

Міністерство освіти і науки України  
Одеський національний політехнічний університет  
Інститут комп'ютерних систем  
Кафедра інформаційних систем

## **КУРСОВА РОБОТА**

з дисципліни «Технології створення програмних продуктів»

за темою

«TrackTeacher»

Виконав(ла):  
студент 3-го курсу  
групи АІ-183  
Присяжний Ю. О.  
Перевірив:  
Блажко О. А.

Одеса-2020

## Анотація

В курсовій роботі розглядається процес створення програмного продукту «TrackTeacher». В пояснювальній записці у розділах «Проектування» та «Конструювання» детально описано особливості конструювання:

- структур даних в системі керування базами даних PostgreSQL;
- програмних модулів в інструментальному середовищі PHPStorm з використанням фреймворку Laravel та мови програмування PHP.

Результати роботи розміщено на *github*-репозиторіях за адресами:

- [Документація](#)
- [Фронтенд](#)
- [Сервер](#)
- [Інструкція з розгортання](#)

## **Перелік скорочень**

ОС – операційна система

ІС – інформаційна система

БД – база даних

СКБД – система керування базами даних

ПЗ – програмне забезпечення

ПП– програмний продукт

UML – уніфікована мова моделювання

SPA – Single page application

## Зміст

	стор.
1 Вимоги до програмного продукту	7
1.1 Визначення потреб споживача	7
1.1.1 Ієрархія потреб споживача	7
1.1.2 Деталізація матеріальної потреби	7
1.2 Бізнес-вимоги до програмного продукту	8
1.2.1 Опис проблеми споживача	8
1.2.1.1 Концептуальний опис проблеми споживача	8
1.2.1.2 Метричний опис проблеми споживача	8
1.2.2 Мета створення програмного продукту	9
1.2.2.1 Проблемний аналіз існуючих програмних продуктів	9
1.2.2.2 Мета створення програмного продукту	9
1.2.3 Назва програмного продукту	9
1.2.3.1 Гасло програмного продукту	9
1.2.3.2 Логотип програмного продукту	10
1.3 Вимоги користувача до програмного продукту	10
1.3.1 Історія користувача програмного продукту	10
1.3.2 Діаграма прецедентів програмного продукту	11
1.3.3 Сценаріїв використання прецедентів програмного продукту	11
1.4 Функціональні вимоги до програмного продукту	14
1.4.1. Багаторівнева класифікація функціональних вимог	14
1.4.2 Функціональний аналіз існуючих програмних продуктів	16
1.5 Нефункціональні вимоги до програмного продукту	16
1.5.1 Опис зовнішніх інтерфейсів	16
1.5.1.1 Опис інтерфейса користувача	16
1.5.1.1.1 Опис INPUT-інтерфейса користувача	16
1.5.1.1.2 Опис OUTPUT-інтерфейса користувача	17
1.5.1.2 Опис інтерфейсу із зовнішніми пристроями	18

1.5.1.3	Опис програмних інтерфейсів	18
1.5.1.4	Опис інтерфейсів передачі інформації	19
1.5.1.5	Опис атрибутів продуктивності	19
2	Планування процесу розробки програмного продукту	21
2.1	Планування ітерацій розробки програмного продукту	21
2.2	Концептуальний опис архітектури програмного продукту	21
2.3	План розробки програмного продукту	22
2.3.1	Оцінка трудомісткості розробки програмного продукту	22
2.3.2	Визначення дерева робіт з розробки програмного продукту	24
2.3.3	Графік робіт з розробки програмного продукту	25
2.3.3.1	Таблиця з графіком робіт	25
2.3.3.2	Діаграма Ганта	25
3	Проектування програмного продукту	26
3.1	Концептуальне та логічне проектування структур даних програмного продукту	26
3.1.1	Концептуальне проектування на основі UML-діаграми концептуальних класів	26
3.1.2	Логічне проектування структур даних	26
3.2	Проектування програмних класів	27
3.3	Проектування алгоритмів роботи методів програмних класів	27
3.4	Проектування тестових наборів методів програмних класів	31
4	Конструювання програмного продукту	33
4.1	Особливості конструювання структур даних	33
4.1.1	Особливості інсталяції та роботи з СУБД	33
4.1.2	Особливості створення структур даних	33
4.2	Особливості конструювання програмних модулів	35
4.2.1	Особливості роботи з інтегрованим середовищем розробки	35
4.2.2	Особливості створення програмної структури з урахуванням спеціалізованого Фреймворку	35

4.2.3 Особливості створення програмних класів	36
4.2.4 Особливості розробки алгоритмів методів програмних класів або процедур/функцій	44
4.3 Модульне тестування програмних класів	51
5 Розгортання та валідація програмного продукту	62
5.1 Інструкція з встановлення програмного продукту	62
5.2 Інструкція з використання програмного продукту	62
5.3 Результати валідації програмного продукту	66
Висновки до курсової роботи	67

## 1 Вимоги до програмного продукту

### 1.1 Визначення потреб споживача

#### 1.1.1 Ієрархія потреб споживача

Відомо, що в теорії маркетингу потреби людини можуть бути представлені у вигляді ієрархії потреб ідей американського психолога Абрахама Маслоу включають рівні:

- фізіологія (вода, їжа, житло, сон);
- безпека (особиста, здоров'я, стабільність),
- приналежність (спілкування, дружба, любов),
- визнання (повага оточуючих, самооцінка),
- самовираження (вдосконалення, персональний розвиток).

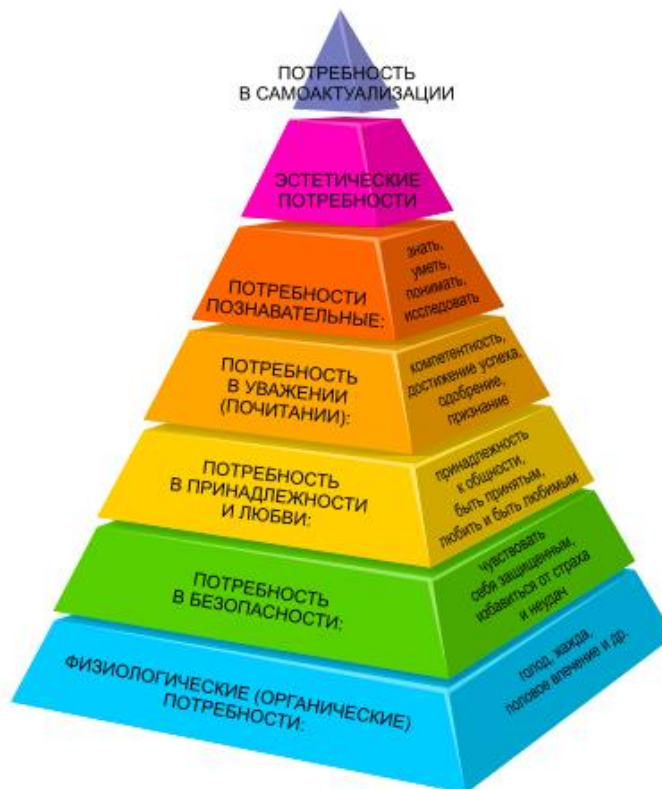


Рисунок 1.1 – Піраміда потреб Маслоу

Даний програмний продукт задовольняє потреби рівня пізнання в піраміді Маслоу.

### 1.1.2 Деталізація матеріальної потреби

Для деталізації матеріальних потреб використовують MindMap.

MindMap – це спосіб структуризації концепцій з використанням графічного запису в вигляді діаграми зв'язків в деревоподібній формі. MindMap цього ПП наведено на рис. 1.2.

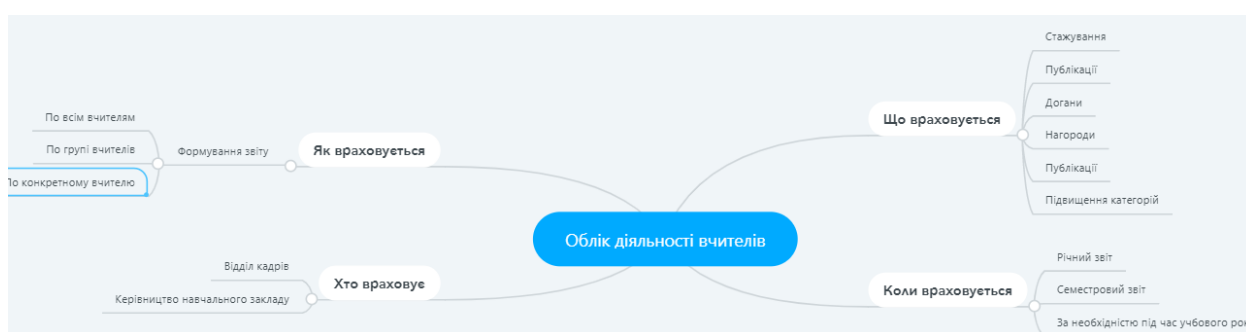


Рисунок 1.2 - Mindmap майбутнього ПП

## 1.2 Бізнес вимоги до ПП

### 1.2.1 Опис проблем користувача

#### 1.2.1.1 Концептуальний опис проблеми споживача

Відділ кадрів допускається багатьох помилок під час обліку викладачів навчального закладу та занадто довго виконує дані їм завдання.

#### 1.2.1.2 Метричний опис проблеми користувача

№	Параметр незадоволеності
1	Великий відсоток помилок під час роботи відділу кадрів

Таблиця 1.1 – Параметри незадоволеності



Відсоток помилок EP (EP – Error percent) можна визначити як

$$EP = E / N,$$

де

E – кількість помилок під час роботи;

N – загальна кількість роботи

## **1.2.2 Мета створення ПП**

### **1.2.2.1 Проблемний аналіз існуючих ПП**

<b>№</b>	<b>Назва</b>	<b>Вартість</b>	<b>Ступінь</b>	<b>Примітка</b>
1	uPortfolio	Безкоштовно	2	Немає можливості централізованого обліку усіх вчителів навчального закладу
2	Portfolio-edu	Безкоштовно	1	Немає можливості централізованого обліку усіх вчителів навчального закладу

Таблиця 1.2 – Аналіз існуючих ПП

### **1.2.2.2 Мета створення ПП**

Метою створення даного ПП є: зниження відсотка помилок під час роботи відділу кадрів в навчальних закладах шляхом впровадження програмного забезпечення з автоматизації обліку професійної діяльності вчителів.

## **1.2.3 Назва ПП**

### **1.2.3.1 Гасло ПП**

TrackTeacher – все про вчителів в одному місці.

### 1.2.3.2 Логотип ПП



Рисунок 1.3 – Логотип ПП

## 1.3 Вимоги користувача до ПП

### 1.3.1 Історія користувача ПП

- Користувач авторизується в системі
- Користувач може переглянути/змінити інформацію про себе в системі
- Адміністрація навчального закладу може переглянути інформацію про всіх користувачів системи
- Відділ кадрів може переглядати/редагувати інформацію про всіх користувачів системи
- Модератори можуть змінювати налаштування роботи системи

### 1.3.2 Діаграма прецедентів ПП

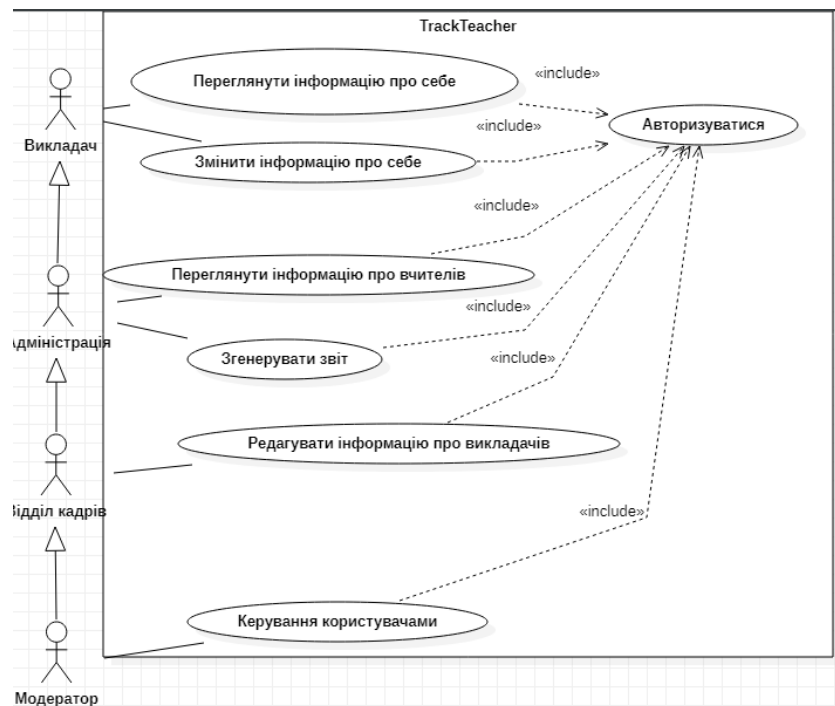


Рисунок 1.4 – Діаграма прецедентів ПП

### 1.3.3 Сценарії використання прецедентів ПП

Альтернативний сценарій для всіх прецедентів:

1. Користувач не має доступу до розділу системи
2. ПП виводить повідомлення про помилку

Прецедент «Авторизуватись»:

- ☞ Умова початку прецеденту: користувач зайшов на сайт неавторизованим
- ☞ Гарантії успіху: авторизація користувача
- ☞ Успішний сценарій:
  1. ПП надає форму для заповнення
  2. Користувач вводить дані(пошту та пароль)
  3. ПП приймає дані та авторизує користувача

☞ Альтернативний сценарій:

1. ПП не отримує дані від користувача або отримує невірні дані
2. ПП виводить повідомлення про помилку

Прецедент «Переглянути інформацію про себе»:

☞ Умова початку прецеденту: користувач авторизувався

☞ Актори: Користувач

☞ Гарантії успіху: користувач перегляне інформацію про себе

☞ Успішний сценарій:

1. Користувач переходить на сторінку профілю
2. ПП надає інформацію про користувача(його догани, нагороди, публікації, стажування і тд)

Прецедент «Змінити інформацію про себе»:

☞ Умова початку прецеденту: користувач авторизувався

☞ Актори: користувач

☞ Гарантії успіху: зміна інформації про користувача в системі

☞ Успішний сценарій:

1. ПП надає форму редагування
2. Користувач надає дані(телефон, ім'я, дату народження, адресу проживання, аватар)
3. ПП змінює інформацію про користувача

☞ Альтернативний сценарій:

1. ПП отримує невірні дані від користувача (дату чи телефон в неправильному форматі, файл аватару в невірному розширенні)
2. ПП виводить повідомлення про помилку

Прецедент «Переглянути інформацію про вчителів»:

- ☞ Умова початку прецеденту: користувач має права не нижче Адміністрації
- ☞ Актор: адміністрація
- ☞ Гарантії успіху: користувач системи отримає інформацію про вчителя
- ☞ Успішний сценарій:
  1. Користувач переходить на сторінку вчителя
  2. ПП надає інформацію про вчителя

#### Прецедент «Згенерувати звіт»:

- ☞ Умова початку прецеденту: користувач має права не нижче Адміністрації
- ☞ Актор: адміністрація
- ☞ Гарантії успіху: користувач системи отримає звіт
- ☞ Успішний сценарій:
  1. Користувач заповнює форму генерування звіту(за який період, для яких вчителів)
  2. ПП генерує звіт на основі інформації в системі та видає його
- ☞ Альтернативний сценарій:
  1. Користувач ввів неправильні дані (дати в невірному форматі, вибрав неіснуючого вчителя)
  2. ПП виводить повідомлення про помилку

#### Прецедент «Редагувати інформацію про вчителів»:

- ☞ Умова початку прецеденту: користувач має права не нижче Відділу кадрів
- ☞ Актор: відділ кадрів
- ☞ Гарантії успіху: користувач системи змінить інформацію про вчителя
- ☞ Успішний сценарій:
  1. Користувач переходить на сторінку редагування
  2. Користувач надає дані(догани, нагороди, стажування)
  3. ПП зберігає зміни

☞ Альтернативний сценарій:

1. Користувач ввів неправильні дані(дати в неправильному форматі, не заповнив обов'язкові поля)
2. ПП виводить повідомлення про помилку

Прецедент «Керування користувачами»:

☞ Умова початку прецеденту: користувач має права Модератора

☞ Актор: модератор

☞ Гарантії успіху: користувач змінить налаштування системи

☞ Успішний сценарій:

1. Користувач переходить на сторінку користувачів
2. Модератор змінює інформацію про користувачів(додає, видаляє, редагує)
3. ПП зберігає зміни

## 1.4 Функціональні вимоги до ПП

### 1.4.1 Багаторівнева класифікація функціональних вимог

Ідентифікатор	Опис
FR1	Авторизуватися
FR1.1	Створення запиту до користувача на отримання його параметрів аутентифікації
FR1.2	Передача параметрів аутентифікації на сервер
FR1.3	Перевірка наданих параметрів
FR1.4	Передача інформації про помилки на клієнт
FR2	Переглянути інформацію про себе
FR2.1	Надати користувачу інформацію про його профіль
FR3	Змінити інформацію про себе

Ідентифікатор	Опис
FR3.1	Створення запиту до користувача на отримання змін до його профілю
FR3.2	Передача наданих даних на сервер
FR3.3	Перевірка наданих даних на коректність
FR3.4	Передача інформації про помилки на клієнт
FR4	Переглянути інформацію про вчителів
FR4.1	Надати користувачу інформацію про вчителя
FR5	Згенерувати звіт
FR5.1	Створення запиту до користувача на отримання налаштувань звіту
FR5.2	Передача даних на сервер
FR5.3	Перевірка даних на коректність
FR5.4	Генерація звіту по заданим параметрам
FR5.5	Передача звіту користувачу
FR6	Редагувати інформацію про викладача
FR6.1	Створення запиту до користувача на отримання нових даних для викладача
FR6.2	Передача даних на сервер
FR6.3	Збереження змін

Таблиця 1.4 – Функціональні вимоги до ПП

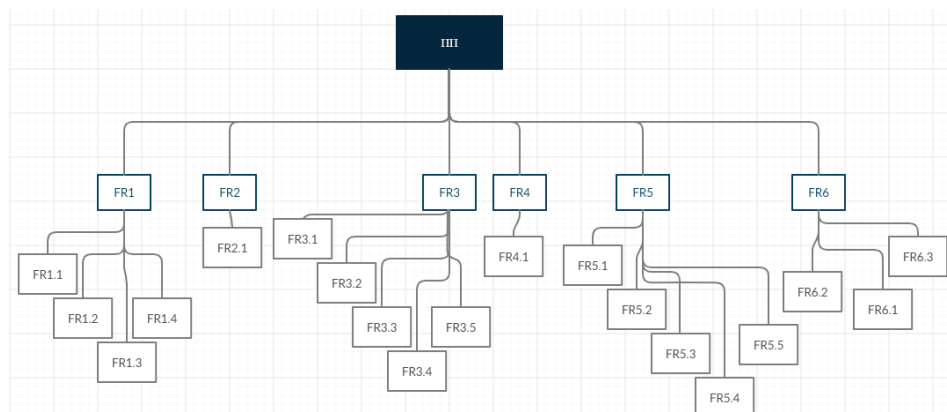


Рисунок 1.5 – WBS структура вимог

## 1.4.2 Функціональний аналіз існуючих ПП

Ідентифікатор	uPortfolio	Portfolio-edu
FR1	+	+
FR2	+	+
FR3	+	+
FR4	-	-
FR5	-	-
FR6	-	-

Таблиця 1.5 – Функціональний аналіз існуючих ПП

## 1.5 Нефункціональні вимоги до ПП

### 1.5.1 Опис зовнішніх інтерфейсів

#### 1.5.1.1 Опис інтерфейса користувача

##### 1.5.1.1.1 Опис INPUT-інтерфейса користувача

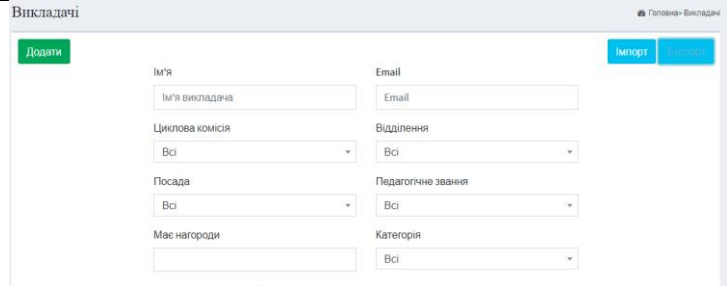
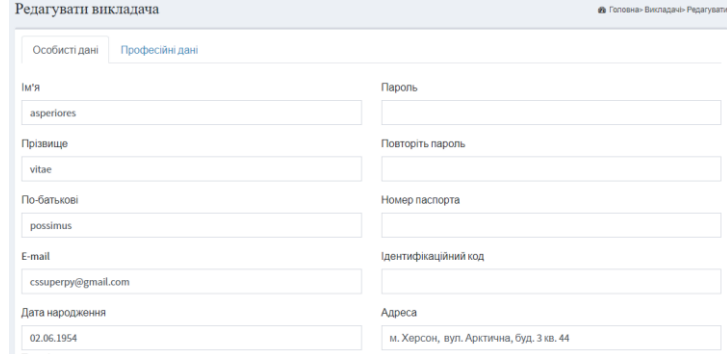
Ідентифікатор	Засіб INPUT	Особливості
FR1.1	Стандартна клавіатура, маніпулятор типу «миша»	
FR2.1	Маніпулятор типу «миша»	Використання колеса миші для завершення процесу вводу даних
FR3.1	Стандартна клавіатура, маніпулятор типу «миша»	Використання колеса миші для завершення процесу вводу даних
FR4.1	Маніпулятор типу «миша»	
FR5.1	Стандартна клавіатура, маніпулятор типу «миша»	Використання колеса миші для завершення процесу вводу даних
FR6.1	Клавіатура, маніпулятор типу «миша»	Використання колеса миші для завершення процесу вводу даних

Таблиця 1.5 – INPUT-інтерфейси



### 1.5.1.1.2 Опис OUTPUT-потоків

Ідентифікатор	Засіб OUTPUT-потоків	Особливості
FR1.1	Графічний інтерфейс	<div> <div>Авторизація</div> <div> <div>Email*</div> <div></div> <div>Пароль*</div> <div></div> </div> <div>Ввійти</div> </div>
FR1.4	Графічний інтерфейс	<div> <div>Авторизація</div> <div> <div>Email or password are not match</div> <div> <div>Email*</div> <div></div> <div>Пароль*</div> <div></div> </div> <div>Ввійти</div> </div> </div>
FR2.1	Графічний інтерфейс	<div> <div>Профіль користувача vitae asperiores possimus</div> <div> <div>50 x 50</div> <div> <div>Ім'я: vitae asperiores possimus</div> <div>Дата народження: 02.06.1954</div> <div>Email: cssuperpy@gmail.com</div> <div>Адреса: м. Херсон, вул. Арктична, буд. 3 кв. 44</div> <div>Роль: Адміністратор</div> <div>Відділ: Не встановлено</div> <div>Циклова комісія: Не встановлено</div> <div>Категорія:</div> <div>Розряд: Не встановлено</div> <div>Педагогічне звання: Немає</div> <div>Стаж: 0</div> </div> </div> </div>
FR3.1	Графічний інтерфейс	<div> <div>Редагувати профіль</div> <div> <div> <div>Е-mail</div> <div>cssuperpy@gmail.com</div> <div>Пароль</div> <div></div> <div>Повторіть пароль</div> <div></div> <div>Телефон</div> <div></div> </div> <div> <div>Дата народження</div> <div>02.06.1954</div> <div>Номер паспорта</div> <div></div> <div>Ідентифікаційний код</div> <div></div> <div>Адреса</div> <div>м. Херсон, вул. Арктична, буд. 3 кв. 44</div> </div> <div> <div>Аватар</div> <div>50 x 50</div> <div>Выберите файл</div> <div>Файл не выбран</div> </div> </div> </div>
FR4.1	Графічний інтерфейс	<div> <div>Профіль користувача vitae asperiores possimus</div> <div> <div>50 x 50</div> <div> <div>Ім'я: vitae asperiores possimus</div> <div>Дата народження: 02.06.1954</div> <div>Email: cssuperpy@gmail.com</div> <div>Адреса: м. Херсон, вул. Арктична, буд. 3 кв. 44</div> <div>Роль: Адміністратор</div> <div>Відділ: Не встановлено</div> <div>Циклова комісія: Не встановлено</div> <div>Категорія:</div> <div>Розряд: Не встановлено</div> <div>Педагогічне звання: Немає</div> <div>Стаж: 0</div> </div> </div> </div>

FR5.1	Графічний інтерфейс	
FR6.1	Графічний інтерфейс	

Таблиця 1.6 – Засоби OUTPUT-потоків

#### 1.5.1.2 Опис інтерфейсу з зовнішніми пристроями

Ідентифікатор функції	Зовнішній пристрій
FR 1.1 – FR 1.4	Смартфон, планшет, Desktop-персональний комп'ютер, Notebook;
FR 2.1	
FR 3.1 – FR 3.4	
FR 4.1	
FR 5.1 – 5.5	
FR 6.1 – 6.3	

Таблиця 1.7 – Опис інтерфейсу з зовнішніми пристроями

#### 1.5.1.3 Опис програмних інтерфейсів

Версії операційних систем та програмних бібліотек, які знадобляться при реалізації більшості функцій ПП.

- Linux
- Windows

- Android
- IOS
- Підтримка браузерів з html 5 та JS
- Apache Server
- PHP

#### **1.5.1.4 Опис інтерфейсів передачі інформації**

Опис інтерфейсів передачі інформації, які знадобляться при реалізації більшості функцій ПП.

- Провідні інтерфейси:
  - Ethernet
- Безпроводні інтерфейси:
  - Wi-Fi;

#### **1.5.1.5 Опис атрибутів продуктивності**

Ідентифікатор	Максимальний час реакції, мс
FR1	2.5
FR1.1	1
FR1.2	0.7
FR1.3	0.4
FR1.4	0.4
FR2	1.5
FR2.1	1.5
FR3	2.5
FR3.1	1

<b>Ідентифікатор</b>	<b>Максимальний час реакції, мс</b>
FR3.2	0.8
FR3.3	0.3
FR3.4	0.4
FR4	2
FR4.1	2
FR5	3
FR5.1	1
FR5.2	0.5
FR5.3	0.3
FR5.4	1
FR5.5	0.2
FR6	2
FR6.1	1
FR6.2	0.6
FR6.3	0.4

Таблиця 1.8 – Опис атрибутів продуктивності

## 2 Планування процесу розробки програмного продукту

### 2.1 Планування ітерацій розробки програмного продукту

Ідентифікатор функції	Функціональні залежності	Вплив на досягнення мети, %	Пріоритет
F1.1	-	75	M
F1.2	-	75	M
F1.3	-	75	M
F1.4	-	75	M
F2.1	F1	25	S
F3.1	F1	25	S
F3.2	F1	25	S
F3.3	F1	25	S
F3.4	F1	25	S
F4.1	F1	80	M
F5.1	F1	60	S
F5.2	F1	60	S
F5.3	F1	60	S
F5.4	F1	60	S
F5.5	F1	60	S
F6.1	F1	30	C
F6.2	F1	30	C
F6.3	F1	30	C

Таблиця 2.1 – Планування ітерацій розробки

### 2.2 Концептуальний опис архітектури програмного продукту

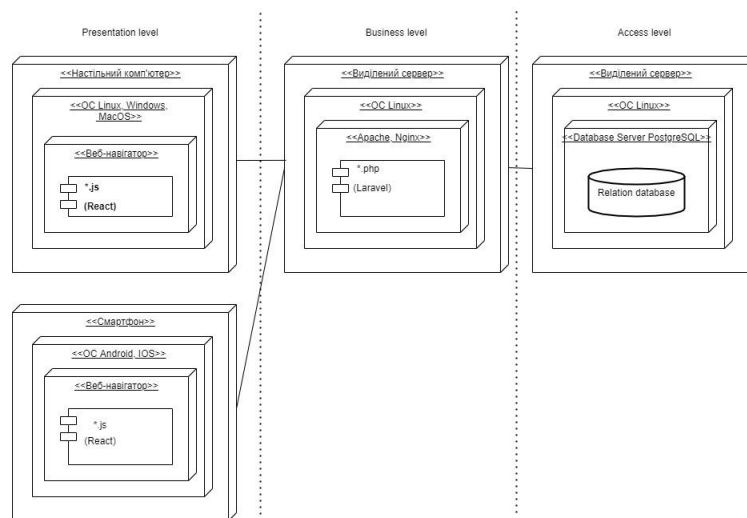


Рисунок 2.1 – Концептуальний опис архітектури ПП

## 2.3 План розробки ПП

### 2.3.1 Оцінка трудомісткості розробки ПП

#### 1. Визначення вагових показників акторів

Актор	Ваговий коефіцієнт
Викладач	2
Адміністрація	2
Відділ кадрів	3
Модератор	3

Таблиця 2.2 – Вагові коефіцієнти акторів

$$A = 2 + 2 + 3 + 3 = 10$$

#### 2. Визначення вагових показників прецедентів UC

Прецедент	Кількість кроків	Ваговий коефіцієнт
Авторизація	$\leq 3$	5
Переглянути інформацію про себе	$\leq 3$	5
Редагувати профіль	$\leq 3$	5
Переглянути інформацію про вчителя	$\leq 3$	5
Згенерувати звіт	4-7	10
Редагувати інформацію про вчителя	4-7	10
Керувати користувачами	4-7	10

Таблиця 2.3 – Вагові показники прецедентів

$$UC = 5 * 4 + 3 * 10 = 50$$

#### 3. Визначення UUCP

$$UUCP = A + UC = 10 + 50 = 60$$

#### 4. Визначення технічної складності проекту

Показники	Опис	Вага	ST
T1	Розподілена система	2	3
T2	Висока продуктивність	1	3
T3	Робота в режимі онлайн	1	3
T4	Складна обробка даних	-1	3
T5	Повторне використання коду	1	3
T6	Простота встановлення	0.5	4
T7	Переносимість	2	5
T8	Простота внесення змін	1	4
T9	Паралелізм	1	1
T10	Простота використання	0.5	5
T11	Спеціальні вимоги до безпеки	1	4
T12	Доступ до системи зовнішніми користувачами	1	1
T13	Спеціальні вимоги до навчання користувачів	1	2

Таблиця 2.4 – Визначення технічної складності проекту

$$TCF = 0.6 + (0.01 * (3*2 + 3*1 + 3*1 + 3*(-1) + 3*1 + 4*0.5 + 5*2 + 4*1 + 1*1 + 5*0.5 + 4*1 + 1*1 + 2*1)) = 0.985$$

#### 5. Визначення рівня кваліфікації розробників

Показник	Опис	Вага	SF
F1	Знання технологій	1.5	4
F2	Досвід розробки	0.5	3
F3	Досвід використання ООП	1	3
F4	Наявність аналітика	0.5	5
F5	Мотивація	1	5

F6	Стабільність вимог	2	2
F7	Часткова занятість	-1	2
F8	Складні мови розробки	-1	2

Таблиця 2.5 – Визначення рівня кваліфікації розробників

$$EF = 1.4 + (-0.03 * (1.5 * 4 + 0.5 * 3 + 3*1 + 5*0.5 + 5*1 + 2*2 + 2*(-1) + 2*(-1))) = 0.86$$

## 6. Визначення UCP

$$UCP = UUCP * TCF * EF = 60 * 0.985 * 0.86 = 50.83$$

## 7. Оцінка трудомісткості проекту

$$\text{Трудомісткість} = UCP * 20 = 1016.52 \text{ люд\год}$$

### 2.3.2 Визначення дерева робіт з розробки ПП

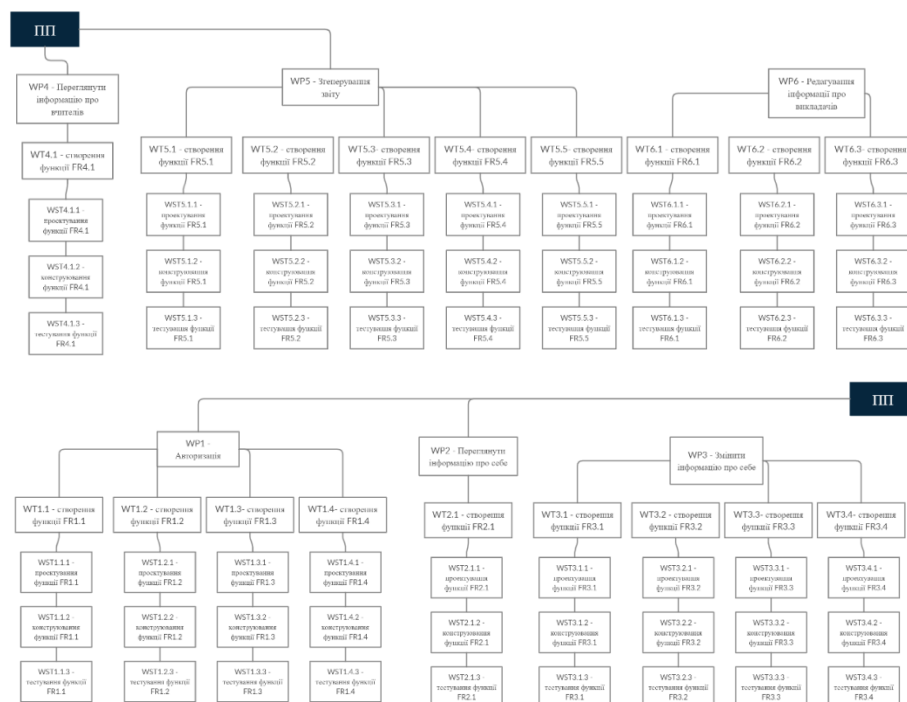


Рисунок 2.2 – WBS дерево робіт



Підзадача	Виконавець
WST1.*.*-WST6.*.*	Присяжний Ю. О.

Таблиця 2.6 – Опис підзадач з виконавцями

### 2.3.3 Графік робіт з розробки ПП

#### 2.3.3.1 Таблиця з графіком робіт

1.1.1	01.10.2020	1	01.10.2020	Присяжний Ю. О.
1.1.2	01.10.2020	2	02.10.2020	Присяжний Ю. О.
1.1.3	01.10.2020	1	01.10.2020	Присяжний Ю. О.
1.2.1	03.10.2020	2	04.10.2020	Присяжний Ю. О.
1.2.2	03.10.2020	1	03.10.2020	Присяжний Ю. О.
1.2.3	03.10.2020	1	03.10.2020	Присяжний Ю. О.
1.3.1	04.10.2020	1	04.10.2020	Присяжний Ю. О.
1.3.2	04.10.2020	2	05.10.2020	Присяжний Ю. О.
1.3.3	05.10.2020	1	05.10.2020	Присяжний Ю. О.
1.4.1	06.10.2020	1	06.10.2020	Присяжний Ю. О.
1.4.2	07.10.2020	2	08.10.2020	Присяжний Ю. О.
1.4.3	08.10.2020	1	08.10.2020	Присяжний Ю. О.
2.1.1	09.10.2020	1	09.10.2020	Присяжний Ю. О.
2.1.2	09.10.2020	2	10.10.2020	Присяжний Ю. О.
2.1.3	10.10.2020	1	10.10.2020	Присяжний Ю. О.
3.1.1	11.10.2020	1	11.10.2020	Присяжний Ю. О.
3.1.2	11.10.2020	2	12.10.2020	Присяжний Ю. О.
3.1.3	12.10.2020	1	12.10.2020	Присяжний Ю. О.

Рисунок 2.3 – Графік робіт

#### 2.3.3.2 Діаграма Ганта

1.1.1	01.10.2020	1	01.10.2020	Присяжний Ю. О.															
1.1.2	01.10.2020	2	02.10.2020	Присяжний Ю. О.															
1.1.3	01.10.2020	1	01.10.2020	Присяжний Ю. О.															
1.2.1	03.10.2020	2	04.10.2020	Присяжний Ю. О.															
1.2.2	03.10.2020	1	03.10.2020	Присяжний Ю. О.															
1.2.3	03.10.2020	1	03.10.2020	Присяжний Ю. О.															
1.3.1	04.10.2020	1	04.10.2020	Присяжний Ю. О.															
1.3.2	04.10.2020	2	05.10.2020	Присяжний Ю. О.															
1.3.3	05.10.2020	1	05.10.2020	Присяжний Ю. О.															
1.4.1	06.10.2020	1	06.10.2020	Присяжний Ю. О.															
1.4.2	07.10.2020	2	08.10.2020	Присяжний Ю. О.															
1.4.3	08.10.2020	1	08.10.2020	Присяжний Ю. О.															
2.1.1	09.10.2020	1	09.10.2020	Присяжний Ю. О.															
2.1.2	09.10.2020	2	10.10.2020	Присяжний Ю. О.															
2.1.3	10.10.2020	1	10.10.2020	Присяжний Ю. О.															
3.1.1	11.10.2020	1	11.10.2020	Присяжний Ю. О.															
3.1.2	11.10.2020	2	12.10.2020	Присяжний Ю. О.															
3.1.3	12.10.2020	1	12.10.2020	Присяжний Ю. О.															

Рисунок 2.4 – Діаграма Ганта



## 3.2 Проектування програмних класів

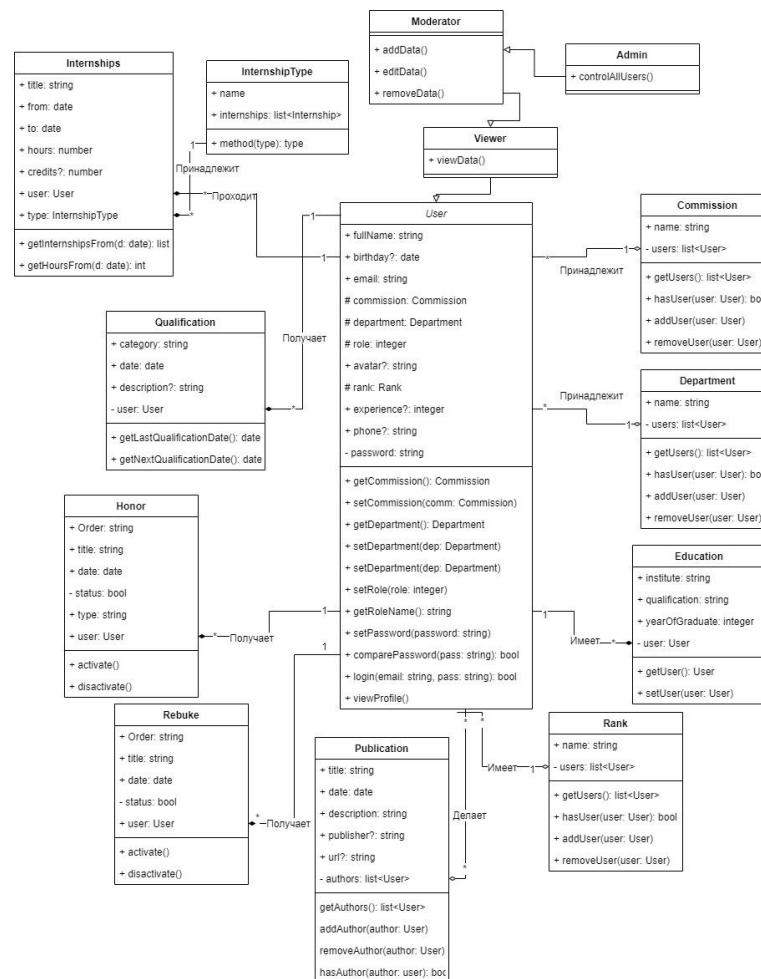


Рисунок 3.3 – Проектування програмних класів

## 3.3 Проектування алгоритмів роботи методів програмних класів

AddUser.puml

@startuml

start

:Выводится форма добавления учителя;

:Пользователь вводит информацию о учителе(отдел, комиссию, ФИО, email, пароль);

if(Данные корректны (Выбран существующий отдел и комиссия)) then (yes)

```

floating note left: Поиск в таблице users

if(Существует пользователь с таким email?) then (yes)

    floating note left: Пользователь с таким email уже существует

    #pink:Вывод ошибок;

    kill

else (no)

    :Создается пользователь;

endif

else (no)

    floating note right: Некорректные данные(отдел, комиссия)

    #pink:Вывод ошибок;

    kill

endif

stop

@enduml

```



Рисунок 3.4 – Діаграма активностей для «Додати користувача»

EditProfile.puml

```

@startuml

start

:Выводится форма редактирования профиля;

:Пользователь вводит новые данные о себе(email, пароль, дата рождения, адрес, телефон);

if(Данные корректны (Правильный формат даты, email и пароль)) then (yes)

```

```
floating note left: SELECT * FROM `users` WHERE `user_id` = __id__
```

```
if(Существует пользователь с таким email?) then (yes)
```

```
#pink:Вывод ошибок;
```

```
kill
```

```
else
```

```
:Сохраняем информацию;
```

```
endif
```

```
else
```

```
#pink:Вывод ошибок;
```

```
kill
```

```
endif
```

```
stop
```

```
@enduml
```

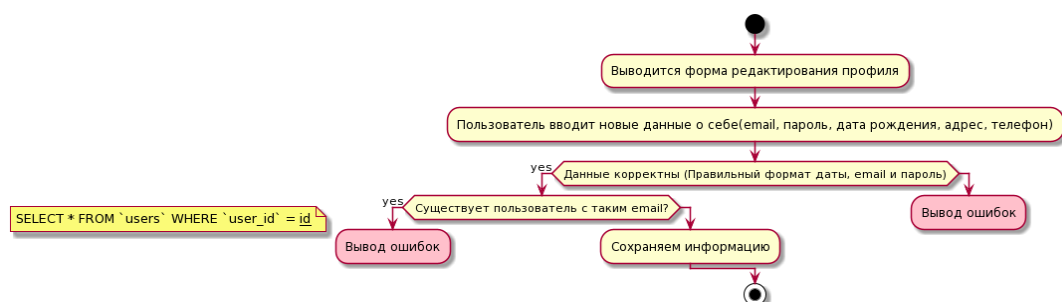


Рисунок 3.5 – Діаграма активностей для «Редагувати профіль»

Login.puml

```
@startuml
```

```
start
```

```
:Выводится форма логина;
```

```
:Пользователь вводит информацию(email и пароль);
```

```
if(Валидация данных(корректный email, пароль)) then (yes)
```

```
floating note left: SELECT * FROM `users` \nWHERE `email` = __email__ AND `password` =  
__password__
```

```

if(Поиск пользователя в базе) then (exists)
    :Авторизировать пользователя;
else (not exists)
    floating note right: Пользователь не существует
    #pink:Вывод ошибки;
    kill
endif
else (no)
    floating note right: Некорректные данные
    #pink:Вывод ошибки;
    kill
endif

stop
@enduml

```

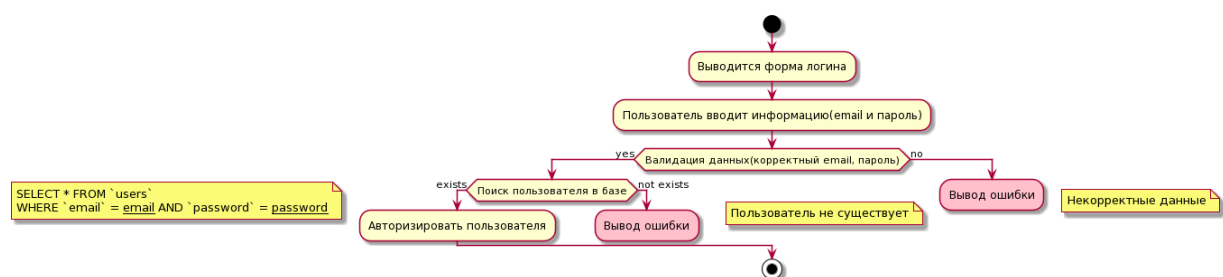


Рисунок 3.6 – Диаграмма активностей для «Логін користувача»

## ViewProfile.puml

```

@startuml
start

floating note left: SELECT * FROM `users` WHERE `token` = __token__;
if(Пользователь авторизован?) then (yes)

```

```
floating note left: SELECT * FROM `internships` WHERE `user_id` = __user_id__; \n\nSELECT
`publications`. * FROM `publications` INNER JOIN `publications_users` \nON id = publication_id WHERE
user_id = __user_id__; \n\nSELECT * FROM `qualifications` WHERE `user_id` = __user_id__;
```

:Показать информацию о пользователе\n(стажировки, публикации, повышения квалификаций);

else (no)

#pink:Показать 403 ошибку;

kill

endif

stop

@enduml

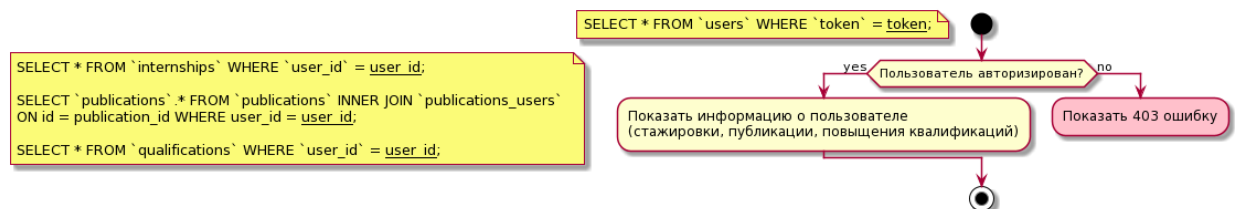


Рисунок 3.7 – Диаграмма активностей для «Перегляд профілю»

### 3.4 Проектування тестових наборів методів програмних класів

Назва функції	№ тесту	Вхідні значення	Опис очікуваних результатів
Авторизация	1	Ввод валидных данных существующих в БД. Пример: Email: test@gmail.com Пароль: 12345678	Токен пользователя
	2	Ввод неправильного email  Пример: bademail, test@gmail, test@32323	Сообщения об ошибке «Неправильный формат email»
	3	Email пуст	Сообщения об ошибке «Email обязательное для заполнения поле»
	4	Пароль пуст	Сообщения об ошибке «Пароль обязательное для заполнения





## **4 Конструювання ПП**

### **4.1 Особливості конструювання структур даних**

#### **4.1.1 Особливості інсталяції та роботи з СУБД**

В даному ПП використовувалася СКБД PostgreSQL версії 12.

Встановлення на Ubuntu:

##### **1) Встановлення**

```
sudo apt update
```

```
sudo apt install postgresql postgresql-contrib
```

##### **2) Перехід на запис postgres**

```
sudo -i -u postgres
```

##### **3) Запуск Postgres**

```
Psql
```

#### **4.1.2 Особливості створення структур даних**

- Таблиця «commissions»

```
CREATE TABLE `commissions`(  
    `id` SERIAL,  
    `name` VARCHAR(255) NOT NULL  
);
```

- Таблица «users»

```
CREATE TABLE `users`(  
    `id` SERIAL,  
    `fullName` VARCHAR(255) NOT NULL,  
    `birthday` DATE,  
    `email` VARCHAR(255) NOT NULL UNIQUE,  
    `password` VARCHAR(255) NOT NULL,  
    `commission_id` INT NOT NULL FOREIGN KEY REFERENCES `commissions`(`id`),  
    `department_id` INT NOT NULL FOREIGN KEY REFERENCES `departments`(`id`),  
    `role` INT NOT NULL DEFAULT 50,  
    `avatar` VARCHAR(255)  
);
```

- Таблица «internships»

```
CREATE TABLE `internships`(  
    `id` SERIAL,  
    `category_id` INT NOT NULL FOREIGN KEY REFERENCES `categories`(`id`),  
    `user_id` INT NOT NULL FOREIGN KEY REFERENCES `user`(`id`),  
    `place` VARCHAR(255),  
    `title` VARCHAR(255) NOT NULL,  
    `from` DATE NOT NULL,  
    `to` DATE NOT NULL,  
    `hours` INT  
);
```

- Таблиця «honors»

```
CREATE TABLE `honors`(  
    `id` SERIAL,  
    `order` VARCHAR(255) NOT NULL UNIQUE,  
    `date_presentation` DATE NOT NULL,  
    `title` VARCHAR(255) NOT NULL,  
    `active` BOOLEAN DEFAULT 0,  
    `user_id` INT FOREIGN KEY REFERENCES `users`(`id`)  
);
```

## **4.2 Особливості конструювання програмних модулів**

### **4.2.1 Особливості роботи з інтегрованим середовищем розробки**

Даний ПП був розроблений в IDE PHPStorm. PHPStorm – це інтегрована среда розробки на мові PHP. Розробляється компанією JetBrains на основі платформи IntelliJ IDEA.

PHPStorm це інтелектуальний редактор для розробки на PHP, JS, HTML з можливостями аналізу коду та рефакторінгу для PHP та JS. Він підтримує останні специфікації мови PHP. Має інтегрований SQL-редактор з можливістю виконання запитів до БД.

Також існує велика бібліотека плагінів для PHPStorm за допомогою яких користувачі можуть розширити можливості редактора.

#### **4.2.2 Особливості створення програмної структури з урахуванням спеціалізованого фреймворку**

Розробка даного ПП велась за допомогою фреймворку Laravel(бекенд) та React(фронтенд).

Laravel – це безкоштовний веб-фреймворк з відкритим кодом, створений для розробки з використанням моделі MVC. В якості ORM використовується Eloquent, шаблонізатор – Blade.

React – це JS-бібліотека для розробки UI. Розробляється Facebook та Instagram. Може використовуватися для розробки SPA додатків або мобільної розробки. Його мета – представити високу швидкість та простоту масштабування фронтенду.

#### **4.2.3 Особливості створення програмних класів**

User.php

```
<?php
```

```
namespace App\Models;
```

```
use Illuminate\Database\Eloquent\Factories\HasFactory;
```

```
use Illuminate\Foundation\Auth\User as Authenticatable;
```

```
use Illuminate\Notifications\Notifiable;
```

```
use Laravel\Passport\HasApiTokens;
```

```
class User extends Authenticatable
```

```

{

    use Notifiable, HasApiTokens, HasFactory;

    /**
     * Свойства которые можно заполнить напрямую через функцию fill
     *
     * @var array
     */
    protected $fillable = [

        'fullName', 'email', 'birthday', 'pedagogical_title', 'address', 'phone',

        'hiring_year', 'experience', 'academic_status_year',

        'scientific_degree_year'

    ];

    /**
     * Свойства, которые можно заполнить только через setter/getter
     *
     * @var array
     */
    protected $hidden = [

        'password', 'remember_token',

    ];

    //Отношения с другими моделями

    public function department(){

        return $this->belongsTo(Department::class);

    }

```

```
public function commission(){  
  
    return $this->belongsTo(Commission::class);  
  
}
```

```
public function publications(){  
  
    return $this->belongsToMany(Publication::class, 'users_publications', 'user_id');  
  
}
```

```
public function internships(){  
  
    return $this->hasMany(Internship::class);  
  
}
```

```
public function qualifications(){  
  
    return $this->hasMany(Qualification::class);  
  
}
```

```
public function honors(){  
  
    return $this->hasMany(Honor::class);  
  
}
```

```
public function rebukes(){  
  
    return $this->hasMany(Rebuke::class);  
  
}
```

```
public function rank(){  
  
    return $this->belongsTo(Rank::class);  
  
}
```

```

public function educations(){

    return $this->hasMany(Education::class);

}

```

//Помогающие методы

//Преобразовать день рождения к строке

```

public function getBirthdayString(){

    if($this->birthday)

        return $this->birthday;

    else

        return __('messages.notSetted');

}

```

```

public function setRole(int $role){

    //if this role id in roles array then set it to user otherwise return

    if(array_search($role, \Constants::$roles) !== false)

        $this->role = $role;

}

```

//Convert role to string

```

public function getRoleString(): string {

    return \Constants::$roles[$this->role];

}

```

//pedagogical title setter

```

public function setTitle(int $title){

    //if exists index in array of titles then change user

```

```

        if($title >= 0 && $title < sizeof(\Constants::$pedagogicalTitles))

            $this->pedagogical_title = $title;

    }

//convert pedagogical title code to string

    public function getTitle(): string {

        return \Constants::$pedagogicalTitles[$this->pedagogical_title];

    }

//academic status setter

    public function setAcademicStatus(int $status){

        //if exists index in array then change user

        if($status >= 0 && $status < sizeof(\Constants::$academicStatusList))

            $this->academic_status = $status;

    }

//convert status code to string

    public function getAcademicStatus(): string {

        return \Constants::$academicStatusList[$this->academic_status];

    }

    public function setScientificDegree(int $scientific){

        if($scientific >= 0 && $scientific < sizeof(\Constants::$scientificDegreeList))

            $this->scientific_degree = $scientific;

    }

    public function getScientificDegree(): string {

        return \Constants::$scientificDegreeList[$this->scientific_degree];

```



```

}

public function setDepartment($department){

    $this->department_id = $department;

}

public function getDepartmentID(){

    if($this->department)

        return $this->department->id;

}

public function getDepartmentName(){

    if($this->department)

        return $this->department->name;

    else

        return __('messages.notSetted');

}

public function setCommission($commission){

    $this->commission_id = $commission;

}

public function getCommissionID(){

    if($this->commission)

        return $this->commission->id;

}

public function getCommissionName(){

```

```

        if($this->commission)

            return $this->commission->name;

        else

            return __('messages.notSetted');

    }


    public function setRank($id){

        if($id)

            $this->rank_id = $id;

    }


    public function getRankID(){

        if($this->rank)

            return $this->rank->id;

    }


    public function getRankName(){

        if(!$this->rank)

            return __('messages.notSetted');

        return $this->rank->name;

    }


    public function getShortName(): string {

        $fullName = explode(' ', $this->fullName);

        //if we have only one word then return without cut

        if(sizeof($fullName) == 1){

```

```

        return $fullName[0];
    }

    else{

        //cut name and return

        list($name, $surname) = $fullName;

        return $surname . ' ' . mb_substr($name, 0, 1) . ' ';

    }

}

public function getAvatar(){

    if($this->avatar)

        return $this->avatar;

    else

        return env('APP_URL') . '/storage/avatars/noAva.jpg';

}

//generate secret values

public function generatePassword($password){

    if($password){

        $this->password = bcrypt($password);

    }

}

public function cryptPassport($passport){

    if(!$passport)

        return;

    $this->passport = encrypt($passport);

```

```

        $this->save();
    }

    public function cryptCode($code){

        if(!$code)

            return;

        $this->code = encrypt($code);

        $this->save();

    }

    public function getToken(bool $long = false){

        $token = $this->createToken(config('app.name'));

        return $token->accessToken;

    }

}

```

#### **4.2.4 Особливості розробки алгоритмів методів програмних класів або процедур/функцій**

##### **UserRepository.php**

```

class UserRepository extends BaseRepository implements UserRepositoryInterface
{

    private $avatarService;

    private $model = User::class;

```

```

private $sortFields = [

    'ID' => 'id',

    'name' => 'fullName',

    'email' => 'email'

];

public function __construct(PhotoUploader $avatarService)

{

    $this->avatarService = $avatarService;

}

public function createRules(array $inputData): array

{

    $rules = [];

    if($inputData['filterName'] ?? null)

        $rules[] = new LikeRule('fullName', $inputData['filterName']);

    if($inputData['filterEmail'] ?? null)

        $rules[] = new LikeRule('email', $inputData['filterEmail']);

    if($inputData['filterCommission'] ?? null)

        $rules[] = new EqualRule('commission_id', $inputData['filterCommission']);

    if($inputData['filterDepartment'] ?? null)

        $rules[] = new EqualRule('department_id', $inputData['filterDepartment']);

    if($inputData['filterRank'] ?? null)

```

```

        $rules[] = new EqualRule('rank_id', $inputData['filterRank']);

    if($inputData['filterTitle'] ?? null)

        $rules[] = new EqualRule('pedagogical_title', $inputData['filterTitle']);

    if($inputData['filterCategory'] ?? null) {

        //$rules[] = new EqualRule('pedagogical_title', $inputData['filterTitle']);

    }

    $rules = array_merge($rules, $this->createSortRules($inputData['sort'] ?? null, $this->sortFields));

    return $rules;

}

public function getModel(): Model

{

    return app($this->model);

}

public function all()

{

    return $this->getModel()->all();

}

public function getForCombo()

{

    return $this->getModel()->all('id', 'name', 'surname', 'patronymic');

}

public function getForExportList(): array

```

```

{
    $users = $this->getModel()->all('id', 'fullName')->toArray();

    return to_export_list($users);
}
}

```

## RebukeRepository.php

class RebukeRepository extends BaseRepository implements RebukeRepositoryInterface

```

{

    private $model = Rebuke::class;

    protected $sortFields = [

        'ID' => 'id',

        'title' => 'title',

        'datePresentation' => 'date_presentation'

    ];

    public function createRules(array $inputData): array

    {

        $rules = [];

        if($inputData['user_id'] ?? null)

            $rules[] = new EqualRule('user_id', $inputData['user_id']);

        if($inputData['filterUser'] ?? null)

            $rules[] = new EqualRule('user_id', $inputData['filterUser']);

        if($inputData['filterTitle'] ?? null)

```

```

        $rules[] = new LikeRule('title', $inputData['filterTitle']);

    if($inputData['filterFrom'] ?? null)

        $rules[] = new DateMoreRule('date_presentation', $inputData['filterFrom']);

    if($inputData['filterTo'] ?? null)

        $rules[] = new DateLessRule('date_presentation', $inputData['filterTo']);

    $rules = array_merge($rules, $this->createSortRules($inputData['sort'] ?? null, $this->sortFields));

    return $rules;
}

public function getModel(): Model
{
    return app($this->model);
}

public function all()
{
    return $this->getModel()->all();
}

public function paginateForUser($user_id, ?int $size = null)
{
    $size = $size ?? config('app.PAGINATE_SIZE', 10);

    return $this->getModel()->query()->where('user_id', $user_id)->paginate($size);
}

```



```

public function getUserString(int $user_id): string
{
    //get all rebukes

    $rebukes = $this->getModel()->query()->where('user_id', $user_id)->get();

    //parse string

    $rebukesString = $rebukes->reduce(function(string $acc, $item){
        return $acc . implode(', ', [$item->title, $item->date_presentation, $item->order]) . ';';
    }, "");

    //return info

    return $rebukesString ? $rebukesString : 'Немає інформації';
}
}

```

## PublicationRepository.php

```

class PublicationRepository extends BaseRepository implements PublicationRepositoryInterface
{
    private $model = Publication::class;

    protected $sortFields = [
        'ID' => 'id',
        'title' => 'title',
        'date' => 'date_of_publication'
    ];

    public function getModel(): Model
    {

```

```

        return app($this->model);
    }

    public function createRules(array $inputData): array
    {
        $rules = [];

        if($inputData['user_id'] ?? null)
        {
            $rules[] = new HasAssociateRule('authors',
                new EqualRule('users.id', $inputData['user_id']));
        }

        if($inputData['filterTitle'] ?? null)
        {
            $rules[] = new LikeRule('title', $inputData['filterTitle']);
        }

        if($inputData['filterFrom'] ?? null)
        {
            $rules[] = new DateMoreRule('date_of_publication', $inputData['filterFrom']);
        }

        if($inputData['filterTo'] ?? null)
        {
            $rules[] = new DateLessRule('date_of_publication', $inputData['filterTo']);
        }

        $rules = array_merge($rules, $this->createSortRules($inputData['sort'] ?? null, $this->sortFields));

        return $rules;
    }

    public function all()
    {
        return $this->getModel()->all();
    }

```

```

public function paginateForUser($user_id, ?int $size = null)
{
    $size = $size ?? config('app.PAGINATE_SIZE', 10);

    return $this->getModel()->query()->whereHas('authors', function (Builder $q) use($user_id){
        $q->where('user_id', $user_id);
    }->paginate($size);
}
}

```

## 4.3 Тестування програмних модулів

### EditProfile.php

```
<?php
```

```
namespace Tests\Feature;
```

```
use App\Models\User;
```

```
use Illuminate\Foundation\Testing\RefreshDatabase;
```

```
use Illuminate\Foundation\Testing\WithFaker;
```

```
use Illuminate\Support\Facades\Auth;
```

```
use Laravel\Passport\Passport;
```

```
use Mockery\Generator\StringManipulation\Pass\Pass;
```

```
use Tests\TestCase;
```

```
class EditProfileTest extends TestCase
```

```
{
```

```
    /**
```

```
     * Тестируем удачное редактирование профиля
```

```
    */
```

```

public function testSuccess()
{
    //Авторизация пользователя для laravel-passport
    $user = User::query()->first();
    Passport::actingAs($user);

    //Данные для авторизации
    $data = [
        'email' => $user->email,
        'address' => 'Test address'
    ];

    $response = $this->postJson('/api/editMe', $data);

    //Проверка результата(200 статус и пользователь с обновленными данными)
    $response->assertSuccessful();
    $this->assertArrayHasKey( 'newUser', $response);
    $this->assertEquals($response['newUser']['address'], 'Test address');
}

/**
 * Тестируем пустой ввод
 */
public function emptyTest()
{
    //Авторизация пользователя для laravel-passport
    $user = User::query()->first();
    Passport::actingAs($user);

    $response = $this->postJson('/api/editMe', []);

    //Проверка результата(Ошибка обязательных полей)
    $response->assertJsonValidationErrors(['email']);
    $this->assertEquals($response['errors']['email'][0], "The email field is required.");
}

```

```

}

/**
 * Тестируем с не валидными данными
 */
public function testInvalid()
{
    //Авторизация пользователя для laravel-passport
    $user = User::query()->first();
    Passport::actingAs($user);

    $data = [
        'email' => 'bademail'
    ];

    $response = $this->postJson('/api/editMe', $data);

    //Проверяем ошибку валидации(почта неверного формата)
    $response->assertJsonValidationErrors(['email']);
    $this->assertEquals($response['errors']['email'][0], "The email must be a valid email address.");
}

/**
 * Тестируем неавторизованного пользователя
 */
public function testUnauthorized()
{
    $response = $this->postJson('/api/editMe', [
        'email' => 'test@gmail.com'
    ]);

    //Проверяем 403 статус
    $response->assertUnauthorized();
}
}

```

## LoginTest.php

```
<?php

namespace Tests\Feature;

use App\Models\User;
use Illuminate\Foundation\Testing\RefreshDatabase;
use Illuminate\Foundation\Testing\WithFaker;
use Tests\TestCase;

class LoginTest extends TestCase
{
    private $user;

    public function setUp(): void
    {
        parent::setUp(); // TODO: Change the autogenerated stub
        $this->user = User::query()->first();
    }

    /**
     * Тестируем удачный вход
     *
     * @return void
     */
    public function testSuccess()
    {
        $response = $this->postJson('/api/login', [
            'email' => $this->user->email,
            'password' => '20012007'
        ]);
    }
}
```

```

        //Проверяем 200 статус
        $response->assertSuccessful();
    }

    /**
     * Тестируем не валидную почту
     *
     * @return void
     */
    public function testInvalid()
    {
        $response = $this->postJson('/api/login', [
            'email' => 'bademail',
            'password' => '20012007'
        ]);

        //Проверяем ошибку валидации почты
        $response->assertJsonValidationErrors(['email']);
        $this->assertEquals($response['errors']['email'][0], "The email must be a valid email address.");
    }

    /**
     * Тестируем вход с пустой почтой
     *
     * @return void
     */
    public function testEmptyEmail()
    {
        $response = $this->postJson('/api/login', [
            'email' => null,
            'password' => '20012007'
        ]);

        //Проверяем ошибку, что почта это обязательное поле
        $response->assertJsonValidationErrors(['email']);
    }

```

```

        $this->assertEquals($response['errors']['email'][0], "The email field is required.");
    }

/**
 * Тестируем с пустым паролем
 *
 * @return void
 */
public function testEmptyPassword()
{
    $response = $this->postJson('/api/login', [
        'email' => $this->user->email,
        'password' => null
    ]);

    //Проверяем ошибку валидации(пароль - это обязательное поле)
    $response->assertJsonValidationErrors(['password']);
    $this->assertEquals($response['errors']['password'][0], "The password field is required.");
}

/**
 * Тестируем слишком короткий пароль
 *
 * @return void
 */
public function testShortPassword()
{
    $response = $this->postJson('/api/login', [
        'email' => $this->user->email,
        'password' => '11'
    ]);

    //Проверяем ошибку валидации данных(Пароль слишком короткий)
    $response->assertJsonValidationErrors(['password']);
}

```



```
$this->assertEquals($response['errors']['password'][0], "The password must be at least 8 characters.");
}

/**
 * Тестируем слишком длинный пароль
 *
 * @return void
 */
public function testLongPassword()
{
    $response = $this->postJson('/api/login', [
        'email' => $this->user->email,
        'password' => '11111111111111111111111111111111'
    ]);

    //Проверяем ошибку валидации данных(Пароль слишком длинный)
    $response->assertJsonValidationErrors(['password']);
    $this->assertEquals($response['errors']['password'][0], "The password may not be greater than 32 characters.");
}

/**
 * Тестируем ввод данных для несуществующего пользователя
 *
 * @return void
 */
public function testUnExistUser()
{
    $response = $this->postJson('/api/login', [
        'email' => 'test@mail.ru',
        'password' => '200120072017'
    ]);

    //Проверяем ошибку, что такая почта не существует
    $response->assertJsonValidationErrors(['email']);
    $this->assertEquals($response['errors']['email'][0], "The selected email is invalid.");
```

```
}  
}
```

## ReportTest.php

```
<?php
```

```
namespace Tests\Feature;  
use App\Models\User;  
use Illuminate\Foundation\Testing\RefreshDatabase;  
use Illuminate\Foundation\Testing\WithFaker;  
use Laravel\Passport\Passport;  
use Tests\TestCase;  
  
class ReportTest extends TestCase  
{  
    public function testUser()  
    {  
  
        //Авторизируем пользователя  
        $user = User::query()->where('role', 50)->first();  
        Passport::actingAs($user);  
  
        $response = $this->get('/api/report');  
  
        //Проверяем статус 403  
        $response->assertForbidden();  
    }  
  
    public function testMoreRole()  
    {  
        //Авторизируем пользователя  
        $user = User::query()->where('role', '<=', 30)->first();  
        Passport::actingAs($user);  
    }  
}
```

```

$response = $this->get('/api/report');

//Проверяем статус 200
$response->assertSuccessful();
}
}

```

## ViewProfile.php

```

<?php

namespace Tests\Feature;

use App\Models\User;
use Illuminate\Foundation\Testing\RefreshDatabase;
use Illuminate\Foundation\Testing\WithFaker;
use Laravel\Passport\Passport;
use Tests\TestCase;

class ViewProfileTest extends TestCase
{
    /**
     * Тестируем получения информации неавторизованным пользователем
     */
    public function testUnauthorized()
    {
        $response = $this->getJson ('/api/users/1');

        //Проверяем статус 401
        $response->assertUnauthorized();
    }

    /**
     * Test get info with role User

```

```

*/

public function testSmallRole()
{
    /**
     * @var User $user
     */

    //Авторизируем пользователя
    $user = User::query()->where('role', '50')->first()->getModel();
    Passport::actingAs($user);

    $response = $this->getJson('/api/users/1');

    //Проверяем статус 403
    $response->assertForbidden();
}

/**
 * Тестируем получения отчета пользователем с ролью больше или равной Просматриватель
 */

public function testSuccess()
{
    /**
     * @var User $user
     */

    //Авторизируем пользователя
    $user = User::query()->where('role', '<=', 30)->first();
    Passport::actingAs($user);

    $response = $this->getJson('/api/users/2');

    //Проверяем статус 200
    $response->assertSuccessful();
}
}

```

Для запуску тестів використовується команда:

```
php artisan test --env=testing,
```

де

php artisan test – команда для консольного помічника Laravel artisan.

--env=testing – вказання середовища виконання тестів(в даному випадку env).

```
$ php artisan test --env=testing
Warning: TTY mode is not supported on Windows platform.

PASS Tests\Feature\EditProfileTest
  success
  invalid
  unauthorized

PASS Tests\Feature\LoginTest
  success
  invalid
  empty email
  empty password
  short password
  long password
  un exist user

PASS Tests\Feature\ReportTest
  user
  more role

PASS Tests\Feature\ViewProfileTest
  unauthorized
  small role
  success

Tests: 15 passed
Time: 8.48s
```

Рисунок 1 – Результат тестирования

## 5 Розгортання та валідація ПП

### 5.1 Інструкція з встановлення ПП

Для встановлення ПП на своєму сервері необхідно мати Docker.

Встановлення цього інструменту можна подивитися в [офіційній документації докера](#).

Кроки встановлення ПП:

1) Клонувати репозиторій

```
git clone https://github.com/HTMLProgrammer2001/trackTeacherDocker.git ./trackteacher
```

2) Перейти в папку з ПП

```
cd trackteacher
```

3) Змінити файл налаштувань(необов'язково)

4) Запустити контейнери

```
docker-compose build && docker-compose up
```

5) Провести міграції та створити тестового адміністратора з email [admin@gmail.com](#) та паролем 12345678

```
docker-compose exec back setup.sh
```

### 5.2 Інструкція з використання ПП

Авторизация

Ошибка в данных

Email\*

test@gmail.com

Выбранный email не валидный.

Пароль\*

Войти

Рисунок 1 – Невдала авторизація

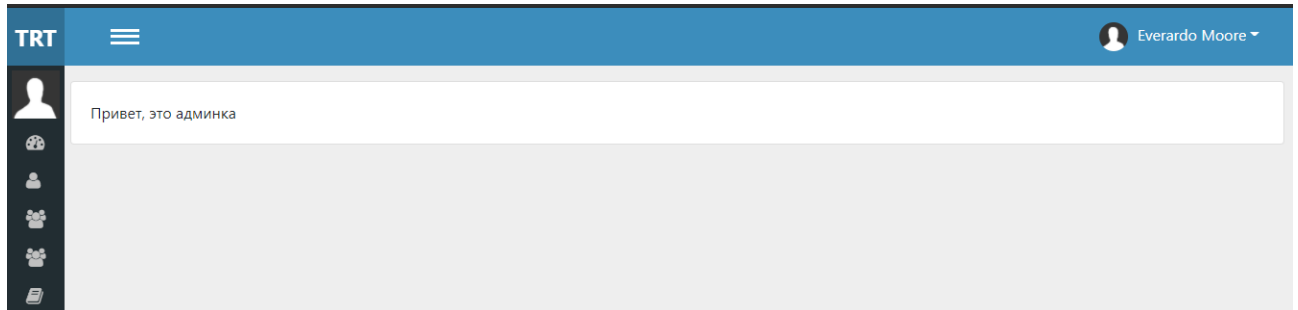


Рисунок 2 – Результат вдалого входу

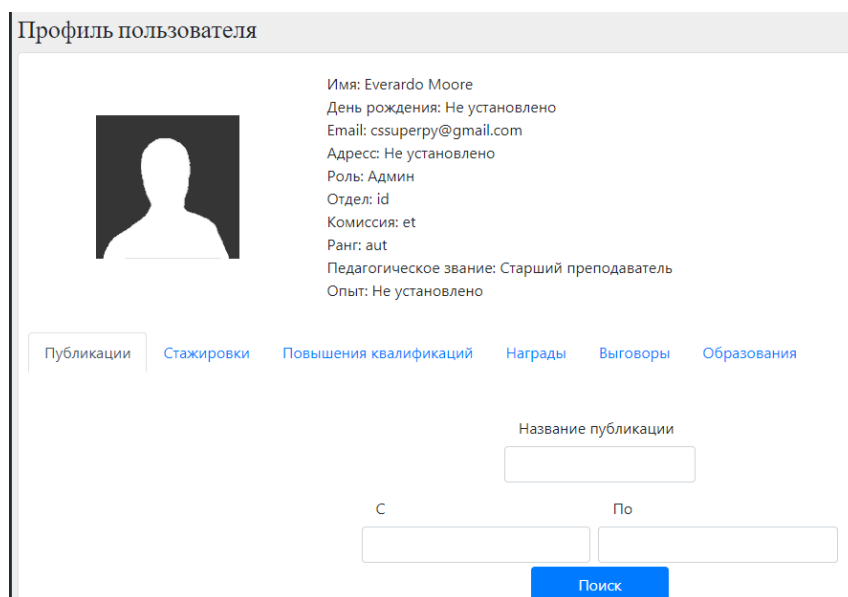


Рисунок 3 – Перегляд профілю

Авторизация

Email\*

Поле обязательно для заполнения

Пароль\*

Войти

Рисунок 4 – Результат перехода на профиль неавторизованым

Редактирование профиля

Email

Поле обязательно для заполнения

Новый пароль

Повторите пароль

Телефон

Адрес

Дата рождения

Перетащите или выберите аватарку

Назад

Сохранить

Рисунок 5 – Неправильні дані в редагуванні профілю

Профиль отредактирован

Редактирование профиля

Email

cssuperpy@gmail.com

Новый пароль

Повторите пароль

Телефон

Адрес

Дата рождения

15.12.2020

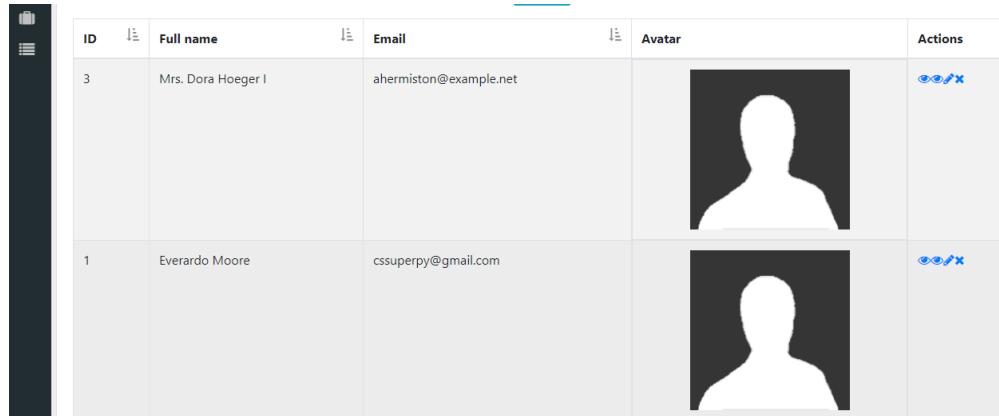
Перетащите или выберите аватарку

Назад

Сохранить



Рисунок 6 – Вдале редагування профілю











ID	Full name	Email	Avatar	Actions
3	Mrs. Dora Hoeger I	ahermiston@example.net		  
1	Everardo Moore	cssuperpy@gmail.com		  

Рисунок 7 – Перегляд користувачів

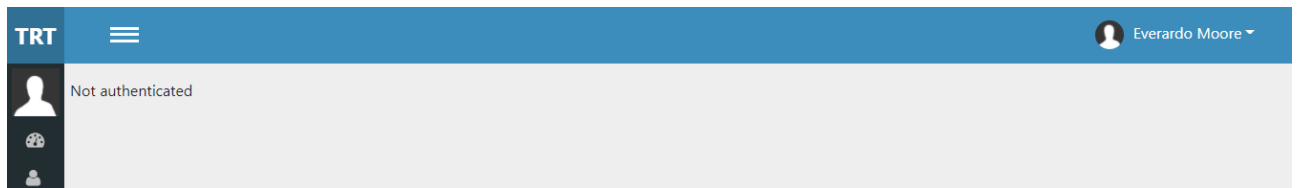


Рисунок 8 – Перегляд користувачів з правами нижче Переглядач

80	Hertha Hilpert	Casandra Gusikowski, 1981, 1;Kaylie Wiegand, 2017, 2;Helena Schimmel, 1980, 1;Prof. Eleanora Littel II, 1978, 1;Jeramie Heaney, 2007, 0;Prof. Demetrius Von MD, 1981, 0;	boris73@example.net, ,
81	Solon Predovic	Maci Koss, 1973, 1;Cooper Corwin, 1976, 0;Raphael Dickens, 2007, 2;	lvon@example.org, ,
82	Sibyl Steuber	Durward Altenwerth, 1982, 1;Kenyon Purdy, 1995, 2;Jensen Kuhn, 1990, 1;Nichole Haag, 1987, 2;	gino01@example.org, ,

Рисунок 9 – Звіт

### 5.3 Результати валідації ПП

Метою створення ПП було зменшення відсотка помилок під час роботи відділу кадрів.

Відсоток помилок EP (EP – Error percent) можна визначити як

$$EP = E / N,$$

де

E – кількість помилок під час роботи;

N – загальна кількість роботи

До введення ПП цей показник складав ~15%. Після введення ПП цей показник становить ~3%. На основі цієї можна сказати, що ПП пройшов валідацію й виконує свою мету.

## **Висновки**

В результаті створення програмного продукту була досягнута наступна мета його споживача: зменшення відсотку помилок під час роботи відділу кадрів.

Доказом цього є значення метрики ЕР, яка зменшилась приблизно в 5 разів.

В процесі створення програмного продукту виникли такі труднощі:

- 1) обмеженість в часі;
- 2) складність розробки ПП;
- 3) недостатні знання де-яких інструментів.

Не зважаючи на це всі прецеденти, які були заплановані було реалізовано.