НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
„КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМ. ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

.

Дипломна робота

на тему: Модуль “Додатковий персонал” системи управління дипломними проектами

Студент групи ТВ-51 Штокал Сергій Сергійович
_____
(прізвище, ім’я, по батькові) (підпис)

Керівник роботи кандидат технічних наук, доцент Коваль О. В. _____
(вчені ступінь та звання, прізвище, ініціали) (підпис)

Кількість сторінок _____

Кількість ілюстрацій _____

Київ – 2019

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено

Завідувач кафедри

_____ О.В. Коваль
(підпис) (ініціали, прізвище)

“___”_____2019р.

ДИПЛОМНА РОБОТА

на здобуття ступеня бакалавра

з напряму підготовки 6.050103 “ Програмна інженерія “

на тему «Модуль “Додатковий персонал” системи управління дипломними проектами»

Виконав : студент 4 курсу, групи ТВ-51

Штокал Сергій Сергійович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник _ кандидат технічних наук, доцент Коваль О. В. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент_____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

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

Студент _____
(підпис)

Київ – 2019

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший , бакалаврський

Напрям підготовки 6.050103 “ Програмна інженерія”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)
”___”_____2019р.

ЗАВДАННЯ
на дипломну роботу студенту
Штокалу Сергію Сергійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Модуль “Додатковий персонал” системи управління дипломними проектами

керівник роботи кандидат технічних наук, доцент Коваль О. В.
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”__”___ 201_р. № __

2. Строк подання студентом роботи _____

3. Вихідні дані до роботи __мова JavaScript, MySQL _платформа Node.js, фреймворки: React,js та Express.js

4.Зміст розрахунково-пояснювальної записки (перелік завдань, які потрібно розробити)___розробити модуль «Додатковий персонал» в якому безпосередньо розробити сторінки для:

1.CRUD функцій студентів, викладачів та працівників додаткового персоналу

2. CRUD функцій графіку виконання, лабораторій

3. Переглядання дипломних робіт та затвердження їх

4. Редагування профілю

5. Перелік ілюстративного матеріалу

1. Загальна проблематика 2. Постановка задачі 3.DFD діаграма 4. USE-CASE діаграма 5. Модель БД 6. Архітектура системи 7. Використані технології 8. Принцип роботи React.js 9. Робота з користувачами, лабораторіями,графіком виконання
10. Формування звіту для дипломних робіт
11.Висноки_____

7. Дата видачі завдання ”_10__”____жовтня_____2018р.

КАЛЕНДАРНИЙ ПЛАН

/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	10.10.2018	
2.	Вивчення та аналіз задачі	14.10.2018-23.12.2018	
3.	Розробка архітектури та загальної структури системи	2.02.2019-3.03.2019	
4.	Розробка структур окремих підсистем	4.03.2019-14.04.2019	
5.	Програмна реалізація системи	18.04.2019-14.05.2019	
6.	Оформлення пояснювальної записки	16.05.2019-10.06.2019	
7.	Захист програмного продукту	15.05.2019	
8.	Передзахист	29.05.2019	
9.	Захист	17.06.2019-22.06.2019	

Студент _____ _____
(підпис) (прізвище та ініціали,)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Бакалаврська дипломна робота присвячена розробці модуля «Додатковий персонал» для системи управління дипломними проектами.

Метою роботи було створення модуля системи управління дипломними проектами для збереження актуальності даних, та отримання звітів в системі.

Під час виконання бакалаврської дипломної роботи була проаналізована предметна область. Проведено бесіди з додатковим персоналом кафедри та реалізовано його побажання в системі. Спроектовано базу даних на основі предметної області. Побудовано архітектуру інтерфейсу користувача та серверу для модуля.

Система є закритою, туди можуть потрапити лише працівники кафедри та студенти що будуть виконувати дипломну роботу.

Записка містить 52 сторінки, 21 рисунків та 16 посилань.

ABSTRACT

The baccalaureate thesis is devoted to the development of the "additional personnel" module for a control system of degree projects.

The creation of the module of a control system of degree projects for maintaining the relevance of data, and obtaining reports in a system was the purpose of work.

During the implementation of the baccalaureate thesis, the subject domain was analyzed. Discussions with additional personnel of the department are led and it is realized its wishes in a system. It is designed the database on the basis of the subject domain. It is constructed the architecture of the interface of the user and the server for the module.

The system is closed, only employees of the department and students can get there that will perform the thesis.

The note contains 52 pages, 21 images and 16 references.

Зміст

Перелік умовних позначень, скорочень і термінів

СКБД — система керування базами даних;

API (англ. Application Programming Interface) — прикладний програмний інтерфейс;

БД — база даних;

CRUD (англ. create read update delete) — 4 базові функції управління даними «створення, зчитування, зміна і видалення»;

SQL (англ. Select query language) — декларативна мова програмування для взаємодії користувача з базами даних;

MS (англ. Microsoft) — багатонаціональна корпорація комп'ютерних технологій.

ПДП — працівник додаткового персоналу.

ВСТУП

Ефективність роботи навчального закладу залежить від ряду факторів, одним з яких є легкий доступ до інформації та зручна комунікація між студентом та викладачем. А як відомо, одним з найважливіших умов успішного функціонування комунікацій між студентом та викладачем є ефективний обмін інформацією. Проте, часто, це затратний процес. Наприклад, розглянемо процес вибору, затвердження і розробки курсових, наукових, дипломних робіт студентами ВНЗ. Щоб обрати наукового керівника та визначитись з темою розробки, яка буде справді цікава і актуальна як для студента так і для викладача часто проходить кілька десятків зустрічей, уточнень, консультацій, що, звісно, займає багато часу. Тому виникає необхідність автоматизувати цей процес, таким чином вдосконалити його.

Саме тому, розробка автоматизованої системи для вибору та управління дипломними роботами для студентів, викладачів ВНЗ є актуальною. А для контролю даних в системі потрібен контент менеджер, роль якого виконує додатковий персонал. Також система допомагає додатковому персоналу формувати графік виконання та звіт.

Метою роботи є підвищення комунікаційних можливостей користувачів за рахунок використання сучасних технологій створення веб додатків, що забезпечує надійний зв'язок користувачів у процесі виконання практичних задач.

Оскільки робота є комплексною, її можна поділити на три модулі – модуль студента, модуль викладача, модуль контент менеджера. Розглянемо третій більш детально.

Головною задачею є створення Web-модуля для додаткового персоналу

У відповідності до поставленої мети потрібно виконати наступні задачі:

- розробити архітектуру системи;
- розробити базу даних;
- розробити серверну частину;

- розробити інтерфейс програмного продукту, який буде легким для розуміння та інтуїтивним користувачу;
- розробити інтерфейс взаємодії між користувачем і основною програмно-апаратною частиною;
- провести тестування програмного продукту для перевірки всіх можливих варіантів використання;
- розробити інструкцію користувача;

Результатом роботи є універсальна та унікальна система що може бути використана у різних кафедрах університетів.

1 Постановка задачі

Мета роботи – це створення модуля «Додатковий персонал» в системі управління дипломними проектами для збереження актуальності даних, та отримання звітів в системі.

1.1 Завдання які потрібно розв’язати

Чинна система управління навчальним процесом у вищих навчальних закладах має ряд суперечностей. У спадок від радянської системи освіти Україна отримала традиційні засоби ведення документообігу у ВНЗ. Ще й зараз у багатьох закладах освіти документи, що відображають хід навчального процесу, який є основним виробничим процесом у навчальному закладі, формуються та зберігаються у вигляді паперових каталогів з різноманітними списками, картками, відомостями, звітами тощо. Такий спосіб організації управління негативно позначається на продуктивності праці в організаційних підрозділах ВНЗ, створює несприятливі умови для ефективної роботи як викладачів, так і студентів та персоналу.

Щоб вирішити цю суперечність, дедалі ширшого застосування набувають комп’ютеризовані інформаційні системи, які дають змогу автоматизувати та впорядкувати цю сферу діяльності, раціоналізувати інформаційні потоки у ній та розвантажити персонал закладів освіти від одноманітної рутинної роботи та спростити комунікацію між викладачами, студентами та персоналом.

Одним з найважливіших умов успішного функціонування комунікацій між студентом та викладачем є ефективний обмін інформацією. Саме тому було прийнято рішення створити програмне забезпечення для вибору тем дипломних робіт студентами ВНЗ.

Було визначено наступні загальні задачі, які розв’язуються системою:

- можливість додавання нових тем дипломних робіт для викладачів;
- вибір теми для дипломної роботи студентами ВНЗ;
- взаємодія наукового керівника зі студентом;
- виведення звітності за списками груп або про конкретного студента;
- надійне збереження інформації.

Програмний продукт складається з трьох модулів, кожен з яких призначений певній категорії користувачів. Розглянемо окремо задачі, які виконує програмний модуль, що призначений для контент менеджера. Цей модуль створюється з метою, надання персоналу навчального закладу керувати конкретними етапами навчального процесу. Отже, маємо такі завдання для модуля «Додатковий персонал»

- CRUD функції для інформацій про студентів, викладачів, персоналу кафедри;
- CRUD функції для користувачів в системі;
- CRUD функції для лабораторій;
    - Додавання та видалення викладачів з лабораторії;
    - CRUD функції для під лабораторій в лабораторії;
- CRUD функції для академічних груп;
- Додавання тем на вимогу наукових керівників;
- Створення та редагування графіка виконання дипломних робіт;
- Затвердження тем дипломних робіт студентів;
- Отримання інформації про затвердженні роботи студентів;
- Формування звіту з затвердженими роботами студентів;

Програмний продукт має ще інші два модулі . Перший призначений для студентів, другий - для викладачів чи наукових керівників.

1.2 Вимоги до системи

Система повинна бути закритою для інших користувачів крім:

- Викладачів кафедри;
- Додаткового персоналу кафедри;
- Студентів які будуть писати дипломний проект в даному навчальному році.

Вимоги доступу в систему виконані за допомогою сторінки входу в профіль користувача. При додаванні інформації про нового студента, викладача, чи додаткового персоналу в систему для цього нового користувача створюється профіль і генерується початковий логін і пароль. Після входу в профіль за допомогою логіну і паролю, користувач зв’язує профіль зі своєю електронною поштою та задає свій пароль. Після цього логін і пароль, що використовувались для входу блокуються, а замість них вхід відбувається за допомогою адреси електронної пошти та паролю що створив сам користувач.

2 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

Сучасний університет характеризується активним використанням інформаційних технологій в освітньому процесі, науковій діяльності. Зараз вже існує ряд програмних додатків, що призначені для організації та підтримки навчального процесу в вищих навчальних закладах . Основною метою таких додатків є скоротити час, що витрачають працівники вищих навчальних закладів на розв`язання повсякденних задач та спростити процедуру роботи з даними, а також сприяти полегшенню комунікації студентів та викладачів.

2.1. Аналіз Аналогів

Наразі не існує, програм, які б поєднували студентів та їх наукових керівників і надавали б можливість за їх допомогою обирати теми дипломних робіт.

Проте є системи які мають схожий функціонал та вирішують схожі проблеми. Розглянемо більш детально деякі з них.

Пакет програм «Деканат» - це автоматизована система управління вищим навчальним закладом. Пакет побудований за клієнт-серверною технологією, що дозволяє встановлювати його на множину комп`ютерів, які об`єднані в локальну мережу та працюють з єдиною базою даних.

Скріншот головного вікна програми зображено на рисунку 2.1.

Використання додаткових web-сценаріїв забезпечує можливість доступу до бази даних в межах окремих програм Пакету з всесвітньої павутини Інтернет.

В ролі сервера управління базами даних використовується FireBird.

До складу пакету додатково входить програма "ПС-Адміністратор", яка призначена для щоденного тестування, резервного копіювання та, при необхідності, відновлення бази даних. Ця програма позбавляє вищий навчальний заклад від

необхідності додатково наймати на роботу фахівця, що відповідає за обслуговування систем управління базами даних.

Недоліком такого продукту є складність інтерфейсу та нагромадженість системи, яка займає багато місця на диску, а також складається з підпрограм, які потребують обов’язкового встановлення.

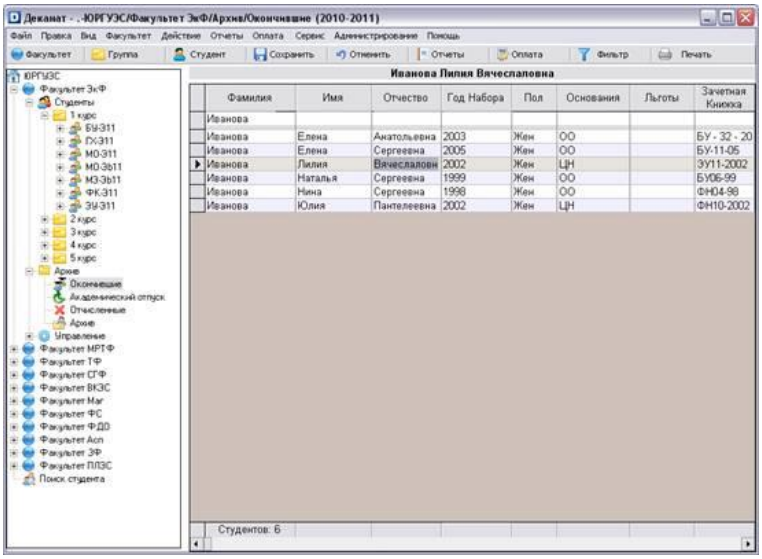


Рисунок 2.1 – Головне вікно програми «Деканат»

Також було розглянуто програмний продукт «Галактика Управління Вузом». Продукт призначений для ведення всього процесу управління як навчальним процесом освітньої установи, так і його фінансово-господарською діяльністю.

Скріншот головного вікна програми зображено на рисунку 2.2.

Система використовується для:

- формування навчальних планів згідно освітніх стандартів;
- розрахунку навантаження кафедр і розподілу її по викладачах;
- по заданим формулам можна врахувати нюанси розрахунку норм часу по різним видам робіт для різних академічних предметів, форм навчання, типів групування студентів;
- планування штату викладачів кафедр;

- складання розкладу занять, враховуючи графік навчального процесу, навантаження викладачів, наявність, місткість і оснащеність навчальних приміщень.

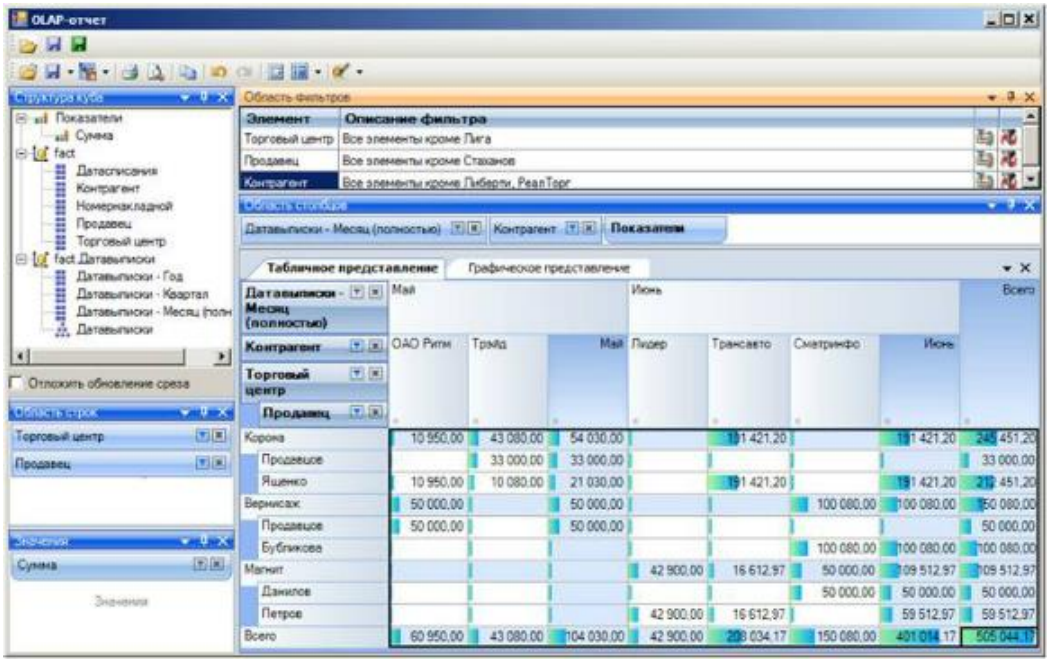


Рисунок 2.2 – Головне вікно програми «Галактика Управління Вузом»

Таким чином, додаток «Галактика Управління Вузом» має широкий функціонал, проте недоліком додатку є те, що він призначений суто для працівників деканату, та не забезпечує прямої взаємодії студентів та їх наукових керівників.

Ще одним прикладом додатку, що полегшує організацію навчального процесу є додаток МКР. Це мобільний додаток, що також має версію і для настільного ПК.

Додаток призначений для складання розкладу занять, та інформування студентів про теми лекцій, та для контролю відвідуваності. Скріншот головного вікна програми зображено на рисунку 2.3.

Додаток працює в режимі он-лайн, тобто всі зміни в розкладі відразу відображаються в додатку. Реалізована функція запам'ятовування особистого і останнього відкритого розкладу (якщо інтернет відсутній, то розклад все одно відображається).У додатку присутні кілька видів відображення - списковий та

календарний. Головним недоліком додатку є відсутність можливості створювати звіти за певними запитами.

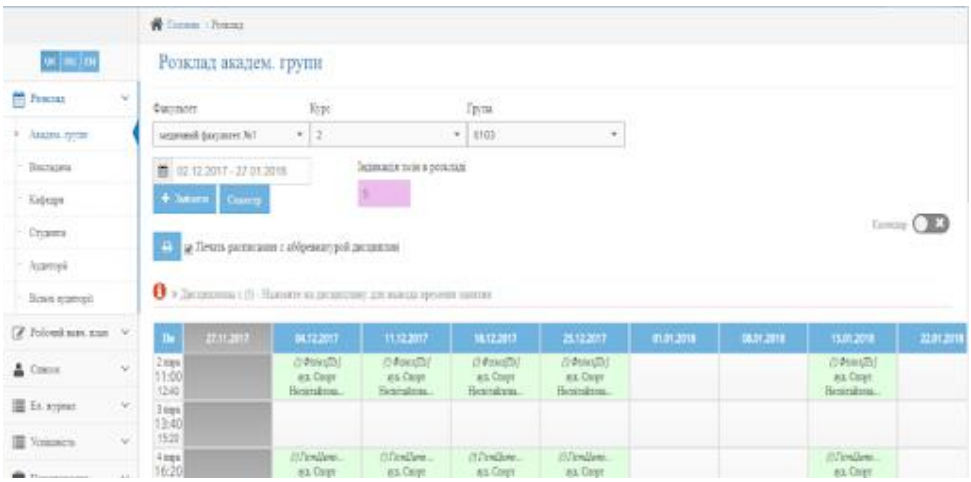


Рисунок 2.3 – Головне вікно програми «Галактика Управління Вузом»

Додаток працює в режимі онлайн, тобто всі зміни в розкладі відразу відображаються в додатку. Реалізована функція запам'ятовування особистого і останнього відкритого розкладу (якщо інтернет відсутній, то розклад все одно відображається).У додатку присутні кілька видів відображення - списковий та календарний. Головним недоліком додатку є відсутність можливості створювати звіти за певними запитами.

Таким чином, проаналізувавши наявні аналоги було прийнято рішення розробити власний програмний продукт, який би врахував вищеописані недоліки.

У цьому розділі було проаналізовано існуючі аналогічні програмні системи на базі практики і взагалі. Було визначено їх недоліки, такі як складність інтерфейсу та громісткість системи , відсутність можливості керувати системою через акаунт контент менеджера, відсутність можливості формувати звітність. Всі ці недоліки будуть враховані при розробці власного додатку, а також буде додано новий функціонал.

3 ОПИС ПРЕДМЕТНОЇ МОДЕЛІ

Тема моєї дипломної роботи «Модуль «Додатковий персонал» системи управління дипломними проектами». Даний модуль буде забезпечувати актуальним контентом систему. Для того щоб сприйняти та описати предметну модель модулю потрібно спочатку описати модель всієї системи, а потім вже перейти локально до модулю «Додатковий персонал». Було проведено консультації з додатковим персоналом кафедри і отримано інформацію про функціонал який повинен бути в системі для модулю.

3.1 Функціонал

Дана система дає змогу викладачам:

- виставляти список тем для дипломних робіт;
- записуватись в лабораторію та виходити з неї;
- приймати та відмовляти заявки студентів на тему дипломної роботи;
- переглядати графік виконання дипломних робіт;
- виставляти виконання пунктів графіка виконання дипломних робіт для студентів;
- входити в профіль;
- редагувати свій профіль;

Студентам:

- переглядати список всіх можливих тем дипломних робіт;
- надсилати заявку викладачам з темою дипломної роботи;
- переглядати графік виконання дипломних робіт;
- входити в профіль;
- редагувати свій профіль;

Додатковому персоналу:

- здійснювати CRUD функції для студентів;

- здійснювати CRUD функції для викладачів;
- здійснювати CRUD функції для лабораторій;
- здійснювати CRUD функції для академічних груп;
- Створювати та редагувати графік виконання дипломних робіт;
- Затверджувати теми дипломних робіт студентів;
- Отримувати інформацію про затвердженні роботи студентів;
- Формувати звіт з затвердженими роботами студентів;

Виділимо основні актори: студент, викладач, додатковий персонал.

Побудуємо діаграму прецедентів (use case) на основі функцій які можуть виконувати актори (рисунок 3.1)

З діаграми ми бачимо, що основні базові функції для всіх користувачів в системі такі:

- **Взаємодія з графіком виконання**
- **Вхід в систему**
- **Взаємодія з налаштуваннями профілю**
- **Взаємодія з дипломними роботами**
- **Взаємодія з лабораторіями**

І окремо для додаткового персоналу є ще функція **роботи з користувачами системи.** Пройдемось по цим основним функціям для кожного користувача:

3.1.1 Взаємодія з графіком виконання

Для початку потрібно створити графік виконання - цим буде займатись додатковий персонал. Тому потрібно створити сторінку з додаванням та редагуванням графіку виконання для груп студентів з різних освітніх ступенів. Графік виконання складається з пунктів. Один пункт складається же з роботи яку потрібно виконати (перелік робіт) ,початкова та кінцева дати , і також певні додаткові примітки, в яких наприклад може бути список певних звітних документів.

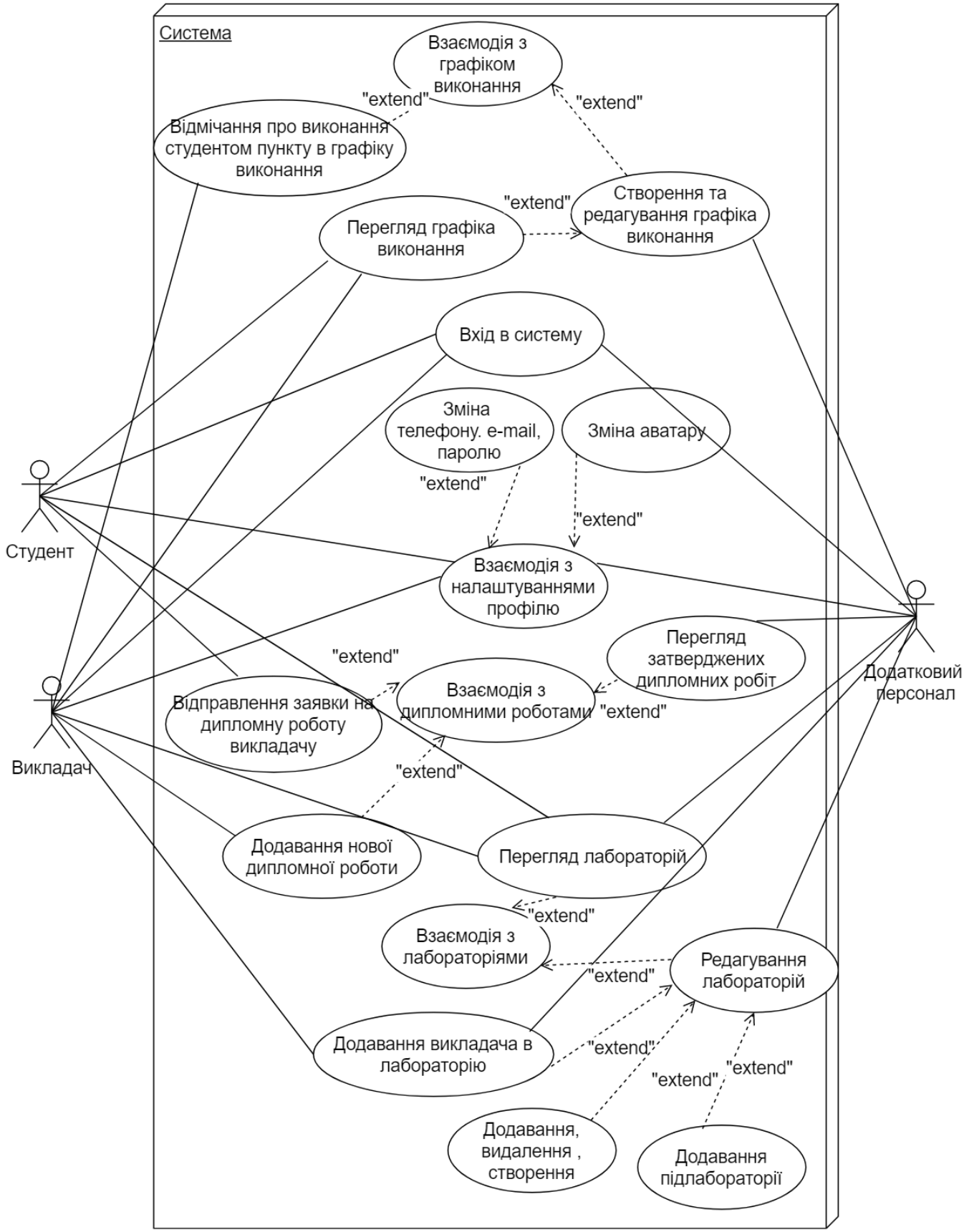


Рисунок 3.1 – Діаграма прецедентів(use-case)

Отже графік виконання з прикладом пункту буде виглядати як на зразку (табл. 3.1).

Перелік робіт	**Початок виконання**	**Кінець виконання**	**Звітні документи**
Навчання за розкладом	31.08.2019	28.12.2019	Залік з кожної дисципліни в заліковій книжці

Таблиця 3.1 – Частина графіка виконання

3.1.2 Вхід в систему

Щоб користувач зайшов до системи йому потрібно знати данні для входу. Тому на обговорені проекту з додатковим персоналом було вирішено що файли з даними для входу роздають кожній групі, викладачам та іншим користувачам системи які мають роль додатковий персонал.

Щоб користувачі отримали остаточні данні для входу потрібно пройти наступний порядок дій в системі:

1. ПДП додає данні про студента, викладача чи ПДП. Система при додаванні цих даних в БД генерує для студента, викладач чи ПДП username і password (логін та пароль).
2. ПДП надсилає запис системі створити звітний файл в якому будуть тимчасові данні для входу в систему (логін та пароль). Завантажується файл. Далі ПДП роздає цей файл користувачам.
3. Користувач заходить по логіну та паролю в систему. Система його просить ввести його email.
4. Користувач вводить свій email та підтверджує його. Тепер система видаляє його логін. Надалі вхід буде відбуватись за email-ом та паролем користувача.

   Увесь цей потік даних з точки зору додаткового персоналу можна побачити у діаграмі DFD на рисунках 3.2, 3.3 ,3.4

Рисунок 3.2 Загальний вигляд діаграми DFD для отримання логіну та паролю користувачем

На рисунку 3.2 є тільки один процес-це «Операції в системі» в який поступає інформація про студента, викладача, ПДП з зовнішніх баз даних (інформація деканату тощо).

На рисунку 3.3 є процеси «Занесення інформації про студента/викладача/ПДП в систему» в які поступає зовнішня інформація. Також є процес генерації логіну та паролю на основі прізвища користувача. Після цього процесу інформація про користувача, згенерований логін та пароль записуються в єдину БД.

Далі інформація з БД поступає в процес Генерації файлу з тимчасовими даними для входу (логін, пароль, та роль користувача).

На рисунку 3.4 Ми бачимо декомпозицію процесу «Реєстрація email» з рисунку 3.3. Логін та пароль передається користувачу. Далі відбувається процес авторизації. Користувач передає туди свої тимчасові данні для входу (логін та пароль)

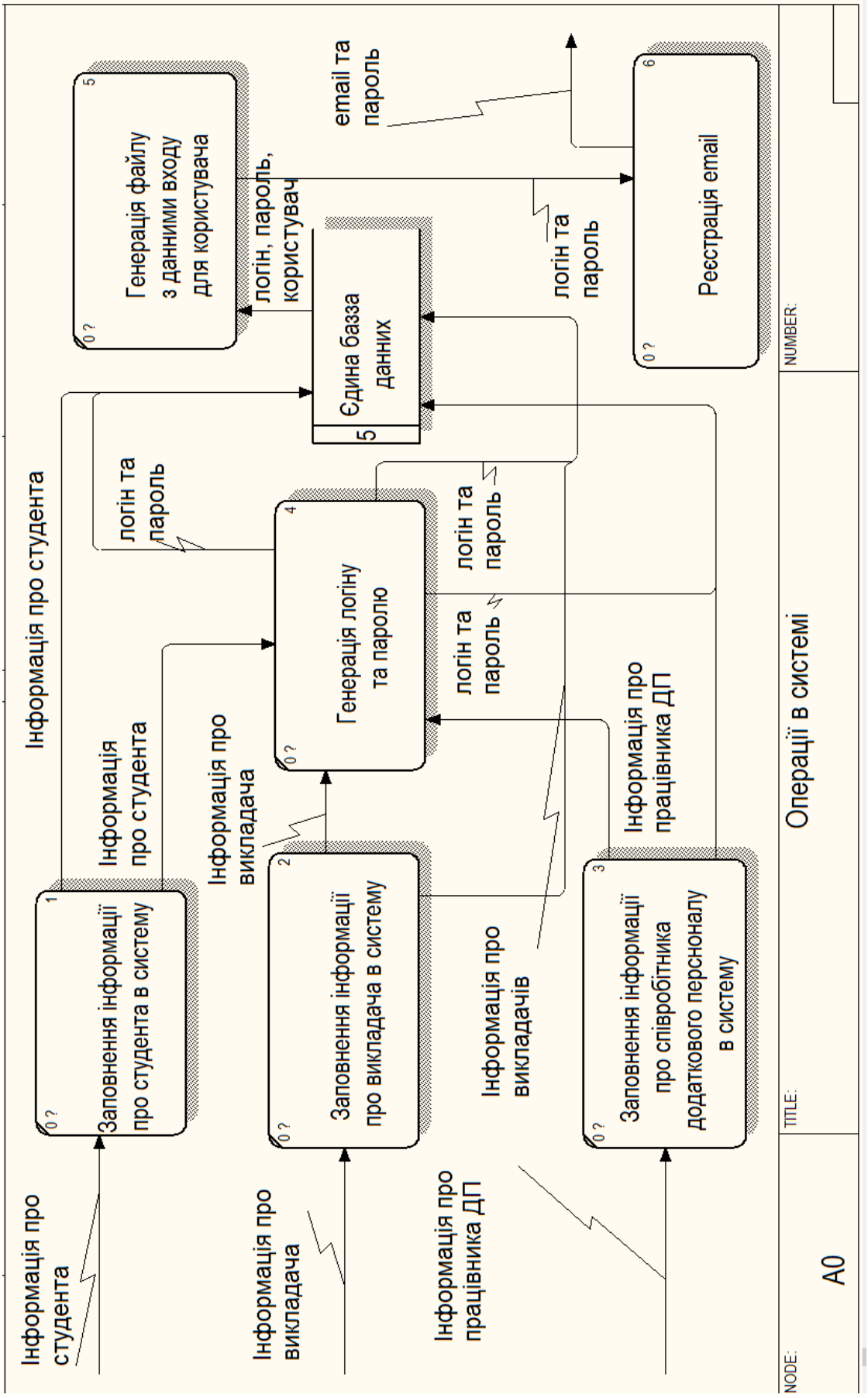


Рисунок 3.3 Декомпозиція процесу «Операції в системі» DFD діаграми.

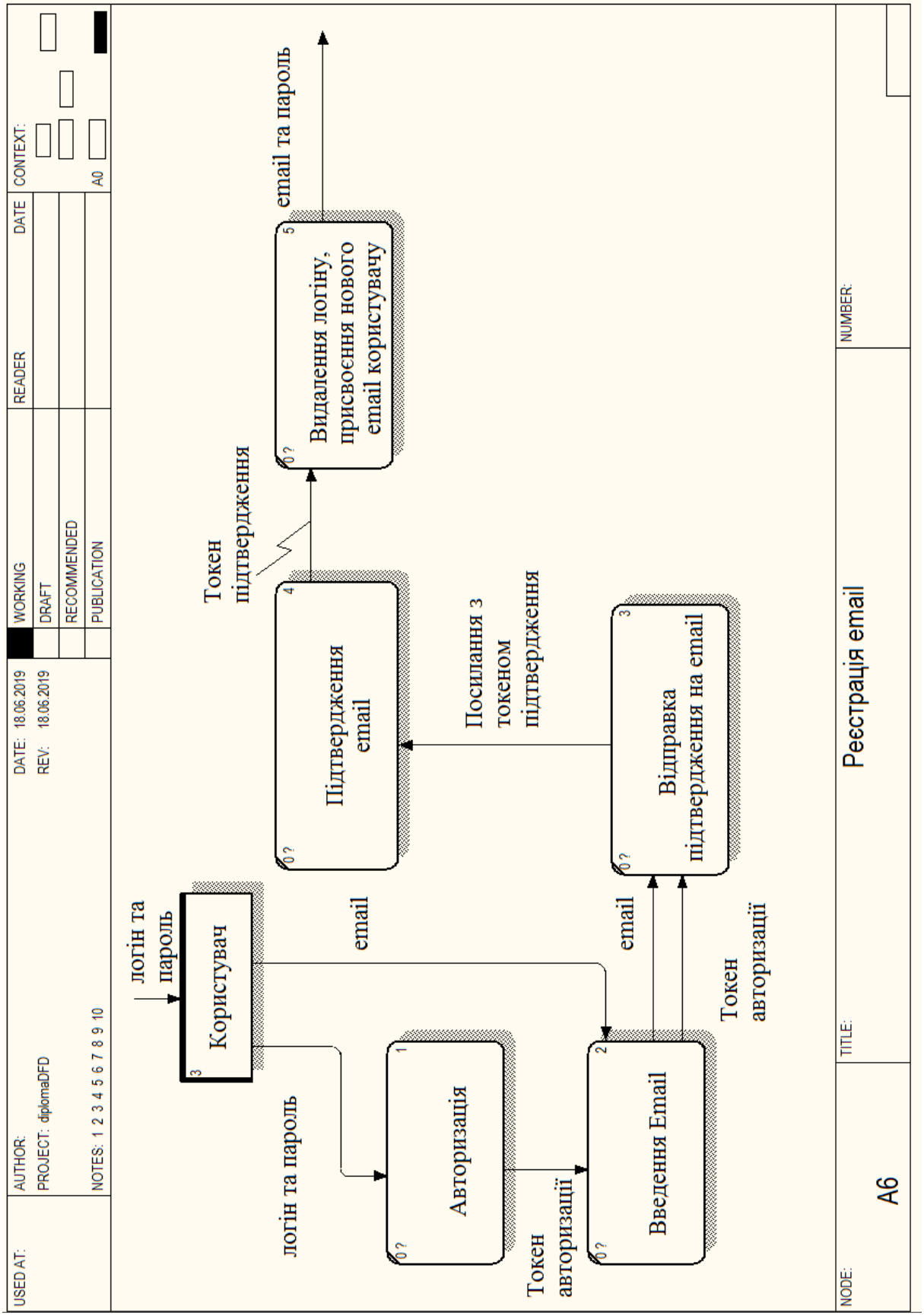


Рисунок 3.4 Декомпозиція процесу «Реєстрація email» DFD діаграми.

Після авторизації користувачу присвоюється токен авторизації який використовується для перебування користувача в системі. Далі відбувається процес «Введення Email». Система просить користувача ввести свій email для подальшого користування системою. Далі система відправляє токен підтвердження на email (процес «Відправка підтвердження на email»). В процесі «Підтвердження email» користувач натискає на посилання, та автоматично надсилає всю необхідну інформацію системі для підтвердження (токен підтвердження).

Далі в процесі «Видалення логіну та присвоєння email користувачу» система видаляє логін та присвоює підтверджений користувачем email цьому ж користувачу.

Отримуємо email та пароль для входу що є постійними даними для входу.

3.1.3 Взаємодія з дипломними роботами

ПДП повинен мати змогу переглядати дипломні роботи всіх студентів. Та завантажувати звіт за зразком, що наданий додатковим персоналом.

Також завідувач кафедри зі свого акаунту повинен мати змогу відмічати дипломні роботи як затверджені чи не затверджені ним. І це тільки має змогу робити завідувач кафедри інші ПДП мають змогу тільки переглядати.

Табличка дипломних робіт повинна мати такий вигляд табл. 3.2

Студент	Група	Викладач	Тема роботи	Лабораторія	Напрям дабораторії
Войтович Сергій Вікторович,	ТР-41	Гурін Артем Леонідович	Серверна частина порталу для обміну інформацією кафедри на базі ASP.NET Проект: Система трекінгу працівників підприємства	Навчально-наукова лабораторія аналізу великих обсягів даних та управління проектами	Розробка та впровадження автоматизованої системи контролю за виконанням доручень, учбових планів та наукових розробок

Таблиця 3.2 Приклад вигляду таблиці «Дипломні роботи» для ПДП

3.1.4 Взаємодія з лабораторіями

ПДП повинен бачити список всіх назв лабораторій. Повинен мати змогу редагувати назви лабораторій, та видаляти самі лабораторії.

При відкритті певної лабораторії там повинна бути інформація прос склад лабораторії (викладачі що записані за лабораторією) та напрямки лабораторії. ПДП повинен мати змогу змінювати склад лабораторії (видаляти,додавати викладачів). Також у ПДП повинні бути CRUD функції для напрямків лабораторії

3.1.5 Взаємодія з налаштуваннями профілю

Кожен користувач повинен мати змогу змінювати свій email, телефон та пароль у себе на сторінці профілю.

3.1.6 Робота з користувачами системи

ПДП має змогу переглядати всіх студентів,викладачів та таких же ПДП як він сам. Також має всі CRUD функції над користувачами в системі. Як було описано в пункті **3.1.2.** система сама генерує логін та пароль для користувача для входу в систему. Додатковому персоналу залишається лише завантажити звіт (документ) з даними для входу та роздати їх всім новим користувачам.

Отже, загальний потік даних в системі з токи зору додаткового персоналу можна зобразити на DFD діаграмі (рисунок 3.5)

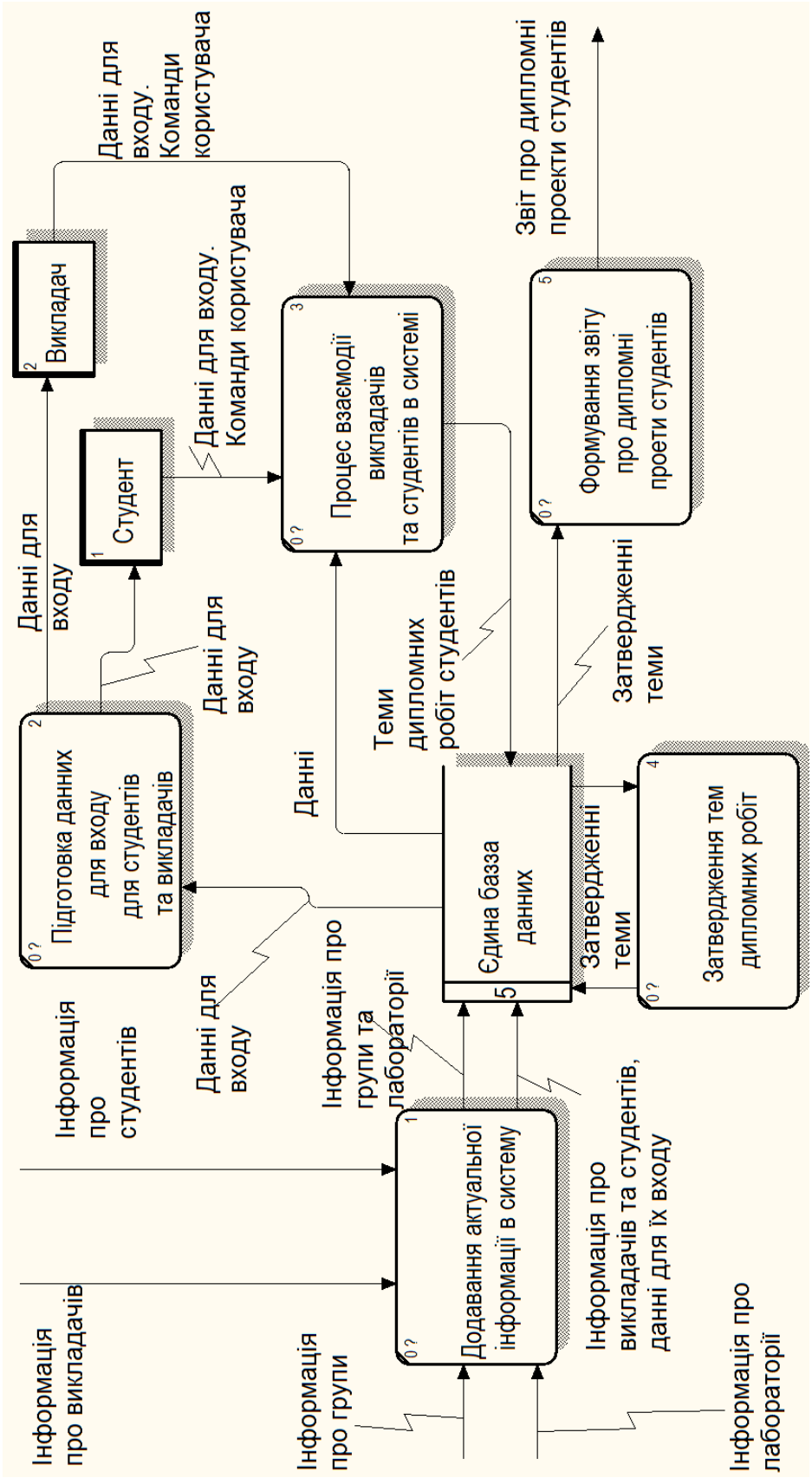


Рисунок 3.5- Загальний потік даних в модулі «Додатковий персонал»

3. ЗАСОБИ РОЗРОБКИ

Основним середовищем розробки було середовище JetBrains WebShtorm.

Для створення бази даних, написання правильних запитів та для зручного її контролю використовувався phpMyAdmin, що є веб-застосунком з відкритим кодом на мові PHP із графічним веб-інтерфейсом для адміністрування бази даних MySQL.

Для розробки серверної частини веб-застосунку використовувалась платформа Node.js з розширенням RESTful API. Це платформа з відкритим кодом для виконання високопродуктивних мережевих застосунків, написаних мовою JavaScript.

Для розробки безпосередньо порталу використовувались мови розмітки HTML5, CSS3, та мова програмування JSX, зокрема надбудови та бібліотеки , React.js.

3.1. Середовище розробки JetBrains WebStorm

JetBrains WebStorm — інтегроване середовище розробки для JavaScript, HTML та CSS від компанії JetBrains, розроблена на основі платформи IntelliJ IDEA. WebStorm є спеціалізованою версією PhpStorm, пропонуючи підмножину з його можливостей. WebStorm постачається з перед-установленим плаґінами JavaScript (такими як для Node.js), котрі доступні для PhpStorm безоплатно.

WebStorm підтримує мови JavaScript, CoffeeScript, TypeScript та Dart..

Особливості середовища JetBrains WebStorm:

3.1.1 Інтелектуальна допомога в написанні коду

WebStorm допомагає писати чудовий код. Його інтелектуальний редактор з розширенням «кодом завершення», на льоту аналізує код, форматує код і рефакторить його. Це підвищує продуктивність і підіймає досвід розвитку на абсолютно новий рівень.

Підтримувані мови та фреймворки:

Середовище WebStorm надає першокласну допомогу для кодування JavaScript, ECMAScript6, TypeScript, CoffeeScript, Dart і Flow.

Також середовище допомагає написати код HTML, CSS, Less, Sass і Stylus.

Крім того, WebShtorm надає передову підтримку Node.js і популярних фреймворків, таких як React, Angular, Vue.js, Meteor і багато інших.

Завершення коду

Середовище WebStorm аналізує ваш проект, щоб забезпечити найкращі результати завершення коду для всіх методів, функцій, модулів, змінних і класів, визначених у вашій програмі. Допомога у кодуванні є контекстною і може також бути специфічною для конкретної галузі.

При роботі з CSS можна користуватись завершенням коду для властивостей та їх значень. У Less та Sass можна отримати допомогу для mixins. І, звичайно, в HTML ви можете отримати код завершення для всіх тегів і атрибутів.

Аналіз якості коду

Середовище WebStorm має сотні вбудованих перевірок, що охоплюють всі підтримувані мови. Крім того, ви можете використовувати ESLint, TSLint, Stylelint, JSCS, JSHint і JSLint.

Всі помилки та попередження повідомляються безпосередньо в редакторі під час введення тексту.

Будь-який рядок коду з можливим показом позначок у правому каналі редактора, так що ви можете легко виявити помилки та попередження у довгому файлі.

Ви також можете запустити аналіз якості коду для всього проекту і автоматично застосувати вибрані швидкі виправлення.

Emmet - найважливіший інструментарій для веб-розробників

У WebStorm ви можете використовувати повну силу скорочень Emmet для підвищення продуктивності. Введіть абревіатуру в HTML, потім натисніть Tab, щоб розгорнути її в розмітку. Emmet також працює в CSS і JSX.

Стиль коду

Можна використовувати відповідний стиль коду, за допомогою WebStorm є можливість автоматично застосовувати налаштований стиль коду під час написання коду або переформатування цілих файлів одночасно.

Стиль коду налаштовується для будь-якої мови, включаючи відступи, пробіли, правила вирівнювання тощо. Схема стилю коду зберігається в налаштуваннях проекту і, якщо хочете, нею можна поділитися з вашими товаришами по команді через VCS.

Навігація

Завдяки потужним навігаційним можливостям WebStorm ви зможете ефективно обійти код і заощадити час при роботі з великими проектами.

Для будь-якого методу, функції або змінної у вашому коді переходьте до її визначення за допомогою простого Ctrl + Click або шукайте його використання.

Шукайте по всьому проекту символ, файл або ім'я класу, використовуючи діалогове вікно Пошук скрізь (Double Shift).

Вигляд структури може допомогти легко переміщатися по відкритому файлу.

Редагування в прямому ефірі

Редагування в прямому ефірі дозволяє одразу переглядати оновлення вмісту сторінки у веб-переглядачі (лише в Google Chrome), не перезавантажуючись, із внесенням змін у файли HTML і CSS. Він працює як частина сеансу налагодження JavaScript.

3.1.2 Налагодження, відстеження та тестування

Середовище WebStorm надає потужні вбудовані інструменти для полегшення налагодження, тестування та відстеження. Якщо ви працюєте з клієнтського або Node.js застосунку, WebStorm може допомогти у виконанні цих важливих завдань.

Налагодження клієнтського JavaScript і Node.js

Середовище WebStorm надає розширений відладчик для коду на стороні клієнта, який працює разом з Google Chrome. Він побудований безпосередньо в середовищі IDE, тому не потрібно перемикатися між редактором і браузером для

налагодження.

Можна легко налагоджувати код ECMAScript 6, TypeScript або CoffeeScript, покладаючись на підтримку відладчика WebStorm на картах джерел.

Повнофункціональний вбудований відладчик для Node.js також працює прямо з коробки. Використовуйте його для налагодження програм, запущених локально або на віддаленому комп'ютері.

Відладчик WebStorm має декілька переглядів, включаючи фрейми, глобальні та локальні змінні та спостерігачі. Змінні значення відображаються вбудовано, поряд з їх використанням в редакторі. Ви можете легко оцінювати вирази JavaScript у середовищі виконання. Точки зупинки підтримують режим призупинення та умови.

Трасування

Розширення spy-js - це вбудований інструмент, який допомагає вам відстежувати код і ефективно визначати можливі вузькі місця. Він працює як на стороні клієнта JavaScript і Node.js, і навіть підтримує мови, створені на JavaScript.

За допомогою spy-js можна побачити повний список подій, які ініціювали виконання коду, а потім викопати трасування стека для події і дослідити виділений слід у вихідному коді. Ці дані також використовуються для отримання результатів завершення коду.

Розширення spy-js також може допомогти візуалізувати структуру програми. За допомогою діаграми spy-js, можна побачити, як файли проекту пов'язані з викликами функцій на основі даних про час виконання.

Тестування Unit

Модульне тестування відбувається з легкістю, оскільки WebStorm інтегрується з популярними фреймворками тестування JavaScript.

Можна вибрати Karma або Jest для тестування JavaScript-коду на стороні клієнта або Mocha для тестування Node.js. Виконуйте та налагоджуйте тести безпосередньо в середовищі IDE, переглядайте результати у візуальному форматі та перейдіть до тестового коду.

Звіти про покриття коду також доступні для пускача Карма.

Середовище WebStorm також підтримує Protractor для кінцевого тестування Angular, JSTestDriver, Cucumber.js та Nodeunit.

3.1.3 Функції IDE

Абривіатура VCS- система контролю версій

Середовище розробки WebStorm надає уніфікований інтерфейс користувача для роботи з багатьма популярними системами керування версіями, забезпечуючи послідовний досвід користувачів у git, SVN, Mercurial та Perforce.

Будь-які неприйняті зміни виділяються в лівому жолобі редактора й у вікні "Проект". Можна легко відхилити будь-яку зміну лише за два кліки.

Вбудований інструмент візуального злиття вирішує всі конфлікти швидким і інтуїтивним способом.

Під час роботи з GitHub можна перевірити проекти та зробити запит на вилучення (pull request) безпосередньо в IDE.

Локальна історія

Незалежно від того, чи використовуєте ви VCS, локальна історія може бути реальною заставкою коду. WebStorm відстежує будь-які зміни, зроблені у ваших вихідних файлах, захищаючи вас від будь-яких випадкових втрат або модифікацій, навіть якщо вони зроблені іншими програмами. У будь-який час можна перевірити історію або певного файлу, або каталогу, і повернути його до будь-якої з попередніх версій.

Кастомізація

Середовище розробки надзвичайно кастомізоване. Налаштуйте її так, щоб вона відмінно відповідала вашому стилю кодування, від ярликів і візуальних тем до вікон інструментів і макета редактора.

Редактор WebStorm має світлі та темні види, з яких можна вибрати. Колірні схеми можна налаштувати в меню Налаштування на мові, або ви можете знайти й використовувати одну з популярних тем, доступних в Інтернеті.

3.2 Веб застосунок phpMyAdmin

Застосунок phpMyAdmin є вільним програмним засобом, написаним на PHP, призначеним для керування адміністрацією MySQL через Інтернет. phpMyAdmin підтримує широкий спектр операцій на MySQL і MariaDB. Часто використовувані операції (керування базами даних, таблицями, стовпцями, відносинами, індексами, користувачами, дозволами тощо) можна виконувати за допомогою інтерфейсу користувача, в той час як ви все ще маєте можливість безпосередньо виконувати будь-який оператор SQL.

Застосунок phpMyAdmin постачається з широким діапазоном документації, і користувачі можуть оновлювати наші вікі-сторінки для обміну ідеями та вимогами до різних операцій.

Застосунок phpMyAdmin також дуже глибоко задокументований у книзі, написаній одним з розробників - Mastering phpMyAdmin для ефективного управління MySQL, який доступний англійською та іспанською мовами.

Щоб полегшити використання широкому колу людей, phpMyAdmin перекладається на 72 мови і підтримує мови LTR і RTL.

Features

- Інтуїтивно зрозумілий веб-інтерфейс
- Підтримка більшості функцій MySQL:
    - переглядати та видаляти бази даних, таблиці, види, поля та індекси
    - переглядати та видаляти бази даних, таблиці, види, поля та індекси
    - обслуговування серверів, баз даних і таблиць з пропозиціями про конфігурацію сервера
    - виконання, редагування та закладки будь-яких SQL-операторів, навіть пакетних запитів
    - керувати збереженими процедурами і тригерами
- Імпортувати дані з CSV і SQL

-
- Експортувати дані в різні формати: CSV, SQL, XML, PDF, ISO / IEC 26300 - Текст і таблицю OpenDocument, Word, LATEX та інші
- Адміністрування декількох серверів
- Створення графіки макета бази даних у різних форматах
- Створення складних запитів за допомогою Query-by-example (QBE)
- Пошук по всьому світу в базі даних або його підмножині
- перетворення збережених даних у будь-який формат за допомогою набору попередньо визначених функцій, наприклад відображення BLOB-даних як зображення або посилання для завантаження;

3.3 Середовище Node.js

Середовище Node.js є крос-платформним середовищем виконання з відкритим вихідним кодом, яке виконує код JavaScript поза браузером. Node.js дозволяє розробникам використовувати JavaScript для написання інструментів командного рядка і для сценаріїв на стороні сервера - запуск скриптів на стороні сервера для створення динамічного вмісту веб-сторінки до того, як сторінка буде надіслана веб-браузеру користувача. Отже, Node.js представляє парадигму "скрізь JavaScript", що об'єднує розробку веб-додатків навколо однієї мови програмування, а не на різних мовах серверних і клієнтських скриптів.

Хоча .js є стандартним розширенням файлу для коду JavaScript, ім'я "Node.js" не посилається на конкретний файл у цьому контексті і є лише назвою продукту. Node.js має керовану подіями архітектуру, здатну до асинхронного вводу-виводу. Ці варіанти проектування мають на меті оптимізувати пропускну здатність і масштабованість у веб-додатках з безліччю операцій введення / виводу, а також для веб-додатків у реальному часі (наприклад, програми реального часу та браузерні ігри).

3.3.1 Цикл подій Threading

Середовище Node.js працює на циклі подій одного потоку, використовуючи неблокуючі виклики вводу-виводу, дозволяючи йому підтримувати десятки тисяч одночасних з'єднань без перенесення витрат на перемикання контекстних потоків. Конструкція спільного використання одного потоку серед усіх запитів, які використовують шаблон спостерігача, призначена для побудови високо паралельних додатків, де будь-яка функція, що виконує ввід / вивід, повинна використовувати зворотний виклик. Щоб розмістити однопоточний цикл подій, Node.js використовує бібліотеку libuv, яка, у свою чергу, використовує пул потоків фіксованого розміру, який обробляє деякі неблокуючі асинхронні операції вводу-виводу.

Пул потоків обробляє виконання паралельних завдань у Node.js. Основний виклик функції потоку повідомляє завдання в спільну чергу завдань, потоки якої в пулі потоків тягнуть і виконують. По суті, неблокуючі системні функції, такі як мережа, переводять на неблокуючі сокети на стороні ядра, при цьому заблокуючи функції системи, такі як файловий ввід / вивід, запускають блокуючим способом на власних потоках. Коли потік у пулі потоків завершує завдання, він інформує головний потік цього, який, у свою чергу, прокидається і виконує зареєстрований зворотний виклик.

Недоліком цього однопоточного підходу є те, що Node.js не дозволяє здійснювати вертикальне масштабування, збільшуючи кількість ядер процесора машини, на якому він працює, без використання додаткового модуля, наприклад кластера, або pm2. Проте розробники можуть збільшити кількість потоків за замовчуванням у пулі потоку libuv. Серверна операційна система (ОС), ймовірно, буде поширювати ці потоки в декількох ядрах. Інша проблема полягає в тому, що довготривалі обчислення та інші завдання, пов'язані з процесором, заморожують весь цикл подій до завершення.

3.3.2 Движок V8

Движок V8 - движок виконання JavaScript, спочатку створений для Google

Chrome. Потім вона була відкрита Google у 2008 році. Написана на мові C ++, V8 компілює вихідний код JavaScript на рідний машинний код, а не інтерпретує його в реальному часі.

Средеовище Node.js використовує libuv для обробки асинхронних подій. Libuv - це рівень абстракції для функціональності мережевих і файлових систем як на Windows, так і на системах на основі POSIX, таких як Linux, macOS, OSS на NonStop і Unix.

Основна функціональність Node.js знаходиться в бібліотеці JavaScript. Прив'язки Node.js, написані на C ++, з'єднують ці технології один з одним і з операційною системою.

3.3.3 Управління пакетами

Менеджер npm - це попередньо встановлений менеджер пакетів для серверної платформи Node.js. Він встановлює програми Node.js з реєстру npm, організовуючи встановлення та керування сторонніми програмами Node.js. Пакети в реєстрі npm можуть варіюватися від простих допоміжних бібліотек, таких як Lodash, до бігунів завдань, таких як Grunt.

3.3.4 Єдиний API

Средеовище Node.js можна комбінувати з браузером, базою даних, яка підтримує дані JSON (наприклад, Postgres, MongoDB або CouchDB) і JSON для уніфікованого стека розробки JavaScript. З адаптацією того, що було, по суті, шаблонами розробки на стороні сервера, такими як MVC, MVP, MVVM і т.д., Node.js дозволяє повторно використовувати ту ж саму модель і інтерфейс сервісу між клієнтською і серверною сторонами.

3.3.5 Цикл подій

Средеовище Node.js реєструється в операційній системі, тому ОС повідомляє про з'єднання і видає зворотний виклик. У середовищі виконання Node.js кожне підключення є невеликим розподілом купи. Традиційно процеси або потоки відносно важкої ваги обробляються кожним з'єднанням. Node.js використовує цикл подій для масштабованості, а не процесів або потоків. На відміну від інших

серверів, керованих подіями, цикл подій Node.js не потрібно називати явно. Замість цього визначаються зворотні виклики, і сервер автоматично вводить цикл подій в кінці визначення зворотного виклику. Node.js виходить з циклу подій, коли відсутні додаткові зворотні виклики.

3.4 Технології для розробки інтерфейсу

Для створення клієнтської частини порталу були використані такі технології як HTML5, CSS3, JavaScript, а також додаткові бібліотека React, яка автоматизують створення інтерфейсу та спрощують доступ до різних частин порталу.

Сама структура інтерфейсу була реалізована за допомогою мови розмітки HTML, зокрема HTML5.

HTML впроваджує засоби для :

- створення структурованого документу шляхом позначення структурного складу тексту: заголовки, абзаци, списки, таблиці, цитати та інше;
- отримання інформації із Всесвітньої мережі через гіперпосилання;
- створення інтерактивних форм;
- включення зображень, звуку, відео, та інших об’єктів до тексту.

Щодо використання самої останньої версії цієї мови, то вона має ряд суттєвих переваг, так як при прийнятті рішення про введення нових тегів було розглянуто більшість популярних сайтів і виділено основні елементи, які були спільними для всіх веб-сторінок.

Розмічаючи області на сторінці за допомогою певних елементів, ця технологія може допомогти полегшити користувачеві навігацію. Наприклад, він може легко пропустити розділ навігації або швидко переходити від однієї статті до іншої без необхідності для авторів робити відповідні посилання. Автори також отримують вигоду в результаті заміни великої кількості div-ів одним з декількох відповідних елементів, що також приводить до чистого і легкого для автора початкового коду.

Елементами header є заголовки розділів. Вони можуть складатися з декількох частин — наприклад, було б виправдано розділяти блок заголовка на підзаголовки,

історію версій або вказання авторства. Елемент footer визначає нижню частину розділу, до якого він відноситься. Зазвичай він містить інформацію про розділ — наприклад, ім’я автора, посилання на схожі документи, копірайт і тому подібне. Блок nav містить список посилань для навігації. Підходить, наприклад, для навігації по сайту, або для змісту. Елемент aside підходить для розміщення вмісту яким-небудь чином спорідненого основному контенту. У звичайному випадку буде корисний для розмітки бічної колонки. Тег section представляє загальний розділ документа або додатку, наприклад, такий як розділ. Тег article відзначає незалежний розділ документа, сторінки або сайту. Застосовується для такого вмісту як новини, запису блога, повідомлення у форумі або коментарі користувачів.

Підтримкою для HTML є CSS .

Каскадні таблиці стилів (англ. Cascading Style Sheets або скорочено CSS) — спеціальна мова, що використовується для опису сторінок, написаних мовами розмітки даних.

Стилі CSS використовується авторами та відвідувачами веб-сторінок, щоб визначити кольори, шрифти, верстку та інші аспекти вигляду сторінки. Одна з головних переваг — можливість розділити зміст сторінки (або контент, наповнення, зазвичай HTML, XML або подібна мова розмітки) від вигляду документу (що описується в CSS).

Таке розділення може покращити сприйняття та доступність контенту, забезпечити більшу гнучкість та контроль за відображенням контенту в різних умовах, зробити контент більш структурованим та простим, прибрати повтори тощо. CSS також дозволяє адаптувати контент до різних умов відображення (на екрані монітора, мобільного пристрою (КПК), у роздрукованому вигляді, на екрані телевізора, пристроях з підтримкою шрифту Брайля або голосових браузерах та ін.) .

Переваги CSS:

- інформація про стиль для усього сайту або його частин може міститися в одному .css-файлі, що дозволяє швидко робити зміни в дизайні та презентації сторінок;

- різна інформація про стилі для різних типів користувачів: наприклад великий розмір шрифту для користувачів з послабленим зором, стилі для виводу сторінки на принтер, стиль для мобільних пристроїв;
- сторінки зменшуються в об’ємі та стають більш структурованими, оскільки інформація про стилі відділена від тексту та має певні правила застосування і сторінка побудована з урахуванням їх;
- прискорення завантаження сторінок і зменшення обсягів інформації, що передається, навантаження на сервер та канал передачі. Досягається за рахунок того, що сучасні браузери здатні кешувати (запам’ятовувати) інформацію про стилі і використовувати для всіх сторінок, а не завантажувати для кожної.

Взаємодія користувача з інтерфейсом відбувається за допомогою мови програмування JavaScript.

Мова JavaScript (JS) — динамічна, об’єктно-орієнтована мова програмування. Реалізація стандарту ECMAScript. Найчастіше використовується як частина браузера, що надає можливість коду на стороні клієнта (такому, що виконується на пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-сторінки.

Мова JavaScript класифікують як прототипну (підмножина об’єктно-орієнтованої), скриптову мову програмування з динамічною типізацією. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування (імперативну та частково функціональну) і деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація, автоматичне керування пам’яттю, прототипне наслідування, функції як об’єкти першого класу.

3.5.Технології для взаємодії з сервером

Окрім вказаних вище технологій, під час розробки було використано

додаткові бібліотеки та технології, створені на їх основі.

Основним фреймворком для взаємодії з сервером був React.js.

Фремворк React.js — це JavaScript-фреймворк з відкритим програмним кодом, який розробляє Google. Призначений для розробки односторінкових застосунків, що складаються з одної HTML сторінки з CSS і JavaScript. Його мета — розширення браузерних застосунків на основі шаблону “Модель-вид-контролер” (MVC), а також спрощення їх тестування та розробки.

Фреймворк працює зі сторінкою HTML, що містить додаткові атрибути і пов’язує області вводу або виводу сторінки з моделлю, яка являє собою звичайні змінні JavaScript. Значення цих змінних задаються вручну або отримуються зі статичних або динамічних JSON-даних.

Фремворк React.js спроектований з переконанням, що декларативне програмування найкраще пасує для побудови інтерфейсів користувача та опису програмних компонентів, в той час як імперативне програмування пасує для опису бізнес-логіки. Фреймворк адаптує та розширює традиційний HTML, щоб забезпечити двосторонню прив’язку даних для динамічного контенту, що дозволяє автоматично синхронізувати модель та вид. У результаті React.js зменшує роль DOM-маніпуляцій з метою підвищення продуктивності і спрощення тестування.

4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Система управління дипломними проектами повинна мати функціонал авторизації користувача, функцію нагадування паролю та реєстрації нових користувачів. Система містить 3 ролі студент, викладач та додатковий персонал.

Моя частина проекту - це додатковий персонал.

Для back-end частини системи я використав сервер на основі Node.js REST API та базу даних MySQL.

Для front-end я використав бібліотеку React.js.

4.1 Бібліотека React.js

Біліотека React (також відомий як React.js або ReactJS) - це бібліотека JavaScript для створення інтерфейсів користувача. Він підтримується Facebook і спільнотою окремих розробників і компаній.

Біліотека React може бути використаний як основа в розробці односторінкових або мобільних додатків, оскільки він оптимальний лише для того, щоб його використання було найшвидшим способом отримання швидко змінюваних даних, які необхідно записати. Однак отримання даних є лише початком того, що відбувається на веб-сторінці, тому складні програми React зазвичай вимагають використання додаткових бібліотек для управління станом, маршрутизації та взаємодії з API.

Тому щоб розділити бізнес логіку та безпосередньо код для відображення сторінки я використав розширення Redux.

В React – кожен елемент та клас є компонентом. В яким можна передавати його стан (state) у вигляді властивостей (props).

Ось приклад простого класу-компонента React

```
class ParentComponent extends React.Component {
  state = { color: 'green' };
  render() {
    return (
      <ChildComponent color={this.state.color} />
    );
  }
}
```

Щоб створити React за стосунок я використовував редактор Jetbrains WebShtorm та npm редактор пакетів і використав create-react-app.

create-react-app - це зручне середовище для навчання React, і це найкращий спосіб почати створення нової односторінкової програми в React.

Це допомагає налаштувати середовище розробки таким чином, що ви можете використовувати останні функції JavaScript, надає приємний досвід розробника та оптимізує роботу вашого додатка. На вашому комп'ютері потрібно мати Node> = 6 і npm> = 5.2. Щоб створити проект, я виконав:

npx create-react-app my-app

cd my-app

npm start

5. РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

Для використання реалізованого веб-застосунку необхідно мати встановленним будь-який Інтернет браузер. Потрібно переконатися, що ваша система відповідає данним вимогам. Хоча це, як правило, не є проблемою, коли мова йде про вимоги до апаратних засобів, ви можете помітити, що це, наприклад, зовсім інша історія, коли мова йде про підтримувані операційні системи. Користувачі Firefox у Windows 2000, наприклад, помітять, що вони не зможуть оновитись з Firefox 12 на 13 у найближчому майбутньому, оскільки Mozilla знизила підтримку цієї операційної системи, починаючи з цієї версії браузера.

5.1 Опис випробування розроблених програмних засобів

Щоб реалізувати доступ до систми лише авторизованим корисувачам, було розробленно авторизацію (Рис.3).

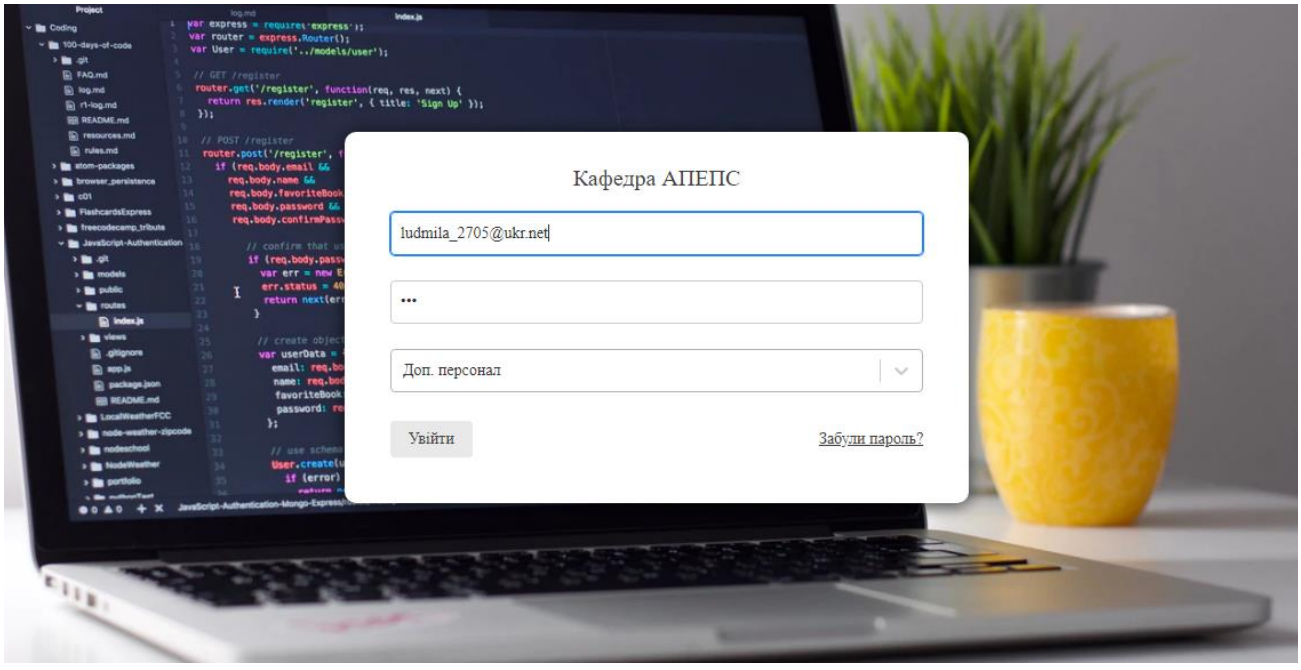


Рисунок 5.1 Форма авторизації користувача в систему

Після успішної авторизації менеджер має змогу шукати, переглядати дипломні роботи, студентів, викладачів ,групи, та редагувати їх.

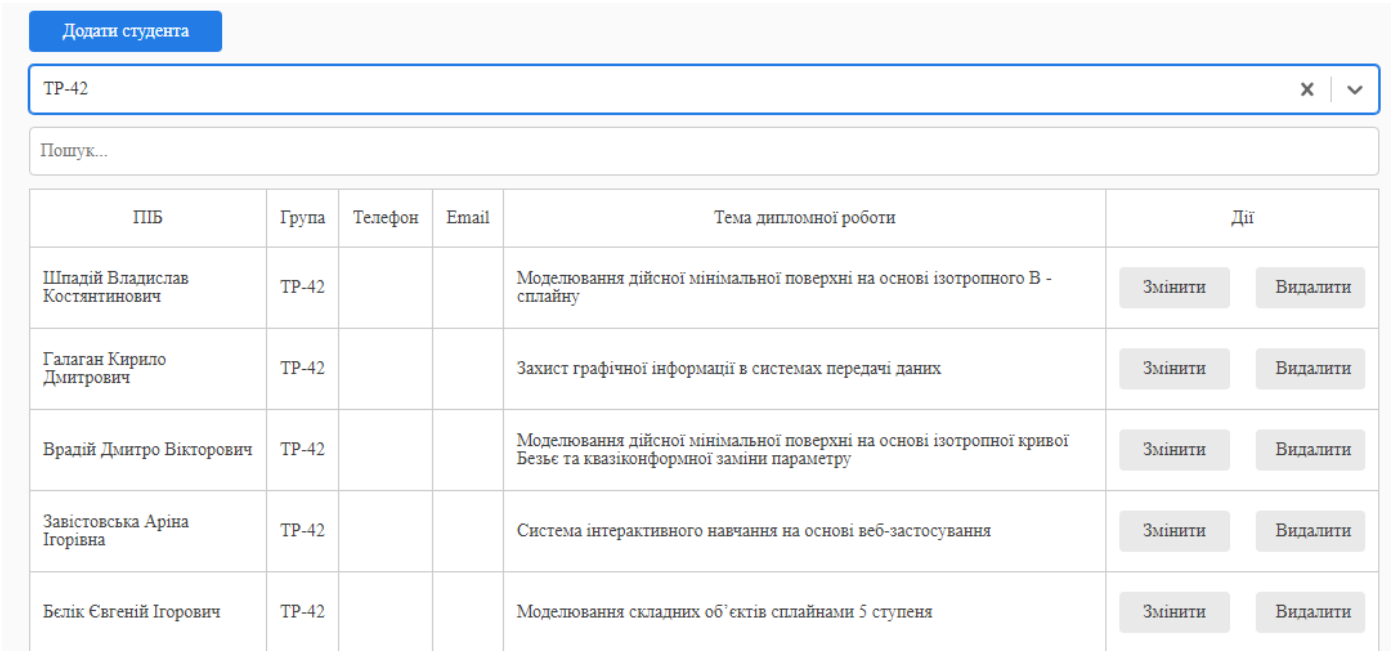


Рис. 5.2. Зразок інтерфейсу для пошуку студентів

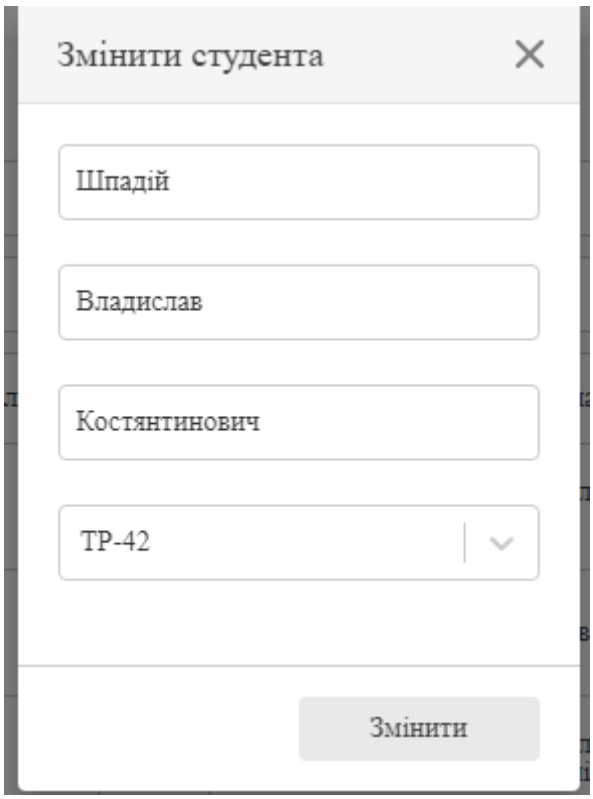
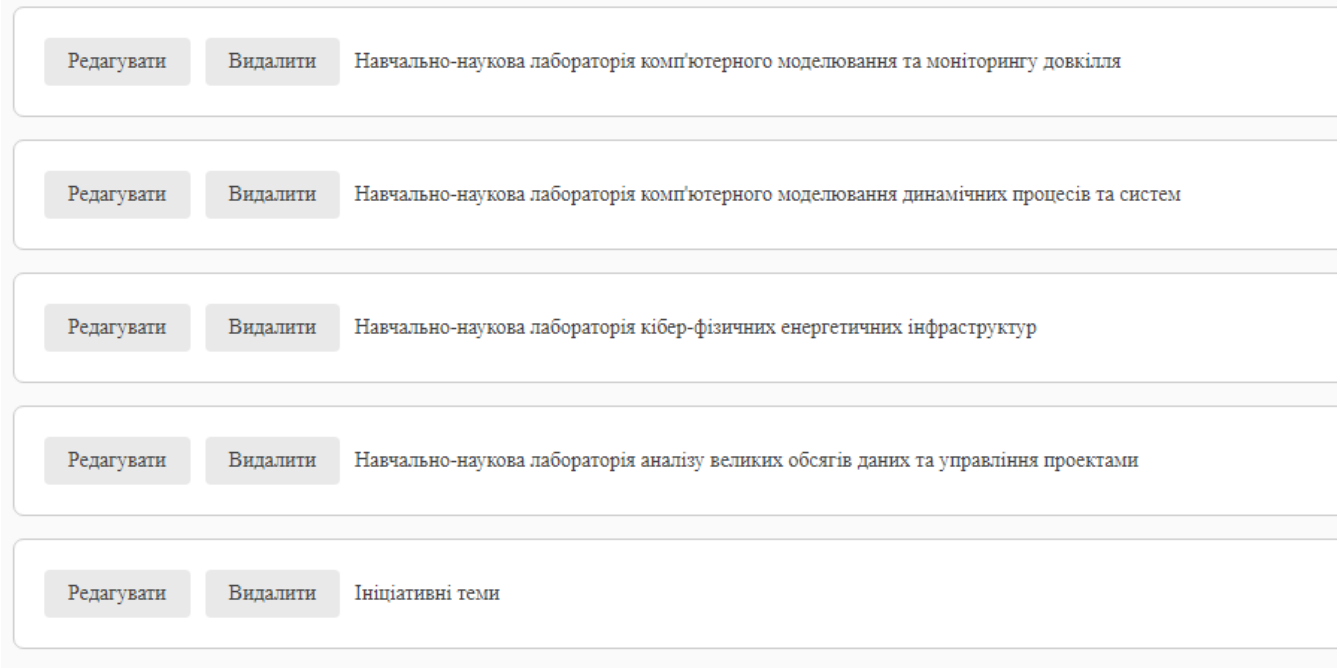


Рис. 5.3. Вікно редагування тсудента

При натисканні на кнопку додати студента/редагувати користувач бачить додаткове спливаюче вікно, яке містить форму для додання нового студента, або редагування існуючої інформації (рисунок 5.3)

Аналогічно зі сторінкою «студенти» сторінка «викладачі та додатковий персонал має ті ж самі функції»

На сторінці лабораторії додатковий персонал може редагувати список лабораторій (рисунок 5.4)



Також може редагувати певну лабораторію , додавати/видаляти/редагувати до неї напрямки лабораторії та викладачів.(рисунок 5.5)

На сторінці «Графік виконання» ПДП переглядає графік (рисунок 5.6) та редагує його для різних навчальних ступенів (кнопки переключення знаходять зверху сторінки). Є можливість редагувати пункти графіку, задавати початкову та кінцеві дати, додавати опис роботи, після натискання кнопки редагувати. (Рисунок 5.7)

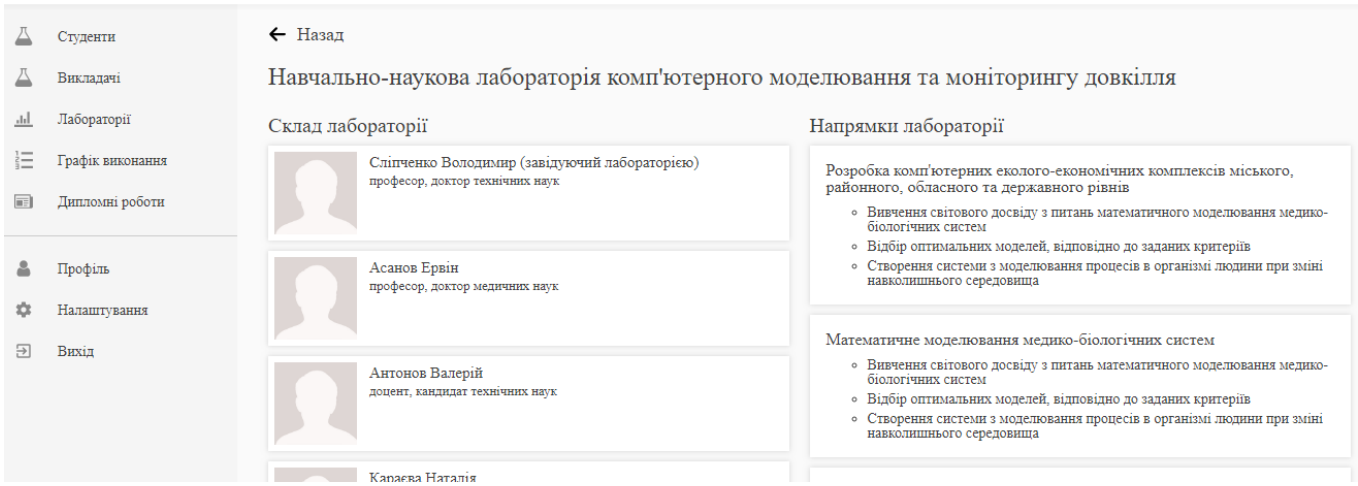
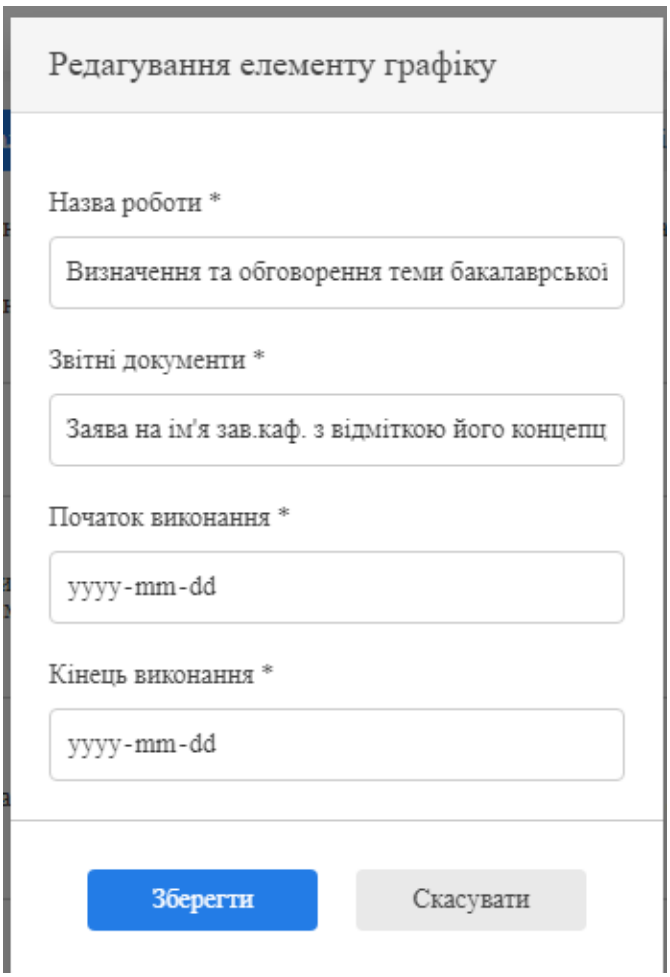


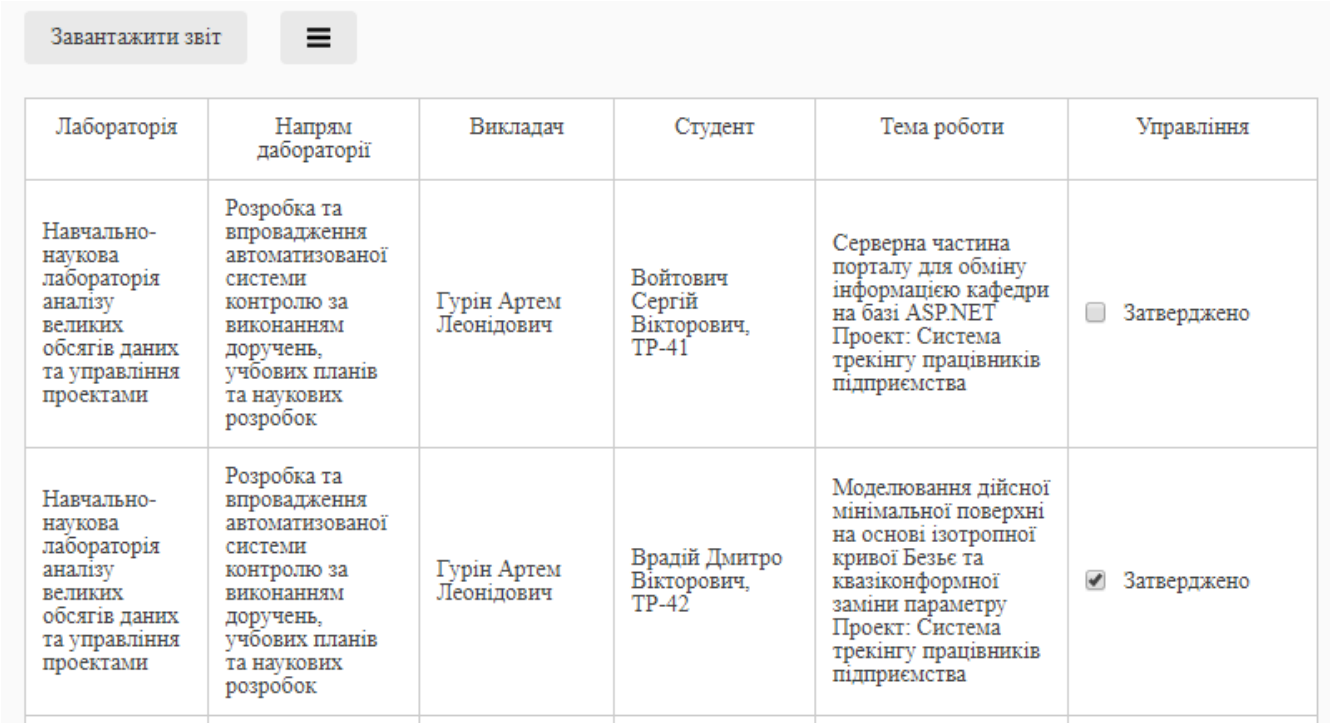
Рисунок 5.5 — Сторінка лабораторії

Рисунок 5.6 — Сторінка «Графік виконання»

На сторінці дипломні роботи ПДП переглядає таблицю дипломних робіт (студент, викладач, тема роботи, напрям лабораторії, лабораторія, управління) – (Рисунок 5.8), і може завантажити звіт у форматі .docx.(табл. 5.1)



(Рисунок 5.7) – Редагування пункту графіка виконання



Лабораторія	Напрям дабораторії	Викладач	Студент	Тема роботи	Управління
Навчально-наукова лабораторія аналізу великих обсягів даних та управління проектами	Розробка та впровадження автоматизованої системи контролю за виконанням доручень, учбових планів та наукових розробок	Гурін Артем Леонідович	Войтович Сергій Вікторович, ТР-41	Серверна частина порталу для обміну інформацією кафедри на базі ASP.NET Проект: Система трекінгу працівників підприємства	☐ Затверджено
Навчально-наукова лабораторія аналізу великих обсягів даних та управління проектами	Розробка та впровадження автоматизованої системи контролю за виконанням доручень, учбових планів та наукових розробок	Гурін Артем Леонідович	Врадій Дмитро Вікторович, ТР-42	Моделювання дійсної мінімальної поверхні на основі ізотропної кривої Безьє та квазіконформної заміни параметру Проект: Система трекінгу працівників підприємства	☑ Затверджено

(Рисунок 5.8) – Сторінка дипломні роботи на якій таблиця дипломних робіт

Група ТВ-41

№ п/п	Прізвище, ім’я та по-батькові студента	Дипломна робота бакалавра	Вчене звання, ПІБ керівника
1	**2**	**3**	**4**
1.	Мельниченко Артем Васильович	Засіб автоматизації звітності про дефекти програмного забезпечення	професор Крамар Ю. М.

(Таблиця 5.1) – Приклад таблиці із файлу звіту по дипломним роботам студентів

5.2. Технічні вимоги до середовищ використання

Мінімальними вимогами для процессорів, які запускаються в браузер наопераційній системі Windows є Internet Explorer 8. Він потребує 233 Mhz процесор;мінімум 64 MB RAM (для Windows XP), рекомендовано 512 MB RAM; мінімум 150MB (Windows XP), 70 MB (Windows Vista) вільного місця на диску.

В свою чергу браузер Google Chrome потребує процесор Pentium 4 дляопераційної системи Windows, та прецессор Intel для операційної системи Mac;мінімум 128 MB RAM; мінімум 100 MB вільного місця на диску. Підтримка браузераопераційними системами починається з Windows XP SP2, OS X 10.5.6, Ubuntu 10.04,Debian 6, OpenSuse 11.3 та Fedora Linux 14.

ВИСНОВКИ

Під час проходження переддипломної практики було теоретично обґрунтовано доцільність використання технологій для розробки програмного продукту, що призначений для вибору тем дипломних робіт у ВНЗ та збільшення ефективності комунікацій між студентами та науковими керівниками. Було покращено навички розробки клієнтських web-застосунків на всіх стадіях розробки – починаючи від проектування шаблону у графічному редакторі і закінчуючи відображенням результатів запитів користувачів на web-сторінці.

У результаті проходження переддипломної практики отримано практичні результати. А саме, було виконано аналіз основних аналогів, виявлено їхні переваги та недоліки, на основі отриманих результатів було прийнятно рішення про створення власного додатку, який би вирішував проблеми, що присутні в аналогах, а також був доповнений новим функціоналом. В результаті проведення варіантного аналізу, для розробки було обрано мову програмування JavaScript, засоби веб-розробки HTML5, CSS3 та фреймворк React, який має ряд переваг. Зокрема, React дозволяє з легкістю змінювати наявні компоненти і перевикористовувати вже написаний код, що в свою чергу підвищує швидкість розробки, спрощує процес тестування, і, як результат, знижує витрати.

Були вирішенні наступні задачі:

- розроблено загальну структуру програмного продукту;
- розроблено модуль для реєстрації та авторизації користувача;
- розроблено модуль для роботи з сервером;
- розроблено інтерфейс програмного продукту;

Отже, у результаті переддипломної практики було розроблено основні модулі web-додатку, що призначений для вибору тем дипломних робіт у ВНЗ та розроблено кінцевий інтерфейс користувача.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Guidelines for Universal Windows Platform (UWP) apps [Electronic resource]. — Access mode: https://msdn.microsoft.com/en-us/library/windows/apps/hh465424.aspx .
2. Флэнаган Д. JavaScript. Подробное руководство [Электронный ресурс] / Д. Флэнаган. — 2012. — С. 524–550. — Режим доступа: http://www.books.ru/books/javascript-podrobnoe-rukovodstvo-6-e-izdanie-1814274/ .
3. Cape Town Open Education Declaration: Unlocking the promise of open educational resources [Electronic resource]. — Access mode: http://www.capetowndeclaration.org/read-the-declaration .
4. UNESCO IITE [Electronic resource]. — Access mode: http://iite.unesco.org/pics/publications/ru/files/3214680.pd .
5. JetBrains: WebShtorm FEATURES [Electronic resource]. — Access mode: https://www.jetbrains.com/webstorm/features/ .
6. W3C Standerds HTML & CSS [Electronic resource]. — Access mode: https://www.w3.org/standards/webdesign/htmlcss .
7. Флэнаган Д. JavaScript. Подробное руководство [Электронный ресурс] / Д. Флэнаган. — 2012. — С. 468–470. — Режим доступа: http://www.books.ru/books/javascript-podrobnoe-rukovodstvo-6-e-izdanie-1814274/ .
8. phpMyAdmin: — Bringing MySQL to the web [Electronic resource]. — Access mode: https://www.phpmyadmin.net/ .
9. React.js: Tutorial 7: — Routing & Multiple Views [Electronic resource]. — Access mode: https://docs.React.js.org/tutorial/step_07 .
10. Express 4.x - API Reference: — 4.x API[Electronic resource]. — Access mode: https://expressjs.com/en/4x/api.html .

11.Гарнаев, Андрей WEB-программирование на Java и JavaScript / Андрей Гарнаев , Сергей Гарнаев. - М.: БХВ-Петербург, 2012. - 179 с.

12. Ньюмен Сэм. Создание микросервисов. СПб.: Питер, 2016. 304 с.: ил. (Серия «Бестселлеры O’Reilly»).

13. Таненбаум Э., Стеен М. ван. Распределенные системы. Принципы и парадигмы. СПб.: Питер, 2003. 877 с.: ил. (Серия «Классика computer science»).

14. Martinfowler. [Электронный ресурс]: Microservices. a definition of this new architectural term. Режим доступа: https://martinfowler.com/articles/microservices.html/ (дата обращения: 22.05.2019).

15. Хабрахабр. [Электронный ресурс]: Microsoft Azure - Azure Service Fabric и архитектура микросервисов. Режим доступа: https://habrahabr.ru/company/mailru/blog/320962/ (дата обращения: 06.05.2016).

16. MSDN Magazine Blog. [Электронный ресурс]: Архитектура микросервисов. Режим доступа: https://msdn.microsoft.com/ru-ru/magazine/mt595752.aspx/ (дата обращения: 22.05.2019).

ДОДАТОК А

Модуль “Додатковий персонал” системи управління дипломними проектами

Специфікація

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТВ51167_18Б

Аркушів 1

Київ 2019

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТВ51167_18Б	Записка.docx	Пояснювальна записка
Компоненти		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТВ51167_18Б 12-1	addLecturer.js Lecturers.Content.js Laboratory.js	Основні компоненти
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТВ51167_18Б 13-1	Додаток В.doc	Опис програмного коду

ДОДАТОК Б

Модуль “Додатковий персонал” системи управління дипломними проектами

Текст програми

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТВ51167_18Б 12-1

Аркушів 12

Київ 2019

AddLecturer.js

```
const db = require('../../../db/mysql');
const cyrillicToTranslit=require('cyrillic-to-translit-js/CyrillicToTranslit');
const CustomError =require('../../../services/exceptionServices');

module.exports = async function(req, res, next) {

    try {
        const { token } = req.headers;
        let { firstName, middleName, lastName, degreeId, gradeId, officeId , phone,homeAdress, laboratoryId} = req.body;

        const tokenRecord = await db.query(
            `SELECT helper_personnels.helperId
                FROM tokens
                INNER JOIN helper_personnels ON helper_personnels.helperId = tokens.userID
                WHERE token='${token}'`,
        );
        if (tokenRecord.length === 0) {
            throw new CustomError(401,'User is not authorized to access this resource');
        }

        let username=cyrillicToTranslit().transform(lastName)
            +'_'
            +Math.random()
                .toString(36)
                .substring(8, 10);
```

```
username=toSqlValue(username);
firstName=toSqlValue(firstName);
middleName=toSqlValue(middleName);
lastName=toSqlValue(lastName);
degreeId=toSqlValue(degreeId);
gradeId=toSqlValue(gradeId);
officeId=toSqlValue(officeId);
phone=toSqlValue(phone);
homeAdress=toSqlValue(homeAdress);

const result =await db.query(
   `INSERT INTO lecturers
   (lecturerId, lastName, firstName, middleName, username, password, officeId, degreeId, gradeId, phone, homeAdress)
   VALUES
   (NULL, ${lastName}, ${firstName}, ${middleName}, ${username}, ${username}, ${officeId}, ${degreeId}, ${gradeId}, ${phone}, ${homeAdress});

   SET @lecturerId =(SELECT LAST_INSERT_ID());
   SELECT @lecturerId AS 'lecturerId';
   `
);
const {lecturerId}=result[2][0];
if(laboratoryId){
   let q=
      `INSERT INTO laboratoryMembers
   (lecturerId, laboratoryId, isHead)
   VALUES
   (${lecturerId}, ${laboratoryId}, '0')
   `;
```

```
            await db.query(q);
        }

        res.status(201).send({
            status: 'ok',
            lecturerId
        });

    }catch (e) {
        if (e instanceof CustomError) {
            res.status(e.code);
            res.send({
                error: {
                    code: e.code,
                    message: e.message
                },
            });
        } else {
            res.status(500);
            console.log(e);
            res.send({
                error: {
                    code: 500,
                    message: 'Server error: ' + e.message
                },
            });
        }
    }
};
```

```
    const toSqlValue=(value)=>{
        if(!value) {
            value=`NULL`;
        } else {
            value=`'${value.toString().replace("'", "\\'")}'`
        }

        return value;
    };
```

Lecturers.Content.js

```
import React from 'react'

import { Table, Td, Th, Tr } from '../../../UI/Table'

import { Button } from '../../../UI'

import { Flex } from '../../../../styles/common'


const Content = ({ lecturers, openEditModal, openDeleteModal }) => (
  <Table paddingChild='1.5rem 1rem'>
    <thead>
      <Tr>
        <Th>ПІБ</Th>
        <Th>Ступінь</Th>
        <Th>Посада</Th>
        {/*<Th>Навантаження БД БЗ МД МЗ</Th>*/}
```

```
    <Th>Лабораторія</Th>
    <Th>Телефон</Th>
    <Th>Email</Th>
    <Th>Дії</Th>
  </Tr>
</thead>
<tbody>
  {lecturers && lecturers.map(item => (
    <Tr key={item.lecturerId} >
      <Td>{item.lastName + ' ' + item.firstName + ' ' + item.middleName}</Td>
      <Td>{item.degreeTitle}</Td>
      <Td>{item.officeTitle}</Td>
      {/*<Td>{item.studyload && item.studyload.map(el => el.cnt).join(' / ')}</Td>*/}
      <Td>{item.laboratoryName}</Td>
      <Td>{item.phone}</Td>
      <Td>{item.email}</Td>
      <Td>
        <Flex>
          <Button
            onClick={() => openEditModal(item)}
            width='100%'
```

```
            >
              Змінити
            </Button>
            <Button
              onClick={() => openDeleteModal(item.lecturerId)}
              width='100%'
            >
              Видалити
            </Button>
          </Flex>
        </Td>
      </Tr>
    ))}

  </tbody>
</Table>
)export default Content
```

Laboratory.js

```
/* eslint-disable no-undef */
import React, { Component } from 'react'
import { connect } from 'react-redux'
```

```
import history from '/src/utils/history'
import { setFiltersValueAction, resetFiltersAction } from '/src/store/actions/diplomasFiltersActions'
import { changeTabAction } from '/src/store/actions/studentDiplomasActions'
import { toggleFiltersAction } from '/src/store/actions/settingsActions'
import { helpers } from '/src/utils'
import GoBack from '/src/components/GoBack'
import { laboratoriesAction } from '../../../store/actions/laboratoriesActions'
import { Button, Loader, Text } from '../../UI'
import { Flex } from '../../../styles/common'
import { LaboratoryField, LaboratorySection, MemberAvatar } from '../../Laboratory/style'
import { Title } from '../../UI/Title'
import { LaboratoryMember } from './style'
import LaboratoryFieldModal from './Modals/LaboratoryFieldModal'
import {
  openLaboratoryFieldModalAction,
  openLaboratoryMemberModalAction,
  removeLaboratoryFieldAction,
  removeLaboratoryMemberAction,
  selectLaboratoryAction
} from '../../../store/actions/helperActions/laboratoryActions'
import { NotificationContext } from '../../HOC/NotificationProvider'
import { lecturersAction } from '../../../store/actions/referenceActions'
import LaboratoryMemberModal from './Modals/LaboratoryMemberModal'

@connect(
  ({laboratories, rootHelperReducer, auth: {user}}, {id}) => ({
    laboratory: laboratories.laboratories[id],
```

```
    selectedLaboratory: rootHelperReducer.laboratories.selectedLaboratory,
    user
  }),
  {
    laboratoriesAction,
    setFiltersValueAction,
    resetFiltersAction,
    toggleFiltersAction,
    changeTabAction,
    selectLaboratoryAction,
    removeLaboratoryFieldAction,
    openLaboratoryFieldModalAction,
    removeLaboratoryMemberAction,
    lecturersAction,
    openLaboratoryMemberModalAction
}
)
class Laboratory extends Component {
  componentDidMount () {
    this.props.selectLaboratoryAction(this.props.id)

    this.props.laboratoriesAction(this.props.id)
    this.props.lecturersAction()
  }
  static contextType = NotificationContext

  goToLaboratories = () => {
    history.push('/laboratories')
  }
```

```
  removeField = (laboratoryFieldId) => {
    this.props.removeLaboratoryFieldAction({ laboratoryFieldId }, () => {
      this.context.addNotification({
        title: 'Success',
        message: `Напрям лабораторії було успішно видалено`,
        level: 'success'
      })

      this.props.laboratoriesAction(this.props.id)
    })
  }

  removeMember = (lecturerId) => {
    this.props.removeLaboratoryMemberAction( { laboratoryId: this.props.selectedLaboratory, lecturerId }, () => {
      this.context.addNotification({
        title: 'Success',
        message: `Склад лабораторії успішно оновлений`,
        level: 'success'
      })

      this.props.laboratoriesAction(this.props.id)
    })
  }

  render () {
    const {laboratory} = this.props

    if (!laboratory) return <Loader>Завантаження...</Loader>
```

```
      return (
        <div>
          <GoBack clickHandler={this.goToLaboratories}/>
          <Title
            title={laboratory.laboratoryName}
            size='2.4rem'
            margin='1.5rem 0'
          />
          <Flex wrap='wrap'>
            <LaboratorySection>
              <Title title='Склад лабораторії' size='2rem' margin='1rem 0'/>
              <div>
                {laboratory.laboratoryMembers.map(item => (
                  <LaboratoryMember key={item.lecturerId}>
                    <MemberAvatar src={helpers.userAvatar(item.loginURL)}/>
                    <div>
                      <Text padding='0.3rem 0'>
                        {item.lastName}  {item.firstName}  {item.moddleName}
{item.isHead ? '(завідуючий лабораторією)' : ''}
                      </Text>
                      <Text size='1.3rem'>
                        {item.officeTitle}, {item.degreeTitle}
                      </Text>
                      <Button style={{ marginTop: '.7rem' }} onClick={() =>
this.removeMember(item.lecturerId)}>Видалити з лабораторії</Button>
                    </div>
                  </LaboratoryMember>
                ))}
                <LaboratoryMember>
```

```
            <Text size='1.5rem' style={{margin: '1.5rem', textTransform:
'uppercase', cursor: 'pointer'}} onClick={() => {
              this.props.openLaboratoryMemberModalAction()
            }}>
              + Додати викладача
            </Text>
          </LaboratoryMember>
        </div>
      </LaboratorySection>
      <LaboratorySection>
        <Title title='Напрямки лабораторії' size='2rem' margin='1rem 0'/>
        <div>
          {laboratory.laboratoryFields.map(item => (
            <LaboratoryField key={item.laboratoryFieldId}>
              <Title title={item.laboratoryFieldTitle} size='1.6rem' margin='0 0
1rem 0'/>
              <div                              dangerouslySetInnerHTML={{__html:
item.laboratoryFieldDescription}}/>
              <Button style={{ marginTop: '1rem' }} onClick={() => {
                this.props.openLaboratoryFieldModalAction(item.laboratoryFieldId)
              }}>Редагувати</Button>
              <Button style={{ marginTop: '1rem' }} onClick={() =>
this.removeField(item.laboratoryFieldId)}>Видалити</Button>
            </LaboratoryField>
          ))}
          <LaboratoryMember>
            <Text size='1.5rem' style={{margin: '1.5rem', textTransform:
'uppercase', cursor: 'pointer'}} onClick={() => {
              this.props.openLaboratoryFieldModalAction()
            }}>
```

```
                + Додати напрям
              </Text>
            </LaboratoryMember>
          </div>
        </LaboratorySection>
      </Flex>
      <LaboratoryFieldModal/>
      <LaboratoryMemberModal/>
    </div>
  )
 }
}

export default Laboratory
```

ДОДАТОК В

Модуль “Додатковий персонал” системи управління дипломними проектами

Опис програми

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТВ51167_18Б 13-1

Аркушів 1

Київ 2019

Опис програмного коду

AddLecturer.js – це файл в якому описаний endpoint з back-end серверу, що написаний за допомогою Express.js. Цей файл відповідає за додавання викладача в БД У цьому файлі є основна функція яка приймає параметри req і res. Параметр req це зміна де зберігаються всі данні про запит що надіслав користувач. З цієї змінної ми беремо токен авторизації та данні викладача для додавання. Далі функція оброблює данні та додає викладача в БД. І передає в параметр res (response-відповідь користувачу) ідентифікатор викладача що був доданий.

Lecturers.Contennt.js-– це файл в якому описано вигляд таблиці викладачів для додаткового персоналу з front-end серверу, що написаний за допомогою React.js. В цьому файлі є константа яка приймає список (масив) викладачів та в циклі створює нові рядки таблиці для відображення викладачів.

Laboratory.js-– це файл в якому описано вигляд списку лабораторій для додаткового персоналу з front-end серверу, що написаний за допомогою React.js. В цьому файлі є константа яка приймає список (масив) лабораторій та в циклі створює нові пункти відображення лабораторій.