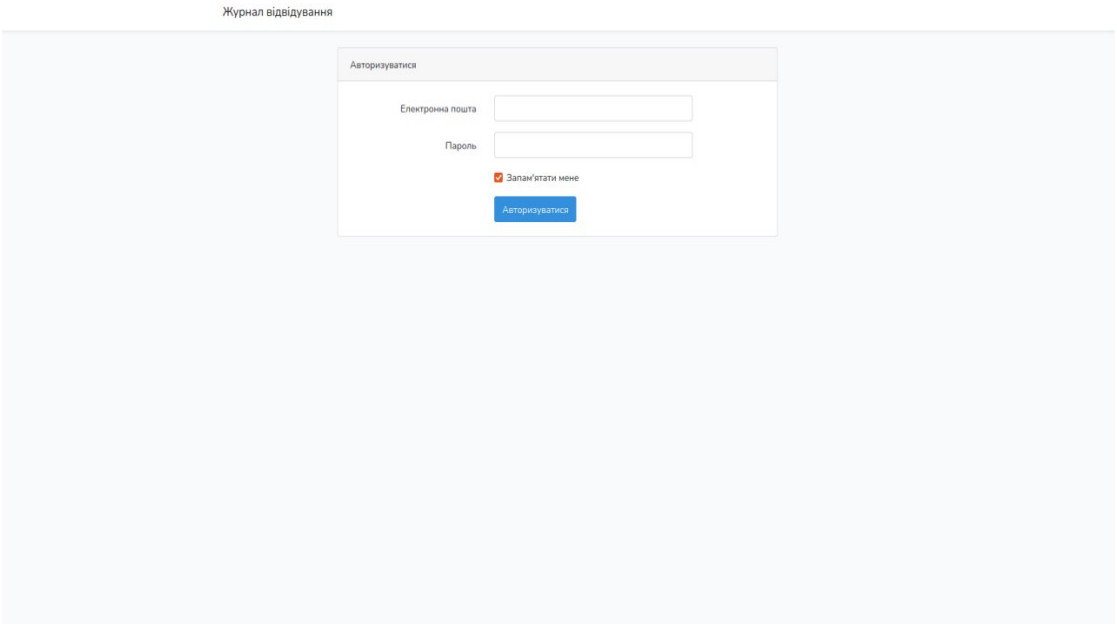


Рисунок 4.1 – Сторінка авторизації

При успішному проходженюю процедури авторизації, можна спостерігати форму для отримання інформації з відвідуванності студентів за потрібний період часу (див. Рисунок 4.2).

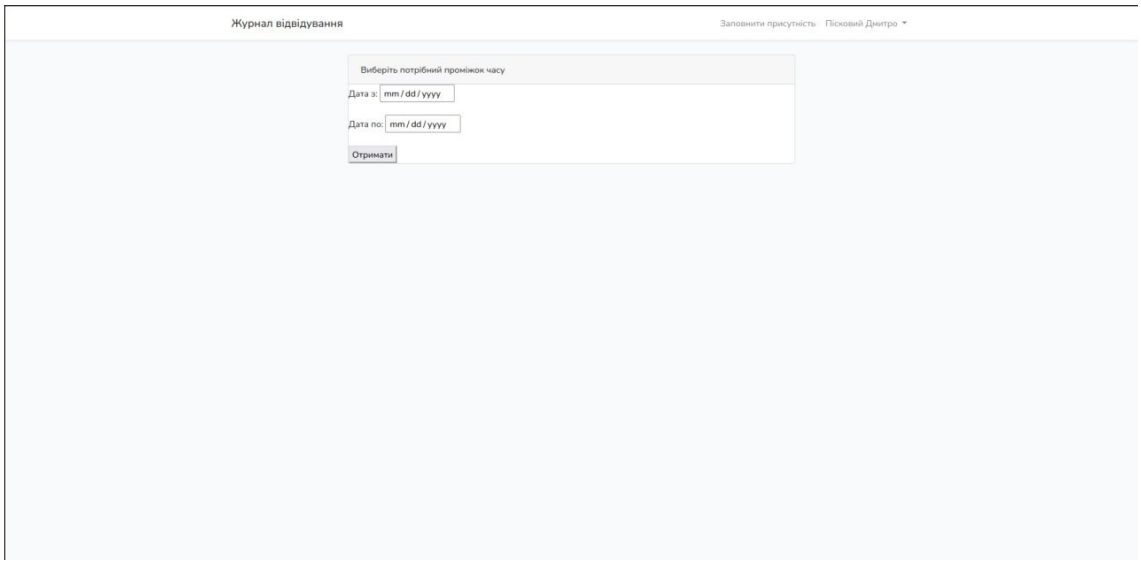


Рисунок 4.2 – Форма для отримання даних

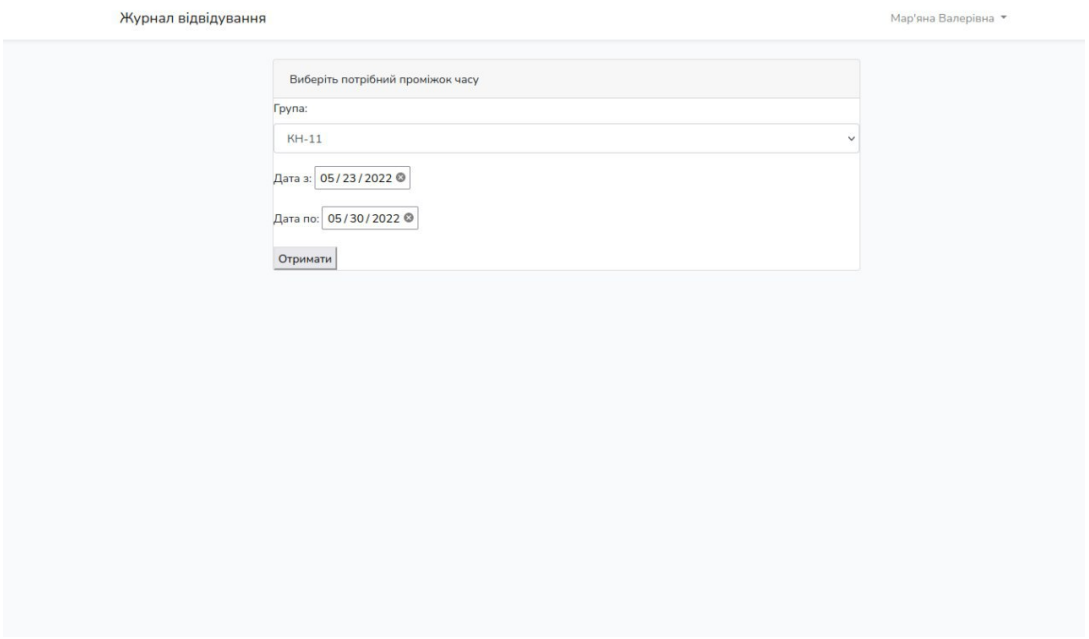


Рисунок 4.3 – Форма для отримання даних представнику адміністрації

Під час взаємодії з порталом староста заповнює дані присутності студентів групи на вказаній парі (див. Рисунок 4.4). Після заповнення відповідної форми відбувається перенаправлення на сторінку з таблицею, де відображена інформація про відвідування пар студентами групи за вибраний період (див. Рисунок 4.5).

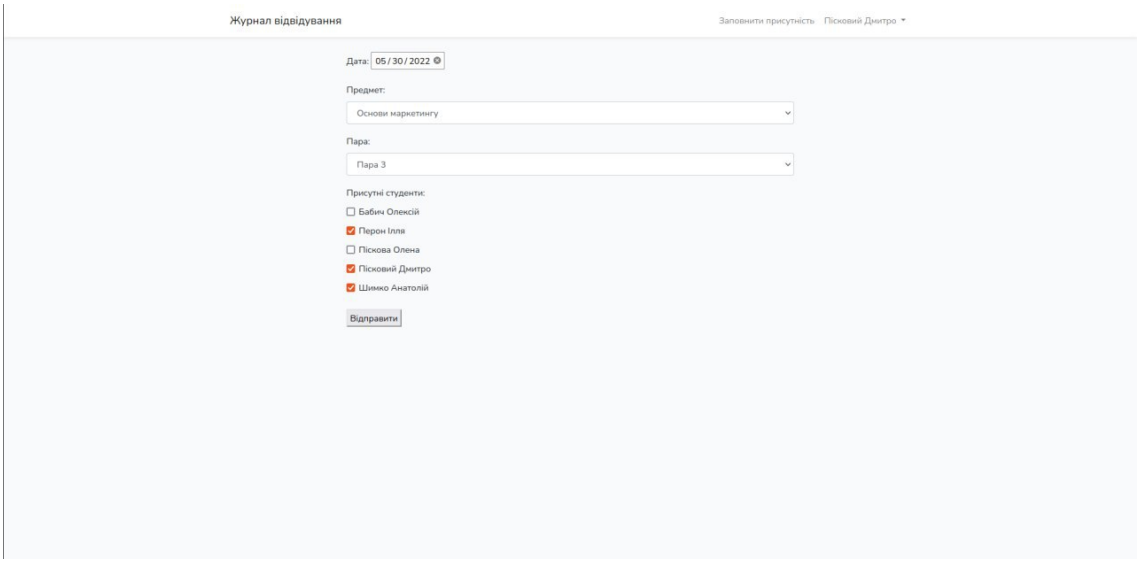
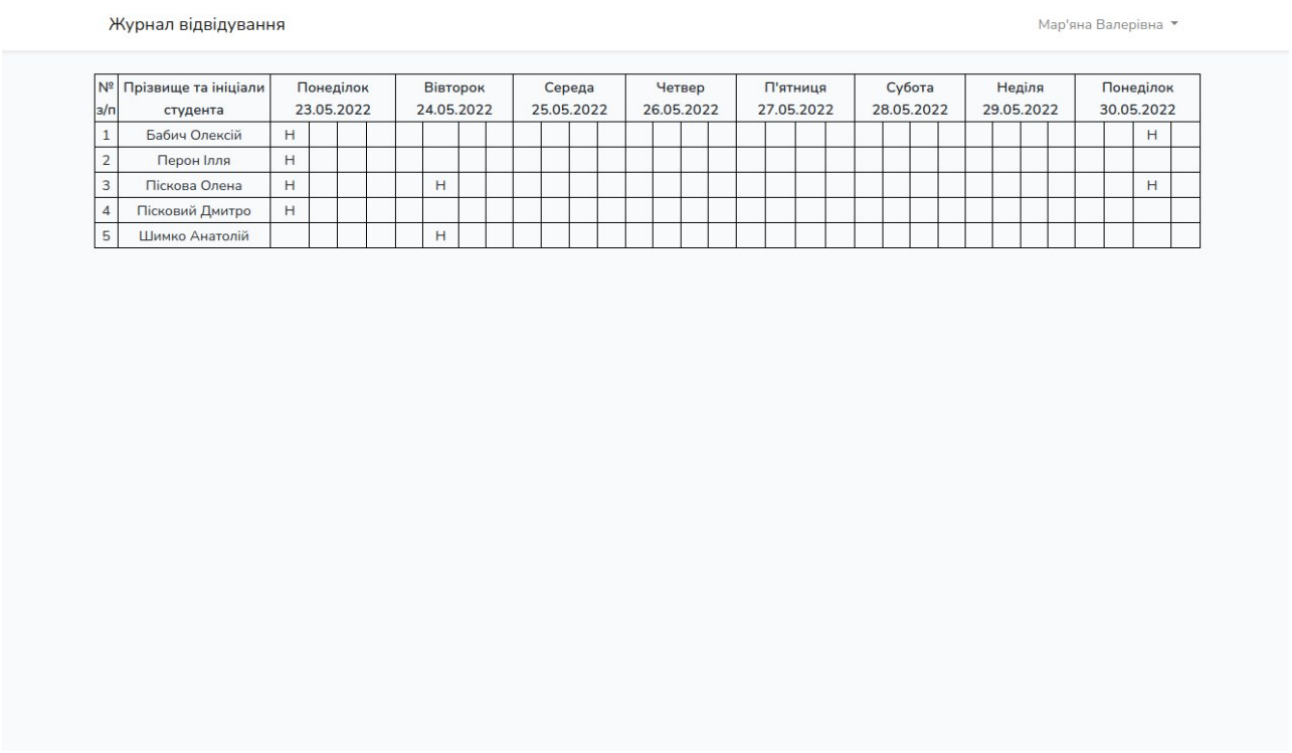


Рисунок 4.4 – Форма для заповнення присутності студентів



№ з/п	Прізвище та ініціали студента	Понеділок 23.05.2022				Вівторок 24.05.2022				Середа 25.05.2022				Четвер 26.05.2022				П'ятниця 27.05.2022				Субота 28.05.2022				Неділя 29.05.2022				Понеділок 30.05.2022			
1	Бабич Олексій	Н																												Н			
2	Перон Ілля	Н																															
3	Піскова Олена	Н				Н																								Н			
4	Пісковий Дмитро	Н																															
5	Шимко Анатолій					Н																											

Рисунок 4.5 – Таблиця з відвідуванністю студентів

4.2. Інструкція по роботі з програмою

Після завантаження сторінки з веб-порталом користувач спостерігає форму для авторизації

Форма має наступні поля:

- Електронна пошта.
- Пароль.

По вказаній електронній адресі визначається роль користувач, і перелік дозволених взаємодій з веб-порталом і інформації доступної на ньому.

Після успішної авторизації в залежності від ролі користувач має змогу:

- Переглядати дані з відвідуванності студентів за поточний тиждень
- Вносити дані з відвідуванності студентів конкретної групи на вказаній парі яка проводилася в вказану дату
- Переглядати дані з відвідуванності будь-якої групи студентів за будь-який період часу

Також маючи доступ до безпосередньо серверу, адміністрація порталу має змогу

- Отримати інформацію з приводу того ким саме і коли була заповнена інформація з відвідуванності
- Додати інформацію стосовно групи: список студентів групи, старосту, уроки які мали бути в групи згідно розкладу

Після ознайомлення з наданою інформацією і її корегування користувач має змогу завершити свою поточну сесію.

ВИСНОВКИ

Метою виконаного проекту є алгоритмізація, програмування, та безпосереднє створення елементів підсистеми рейтингування викладачів «Електронний деканат» для оптимізації, автоматизації та «діджиталізації» частини паперового документообігу та керування існуючими процесами в університеті в цілому .

В проекті описано такі пункти:

- Постановка задачі;
- Середовища та мови програмування, середовища для розробки програмного забезпечення;
- Огляд інформаційного матеріалу з теми;
- Практична частина;
- Алгоритм роботи застосунку.

Технологічна основа навчального процесу, у тому числі і сучасні інформаційні технології, швидко розвиваються. Застосування комп'ютерних технологій підвищує якість, зручність і продуктивність існуючих процесів, веде до перебудови навчального процесу в бік оптимізації і автоматизації. Без перевантажень можна інтенсифікувати процес виконання буденних задач в умовах профільного навчання завдяки раціональному використанню комп'ютерних технологій. Найбільш цінними у навчальному процесі виявляються інтуїтивно зрозумілі програмні засоби без перевантаження інтерфейсу і логіки дій, і які не вимагають спеціальної кваліфікації для швидкого опанування і початку взаємодії, засоби, що надають користувачеві свободу вибору того чи іншого способу взаємодії. За допомогою комп’ютерних технологій, існуючі процеси відчутно спрощуються, і дозволяють виконувати звичні задачі більш якісно і продуктивно, а деякі задачі зникають за рахунок автоматизації.

СПИСОК ЛІТЕРАТУРИ

1. Ємець О. О. Методичні рекомендації щодо оформлення пояснювальних записок до курсових проектів (робіт) для студентів за освітньою програмою «Комп'ютерні науки» спеціальності 122 «Комп'ютерні науки та інформаційні технології», «Комп'ютерні науки» галузь знань - 12 «Інформаційні технології» / О. О. Ємець – Полтава : РВВ ПУЕТ, 2017. – 69с.
2. Палюх Б.В. Электронное обучение в инженерном образовании / Б.В. Палюх, А. В. Твардовский, В.К. Иванов, – 2012.– Качество образования, 10, с.34–37.
3. «Laravel guide» [Електронний ресурс] – Режим доступу: https://docs.laravel.com/en-us/guide/quickstart/.
4. PHP [Електронний ресурс] / Матеріал з Вікіпедії — вільної енциклопедії. – Режим доступу: https://php.net
5. JetBrains PHP Storm [Електронний ресурс] / Матеріал з Вікіпедії — вільної енциклопедії. – Режим доступу: https://uk.wikipedia.org/wiki/Php_Storm
6. PHP: Посібник [Електронний ресурс] – Режим доступу до ресурсу URL: https://programm.top/uk/php/tutorial.

ДОДАТОК А. ПРОГРАМНИЙ КОД

Опрацювання відвідуваності студентів

```
<?php

namespace App\Services;

use App\Dto\StudentPresenceDto;
use App\Models\Lesson;
use App\Models\LessonStudentPresence;
use Illuminate\Support\Facades\DB;

class StudentPresenceService
{
    public function savePresentStudents(StudentPresenceDto $studentPresenceDto): void
    {
        DB::beginTransaction();

        try {
            $lesson = $this->saveLesson($studentPresenceDto);
            $this->clearExistingLessonPresence($lesson);
            $data = $this->prepareStudentPresenceData($studentPresenceDto, $lesson);
            LessonStudentPresence::insert($data);
            DB::commit();
        } catch (\Exception $exception) {
            DB::rollback();
            throw $exception;
        }
    }

    private function prepareStudentPresenceData(StudentPresenceDto $studentPresenceDto, Lesson
$lesson): array
    {
        $result = [];

        $lessonId = $lesson->id;

        foreach ($studentPresenceDto->studentsPresent as $studentId) {
            $result[] = [
                'lesson_id' => $lessonId,
                'student_id' => $studentId,
            ];
        }

        return $result;
    }

    private function saveLesson(StudentPresenceDto $studentPresenceDto): Lesson
    {
        $lesson = Lesson::where(['date' => $studentPresenceDto->date])
            ->where(['group_id' => $studentPresenceDto->groupId])
            ->where(['subject_id' => $studentPresenceDto->subjectId])
            ->where(['lesson by count' => $studentPresenceDto->lessonByCount]);

        if($lesson->exists()){
            return $lesson->first();
        }

        return Lesson::create([
            'date' => $studentPresenceDto->date,
            'group_id' => $studentPresenceDto->groupId,
            'subject_id' => $studentPresenceDto->subjectId,
            'lesson_by_count' => $studentPresenceDto->lessonByCount
        ]);
    }

    private function clearExistingLessonPresence(Lesson $lesson): void
    {
        LessonStudentPresence::where(['lesson_id' => $lesson->id])->delete();
    }
}
```

Відображення даних в таблиці

```
<?php

namespace App\Services;

use App\Models\Lesson;
use App\Models\LessonStudentPresence;
use Illuminate\Support\Collection;

class StudentPresenceTableService
{
    /**
     * @var Lesson[]
     */
    public $lessons;

    /**
     * @var LessonStudentPresence[]
     */
    public $lessonStudentsPresence;

    private $studentPresence;
    private $groupedLessons;

    public function __construct(Collection $lessons, Collection $lessonStudentsPresence)
    {
        $this->lessons = $lessons;
        $this->lessonStudentsPresence = $lessonStudentsPresence;
        $this->studentPresence = $this->prepareStudentPresence($lessons, $lessonStudentsPresence);
        $this->groupedLessons = $this->prepareLessons($lessons);
    }

    public function checkIfStudentWasPresent(string $date, int $lessonByCount, int $studentId): bool
    {
        if(isset($this->studentPresence[$date][$lessonByCount])) {
            $presentStudents = $this->studentPresence[$date][$lessonByCount][0];
            if(!in_array($studentId, $presentStudents, true)){
                return false;
            }
        }

        return true;
    }

    public function getLessonSubjectName(string $date, int $lessonByCount): string
    {
        /** @var $lesson Lesson*/
        if(isset($this->groupedLessons[$date][$lessonByCount])){
            $lesson = $this->groupedLessons[$date][$lessonByCount];
            $lessonName = $lesson->subject->name;
        }

        return $lessonName ?? '';
    }

    private function prepareStudentPresence(Collection $lessons, Collection
$lessonStudentsPresence): array
    {
        $result = [];
        $groupedLessonPresence = $lessonStudentsPresence
            ->groupBy('lesson id')
            ->toArray();

        foreach ($lessons as $lesson) {
            $presentStudents = $groupedLessonPresence[$lesson->id] ?? [];
            $result[$lesson->date][$lesson->lesson by count][] = array column($presentStudents,
'student_id');
        }

        return $result;
    }
  }
```

Модель даних таблиці “lessons_students_presence”

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

/**
 * App\Models\LessonStudentPresence
 *
 * @property int $lesson id
 * @property int $student_id
 * @method static \Illuminate\Database\Eloquent\Builder|LessonStudentPresence newModelQuery()
 * @method static \Illuminate\Database\Eloquent\Builder|LessonStudentPresence newQuery()
 * @method static \Illuminate\Database\Eloquent\Builder|LessonStudentPresence query()
 * @mixin \Eloquent
 * @property int $id
 * @property \Illuminate\Support\Carbon|null $created_at
 * @property \Illuminate\Support\Carbon|null $updated_at
 * @method static \Illuminate\Database\Eloquent\Builder|LessonStudentPresence whereCreatedAt($value)
 * @method static \Illuminate\Database\Eloquent\Builder|LessonStudentPresence whereId($value)
 * @method static \Illuminate\Database\Eloquent\Builder|LessonStudentPresence whereLessonId($value)
 * @method static \Illuminate\Database\Eloquent\Builder|LessonStudentPresence whereStudentId($value)
 * @method static \Illuminate\Database\Eloquent\Builder|LessonStudentPresence whereUpdatedAt($value)
 */
class LessonStudentPresence extends Model
{
    public function getTable(): string
    {
        return 'lessons_students_presence';
    }

    /**
     * The attributes that are mass assignable.
     *
     * @var string[]
     */
    protected $fillable = [
        'lesson id',
        'student id',
        'date',
        'lesson_by_count'
    ];

    /**
     * The attributes that should be cast.
     *
     * @var array
     */
    protected $casts = [
        'created_at' => 'datetime',
        'update_at' => 'datetime',
    ];
}
```

Модель даних таблиці “groups”

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsToMany;
use Illuminate\Database\Eloquent\Relations\HasMany;

/**
 * App\Models\Group
 *
 * @property int $id
 * @property string $name
 * @property \Illuminate\Support\Carbon|null $created_at
 * @property \Illuminate\Support\Carbon|null $updated_at
 * @method static \Illuminate\Database\Eloquent\Builder|Group newModelQuery()
 * @method static \Illuminate\Database\Eloquent\Builder|Group newQuery()
 * @method static \Illuminate\Database\Eloquent\Builder|Group query()
 * @method static \Illuminate\Database\Eloquent\Builder|Group whereCreatedAt($value)
 * @method static \Illuminate\Database\Eloquent\Builder|Group whereId($value)
 * @method static \Illuminate\Database\Eloquent\Builder|Group whereName($value)
 * @method static \Illuminate\Database\Eloquent\Builder|Group whereUpdatedAt($value)
 * @mixin \Eloquent
 * @property Student[] $students
 * @property-read int|null $students_count
 * @property Subject[] $subjects
 * @property-read int|null $subjects count
 */
class Group extends Model
{
    /**
     * The attributes that are mass assignable.
     *
     * @var string[]
     */
    protected $fillable = [
        'name',
    ];

    /**
     * The attributes that should be cast.
     *
     * @var array
     */
    protected $casts = [
        'created_at' => 'datetime',
        'update_at' => 'datetime',
    ];

    public function students(): HasMany
    {
        return $this->hasMany(
            Student::class,
            'group id',
            'id'
        )->orderBy('last name');
    }

    public function subjects(): BelongsToMany
    {
        return $this->belongsToMany(
            Subject::class,
            'groups_subjects',
            'subject_id',
            'group_id',
            'id',
            'id',
        );
    }
}
```

Модель даних таблиці “groups_subjects”

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsToMany;
use Illuminate\Database\Eloquent\Relations\HasMany;

/**
 * App\Models\GroupSubject
 *
 * @property int $id
 * @property int $group_id
 * @property int $subject_id
 * @property \Illuminate\Support\Carbon|null $created_at
 * @property \Illuminate\Support\Carbon|null $updated_at
 * @method static \Illuminate\Database\Eloquent\Builder|GroupSubject newModelQuery()
 * @method static \Illuminate\Database\Eloquent\Builder|GroupSubject newQuery()
 * @method static \Illuminate\Database\Eloquent\Builder|GroupSubject query()
 * @method static \Illuminate\Database\Eloquent\Builder|GroupSubject whereCreatedAt($value)
 * @method static \Illuminate\Database\Eloquent\Builder|GroupSubject whereGroupId($value)
 * @method static \Illuminate\Database\Eloquent\Builder|GroupSubject whereId($value)
 * @method static \Illuminate\Database\Eloquent\Builder|GroupSubject whereSubjectId($value)
 * @method static \Illuminate\Database\Eloquent\Builder|GroupSubject whereUpdatedAt($value)
 * @mixin \Eloquent
 */
class GroupSubject extends Model
{
    public function getTable(): string
    {
        return 'groups subjects';
    }

    /**
     * The attributes that are mass assignable.
     *
     * @var string[]
     */
    protected $fillable = [
        'group_id',
        'subject id',
    ];

    /**
     * The attributes that should be cast.
     *
     * @var array
     */
    protected $casts = [
        'created_at' => 'datetime',
        'update_at' => 'datetime',
    ];
}
```

Модель даних таблиці “lessons”

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\HasOne;

/**
 * App\Models\Lesson
 *
 * @property int $id
 * @property string $date
 * @property int $group_id
 * @property int $subject_id
 * @property int $lesson_by_count
 * @property \Illuminate\Support\Carbon|null $created_at
 * @property \Illuminate\Support\Carbon|null $updated_at
 * @method static \Illuminate\Database\Eloquent\Builder|Lesson newModelQuery()
 * @method static \Illuminate\Database\Eloquent\Builder|Lesson newQuery()
 * @method static \Illuminate\Database\Eloquent\Builder|Lesson query()
 * @method static \Illuminate\Database\Eloquent\Builder|Lesson whereCreatedAt($value)
 * @method static \Illuminate\Database\Eloquent\Builder|Lesson whereDate($value)
 * @method static \Illuminate\Database\Eloquent\Builder|Lesson whereGroupId($value)
 * @method static \Illuminate\Database\Eloquent\Builder|Lesson whereId($value)
 * @method static \Illuminate\Database\Eloquent\Builder|Lesson whereLessonByCount($value)
 * @method static \Illuminate\Database\Eloquent\Builder|Lesson whereSubjectId($value)
 * @method static \Illuminate\Database\Eloquent\Builder|Lesson whereUpdatedAt($value)
 * @mixin \Eloquent
 * @property Student[] $students
 * @property Subject $subject
 */
class Lesson extends Model
{
    /**
     * The attributes that are mass assignable.
     *
     * @var string[]
     */
    protected $fillable = [
        'date',
        'group_id',
        'subject id',
        'lesson by count'
    ];

    /**
     * The attributes that should be cast.
     *
     * @var array
     */
    protected $casts = [
        'created_at' => 'datetime',
        'update_at' => 'datetime',
    ];

    public function subject(): HasOne
    {
        return $this->hasOne(Subject::class, 'id', 'subject id');
    }
}
```

Модель даних таблиці “lessons”

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\HasOne;

/**
 * App\Models\Lesson
 *
 * @property int $id
 * @property string $date
 * @property int $group_id
 * @property int $subject_id
 * @property int $lesson_by_count
 * @property \Illuminate\Support\Carbon|null $created_at
 * @property \Illuminate\Support\Carbon|null $updated_at
 * @method static \Illuminate\Database\Eloquent\Builder|Lesson newModelQuery()
 * @method static \Illuminate\Database\Eloquent\Builder|Lesson newQuery()
 * @method static \Illuminate\Database\Eloquent\Builder|Lesson query()
 * @method static \Illuminate\Database\Eloquent\Builder|Lesson whereCreatedAt($value)
 * @method static \Illuminate\Database\Eloquent\Builder|Lesson whereDate($value)
 * @method static \Illuminate\Database\Eloquent\Builder|Lesson whereGroupId($value)
 * @method static \Illuminate\Database\Eloquent\Builder|Lesson whereId($value)
 * @method static \Illuminate\Database\Eloquent\Builder|Lesson whereLessonByCount($value)
 * @method static \Illuminate\Database\Eloquent\Builder|Lesson whereSubjectId($value)
 * @method static \Illuminate\Database\Eloquent\Builder|Lesson whereUpdatedAt($value)
 * @mixin \Eloquent
 * @property Student[] $students
 * @property Subject $subject
 */
class Lesson extends Model
{
    /**
     * The attributes that are mass assignable.
     *
     * @var string[]
     */
    protected $fillable = [
        'date',
        'group_id',
        'subject id',
        'lesson by count'
    ];

    /**
     * The attributes that should be cast.
     *
     * @var array
     */
    protected $casts = [
        'created_at' => 'datetime',
        'update_at' => 'datetime',
    ];

    public function subject(): HasOne
    {
        return $this->hasOne(Subject::class, 'id', 'subject id');
    }
}
```

Модель даних таблиці “students”

```php
<?php

namespace App\Models;

use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\HasOne;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Laravel\Sanctum\HasApiTokens;


/**
 * App\Models\Student
 *
 * @property int $id
 * @property int $group_id
 * @property string $first_name
 * @property string $last_name
 * @property int $status
 * @property \Illuminate\Support\Carbon|null $created_at
 * @property \Illuminate\Support\Carbon|null $updated_at
 * @method static \Illuminate\Database\Eloquent\Builder|Student newModelQuery()
 * @method static \Illuminate\Database\Eloquent\Builder|Student newQuery()
 * @method static \Illuminate\Database\Eloquent\Builder|Student query()
 * @method static \Illuminate\Database\Eloquent\Builder|Student whereCreatedAt($value)
 * @method static \Illuminate\Database\Eloquent\Builder|Student whereFirstName($value)
 * @method static \Illuminate\Database\Eloquent\Builder|Student whereGroupId($value)
 * @method static \Illuminate\Database\Eloquent\Builder|Student whereId($value)
 * @method static \Illuminate\Database\Eloquent\Builder|Student whereLastName($value)
 * @method static \Illuminate\Database\Eloquent\Builder|Student whereStatus($value)
 * @method static \Illuminate\Database\Eloquent\Builder|Student whereUpdatedAt($value)
 * @mixin \Eloquent
 * @property Group $group
 */
class Student extends Model
{
    /**
     * The attributes that are mass assignable.
     *
     * @var string[]
     */
    protected $fillable = [
        'first_name',
        'last_name',
    ];

    /**
     * The attributes that should be cast.
     *
     * @var array
     */
    protected $casts = [
        'created at' => 'datetime',
        'update_at' => 'datetime',
    ];

    public function group(): HasOne
    {
        return $this->hasOne(Group::class, 'id', 'group id');
    }

    public function fullName(): string
    {
        return $this->last_name . ' ' . $this->first_name;
    }
}
```

Модель даних таблиці “subjects”

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

/**
 * App\Models\Subject
 *
 * @property int $id
 * @property string $name
 * @property \Illuminate\Support\Carbon|null $created_at
 * @property \Illuminate\Support\Carbon|null $updated_at
 * @method static \Illuminate\Database\Eloquent\Builder|Subject newModelQuery()
 * @method static \Illuminate\Database\Eloquent\Builder|Subject newQuery()
 * @method static \Illuminate\Database\Eloquent\Builder|Subject query()
 * @method static \Illuminate\Database\Eloquent\Builder|Subject whereCreatedAt($value)
 * @method static \Illuminate\Database\Eloquent\Builder|Subject whereId($value)
 * @method static \Illuminate\Database\Eloquent\Builder|Subject whereName($value)
 * @method static \Illuminate\Database\Eloquent\Builder|Subject whereUpdatedAt($value)
 * @mixin \Eloquent
 */
class Subject extends Model
{
    /**
     * The attributes that are mass assignable.
     *
     * @var string[]
     */
    protected $fillable = [
        'name',
    ];

    /**
     * The attributes that should be cast.
     *
     * @var array
     */
    protected $casts = [
        'created_at' => 'datetime',
        'update_at' => 'datetime',
    ];
}
```

Модель даних таблиці “users”

```php
<?php

namespace App\Models;

use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Relations\HasOne;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Laravel\Sanctum\HasApiTokens;

/**
 * App\Models\User
 *
 * @property int $id
 * @property string $email
 * @property string $password
 * @property int $roleId
 * @property int|null $student_id
 * @property int $status
 * @property \Illuminate\Support\Carbon|null $created at
 * @property \Illuminate\Support\Carbon|null $updated_at
 * @property-read
\Illuminate\Notifications\DatabaseNotificationCollection|\Illuminate\Notifications\DatabaseNotificat
ion[] $notifications
 * @property-read int|null $notifications_count
 * @property-read \Illuminate\Database\Eloquent\Collection|\Laravel\Sanctum\PersonalAccessToken[]
$tokens
 * @property-read int|null $tokens_count
 * @method static \Database\Factories\UserFactory factory(...$parameters)
 * @method static \Illuminate\Database\Eloquent\Builder|User newModelQuery()
 * @method static \Illuminate\Database\Eloquent\Builder|User newQuery()
 * @method static \Illuminate\Database\Eloquent\Builder|User query()
 * @method static \Illuminate\Database\Eloquent\Builder|User whereCreatedAt($value)
 * @method static \Illuminate\Database\Eloquent\Builder|User whereEmail($value)
 * @method static \Illuminate\Database\Eloquent\Builder|User whereId($value)
 * @method static \Illuminate\Database\Eloquent\Builder|User wherePassword($value)
 * @method static \Illuminate\Database\Eloquent\Builder|User whereRoleId($value)
 * @method static \Illuminate\Database\Eloquent\Builder|User whereStatus($value)
 * @method static \Illuminate\Database\Eloquent\Builder|User whereStudentId($value)
 * @method static \Illuminate\Database\Eloquent\Builder|User whereUpdatedAt($value)
 * @mixin \Eloquent
 * @property-read Student|null $student
 */
class User extends Authenticatable
{
    use HasApiTokens, HasFactory, Notifiable;

    /**
     * The attributes that are mass assignable.
     *
     * @var string[]
     */
    protected $fillable = [
        'name',
        'email',
        'password',
    ];

    /**
     * The attributes that should be hidden for serialization.
     *
     * @var array
     */
    protected $hidden = [
        'password',
        'remember_token',
    ];

}
```