

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
Національний університет «Запорізька політехніка»

Інститут інформатики та радіоелектроніки  
Факультет комп'ютерних наук та технологій  
(повне найменування інституту, факультету)

Кафедра комп'ютерних систем та мереж  
(повне найменування кафедри )

## Пояснювальна записка

до дипломного проекту (роботи)

бакалавра

(ступінь вищої освіти (освітній ступінь))

на тему РОЗРОБКА ІНТЕРНЕТ-ДОДАТКУ ДЛЯ ВЕДЕННЯ ОБЛІКУ ЗАМОВЛЕНЬ  
У ВЕБ-СТУДІЇ

Виконав: студент 4 курсу, групи КНТ-517  
спеціальності \_\_\_\_\_

123 «Комп'ютерна інженерія»

(код і найменування спеціальності)

Освітня програма (спеціалізація) \_\_\_\_\_

«Комп'ютерна інженерія»

Тютюник Андрій Тимурович

(прізвище та ініціали)

Керівник Касьян К.М.

(прізвище та ініціали)

Рецензент Фурманова Н.І.

(прізвище та ініціали)

м. Запоріжжя  
2021 рік

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
**Національний університет «Запорізька політехніка»**  
( повне найменування вищого навчального закладу )

Інститут, факультет інформатики та радіоелектроніки, комп'ютерних наук і технологій  
 Кафедра «Комп'ютерні системи та мережі»  
 Ступінь вищої освіти (освітній ступінь) бакалаврський  
 Спеціальність 123 Комп'ютерна інженерія  
(код і найменування)  
 Освітня програма (спеціалізація) Комп'ютерна інженерія  
(назва освітньої програми (спеціалізації))

**ЗАТВЕРДЖУЮ**

Завідувач кафедри Кудерметов Р.К.

“ ” 2020 року

**З А В Д А Н Н Я**  
**НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТА**

Тютюника Андрія Тимуровича  
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Розробка інтернет-додатку для ведення обліку замовлень у веб-студії

керівник проекту (роботи) Касьян Костянтин Миколайович к.т.н, доцент.  
( прізвище, ім'я, по батькові, науковий ступінь, вчене звання )

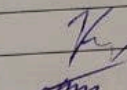
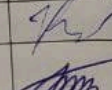
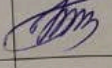
затверджені наказом закладу вищої освіти від « 17 » Березня 2021 року № 81

2. Строк подання студентом проекту (роботи) \_\_\_\_\_
3. Вихідні дані до проекту (роботи) Авторизація в систему, створення та редагування проектів, задач і користувачів. Систематизація виконання задач. Ведення обліку часу затраченого на виконання задач. Система працює на зовнішньому сервері або на локальному сервері платформи ХАМРР.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Аналітичний огляд. Проектування системи. Тестування розробленої системи.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

- Структурна схема бази даних
- Алгоритм роботи системи
- Алгоритм роботи в системі
- Вигляд основних сторінок системи

## 6. Консультанти розділів проекту (роботи)

| Розділ        | Прізвище, ініціали та посада консультанта | Підпис, дата                                                                        |                                                                                     |
|---------------|-------------------------------------------|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|
|               |                                           | завдання видав                                                                      | прийняв виконання завдань                                                           |
| 1-3           | Касьян К.М. к.т.н. доцент                 |  |  |
| нормоконтроль | Щербак Н.В. ст. викл.                     |  |  |
|               |                                           |                                                                                     |                                                                                     |
|               |                                           |                                                                                     |                                                                                     |
|               |                                           |                                                                                     |                                                                                     |
|               |                                           |                                                                                     |                                                                                     |
|               |                                           |                                                                                     |                                                                                     |

7. Дата видачі завдання « 01 » березня 2021 року.

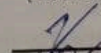
## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів дипломного проекту (роботи) | Строк виконання етапів проекту (роботи) | Прим. |
|-------|------------------------------------------|-----------------------------------------|-------|
| 1     | Складення плану                          | 22.02.2021 р.                           |       |
| 2     | Вибір технологій                         | 01.03.2021 р.                           |       |
| 3     | Проектування бази даних                  | 07.03.2021 р.                           |       |
| 4     | Проектування алгоритму роботи            | 15.03.2021 р.                           |       |
| 5     | Розробка фронт-енду                      | 22.03.2021 р.                           |       |
| 6     | Розробка бек-енду                        | 03.04.2021 р.                           |       |
| 7     | Тестування                               | 17.04.2021 р.                           |       |
|       |                                          |                                         |       |
|       |                                          |                                         |       |
|       |                                          |                                         |       |
|       |                                          |                                         |       |
|       |                                          |                                         |       |
|       |                                          |                                         |       |
|       |                                          |                                         |       |
|       |                                          |                                         |       |

Студент(ка)

Керівник проекту (роботи)

  
 (підпис)

  
 (підпис)
Тютюник А.Т.  
(прізвище та ініціали)К.М. Касьян  
(прізвище та ініціали)

## РЕФЕРАТ

ПЗ: 62 с., 33 рис., 20 джерел.

АРХІТЕКТУРА MVC, БАЗА ДАНИХ, МЕТОДОЛОГІЯ SCRUM, ВЕБ-РОЗРОБКА, ПРОЕКТИ ТА ЗАДАЧІ, СИСТЕМА УПРАВЛІННЯ.

Об'єкт розробки – веб-додаток який використовується у локальній мережі компанії ТОВ “Веб-студія”.

Метою роботи є створення веб-додатку для контролю виконання задач та ведення обліку проектів та задач до них, з функцією ведення обліку користувачів.

Проект складається з 3 розділів.

Перший розділ присвячено аналізу предметної області та технічного завдання, а точніше: дослідження проблеми, її актуальність, які аналізи вже існують і чим вони відрізняються, як розв'язуються подібні проблеми, які технології використовуються.

У другому розділі розглядається проектування структури системи, проектування алгоритмів роботи системи, проектування інтерфейсу, бази даних та програмна реалізація.

Третій розділ присвячений тестуванню системи, дослідження характеристик та демонстрація досягнутої ефективності.

В результаті виконання дипломної роботи проведено аналіз, моделювання та дослідження розроблюваного веб-додатку та розроблені рекомендації щодо її модернізації.

## ЗМІСТ

|                                                            |    |
|------------------------------------------------------------|----|
| Вступ.....                                                 | 6  |
| 1 Аналітичний огляд.....                                   | 7  |
| 1.1 Призначення системи.....                               | 7  |
| 1.2 Аналоги.....                                           | 8  |
| 1.3 Використовувані технології.....                        | 13 |
| 1.4 Технології бек-енд.....                                | 14 |
| 1.5 Технології фронт-енд.....                              | 15 |
| 1.6 Обґрунтування вибору технологій.....                   | 16 |
| 2 Проектування системи.....                                | 22 |
| 2.1 Перший етап проектування, опис бази даних.....         | 22 |
| 2.2 Другий етап проектування, алгоритм роботи системи..... | 25 |
| 2.3 Програмна реалізація функцій системи.....              | 29 |
| 2.4 Проекти та задачі.....                                 | 32 |
| 2.5 Користувачі.....                                       | 44 |
| 3 Тестування розробленої системи.....                      | 51 |
| 3.1 Тестування авторизації.....                            | 51 |
| 3.2 Тестування функцій взаємодії з користувачами.....      | 54 |
| 3.3 Тестування функцій взаємодії з проектами.....          | 56 |
| 3.4 Тестування функцій взаємодії з задачами.....           | 57 |
| Висновки.....                                              | 64 |
| Перелік джерел посилання.....                              | 65 |

## ВТСУП

Від правильного розподілення співробітників між існуючими замовленнями фірми залежить вибір правильних рішень в управлінні фірмою. Необхідна оптимізація цих операцій шляхом розробки інтернет-додатку, який забезпечить введення, зберігання, редагування та отримання необхідної інформації, а також спростить контролювання витрачаємого робочого часу. На сьогоднішній день в сучасних умовах розвитку інформаційних технологій та участі їх в повсякденному житті людей складно знайти людину, у якої б не було в тій чи іншій мірі необхідності використання електронної техніки, як в побутових умовах, так і для виконання ділових і робочих завдань. Впровадження нових або невідомих в даному конкретному колективі інформаційних технологій найчастіше ознаменовано необхідністю до навчання використання їх в робочому процесі. Виходячи з вищевказаного документа, виявлена необхідність в створенні системи управління замовленнями, яка допоможе оптимізувати робочий процес та надасть можливість стежити за часом який витрачається на рішення задач. Таким чином, тема дипломної роботи є актуальною. Метою дипломної роботи є створення інтернет-додатку для оптимізації робочого процесу та спрощення ведення обліку замовлень, що дозволить забезпечити легкість і оптимальну ефективність роботи при виконанні завдань, а також надасть керівникам можливість вести облік часу, який співробітники витрачають на рішення задач, для їх подальшого розподілення між замовленнями та задачами що спричинить ще більшу оптимізацію робочого часу.

Задачами дипломної роботи є:

- Аналіз проблеми і огляд існуючих рішень;
- Проектування алгоритмів роботи системи;
- Розробка бази даних, інтерфейсу та функціоналу системи;
- Тестування розробленої системи для виявлення помилок і їх усунення

# 1 АНАЛІТИЧНИЙ ОГЛЯД

## 1.1 Призначення системи

Веб-додаток був розроблений для ведення обліку та контролю виконання задач до проектів. Для початку, треба зазначити, що ця система розроблялася суто для використання в межах локальної мережі тієї чи іншої компанії. Це означає, що уся система знаходиться на сервері безпосередньо у приміщенні цієї компанії, тобто використовувати цю систему можливо лише знаходячись на робочому місці. Це обмеження зумовлено постановленим завданням. А завдання було таке, створити таку систему в якій одночасно могли працювати будь яка кількість працівників, при чому кожен міг працювати над одним і тим же проектом, тобто працівники можуть виконувати різні завдання для одного проекту одночасно, що дозволяє задіяти будь яку кількість працівників для прискорення виконання проекту. Тому ця система розрахована на використання методу управління проектами SCRUM [1]. SCRUM це гнбка методологія розробки із нестандартним розподілом ролей у команді та унікальною організацією ітерацій. SCRUM - мінімально необхідний набір заходів, ролей, на яких будується процес SCRUM-розробки, що дозволяє за фіксовані невеликі проміжки часу, звані спринтами, надавати кінцевому користувачеві працюючий продукт з новими бізнес - можливостями, для яких визначено найбільший пріоритет. Спринт це проміжок часу, достатній для виконання запланованої сукупності операцій SCRUM, метою якої є створення інкременту бізнес-продукту. Веб-додаток був розроблений з метою створення такої системи, яку можливо легко модифікувати для використання різними компаніями, які придбають цю систему, також щоб робота у такій системі нагадувала роботу за методологією SCRUM. Зараз питання ефективності є дуже актуальним, оскільки проекти які умовна компанія буде одержувати не будуть однаковими за обсягом та складністю. Проекти можна поділити за 2 ознаками, за складністю проекту та за його обсягом. За складність проекту відповідає складність окремих задач, тобто чим більше задач на які

витрачають багато часу тим складніший проект. За обсяг проекту відповідає кількість задач, тобто чим більше задач, не залежно від їх складності, тим більший обсяг проекту. З цього випливає, що найкращим рішенням цієї проблеми буде можливість підключення будь якої кількості працівників до складних чи великих за обсягом проектів, або навпаки, їх відлучення від простих чи малих проектів для більш раціонального використання.

## **1.2 Аналоги**

Перед початком проектування, було знайдено три аналога. Ці аналоги називаються Jira, Trello та Zistemo. Усі аналоги мають в собі весь функціонал веб-додатку і навіть більше. Важливо зауважити, що огляд аналогів проводився у безкоштовних версіях, тому функціонал цих систем був обмежений. Перше що можливо помітити при реєстрації у цих системах, це вибір напрямку роботи компанії яка вибрала цю систему керування проектами. Далі у Trello відкривається головна сторінка компанії, на якій можливо створювати нові проекти. Після створення проекту, відкривається вікно цього проекту, у якому можливо створювати нові картки в одній з трьох колонок, в колонці “Необхідно зробити”, “В процесі”, “Готово”. Окрім цього, в Trello є функція запрошення користувачів з інших систем, якщо встановлена така інтеграція. Користувачі що приєдналися за таким запрошенням зберігають інформацію з попередньої системи.

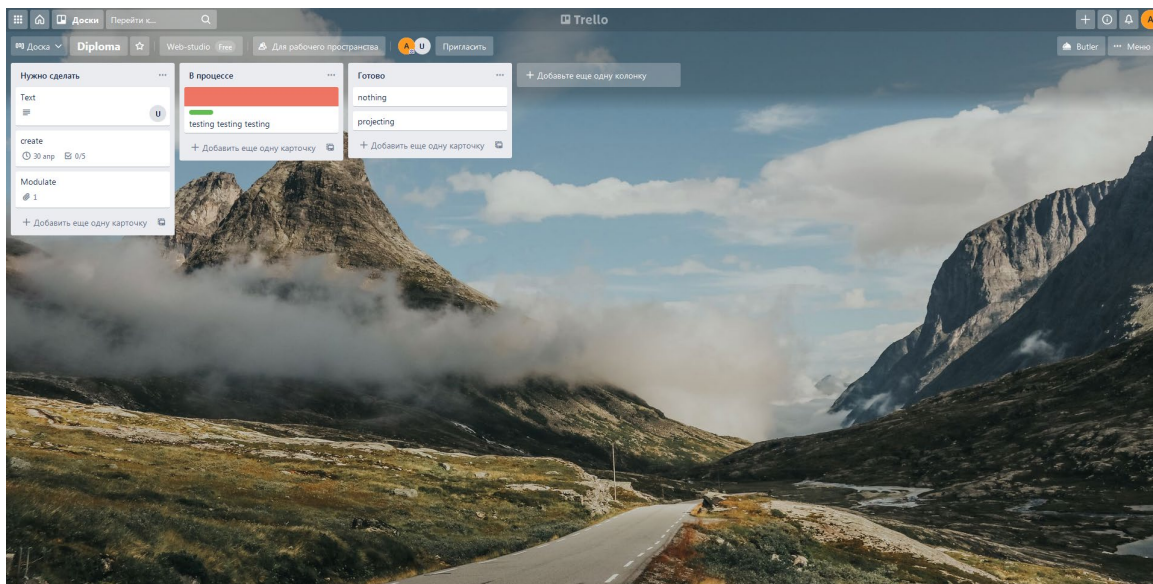


Рисунок 1.1 – Сторінка проекту в системі Trello

На рисунку 1.1 зображена головна сторінка системи Trello. Після створення карток, їх можливо переносити з однієї колонки до іншої, можливо додати опис картки. Також до кожної картки можливо додати учасника, помітку, чек-лист, термін чи вложення. Ці картки є умовними задачами до проектів. Окрім цих функцій, які були також частково реалізовані у веб-додатку, Trello має велику кількість функцій. В системі Jiga ж після реєстрації компанії та вибору напрямку роботи, є можливість вибрати за якою методологією працюватимуть учасники. При ознайомленні був вибраний варіант SCRUM. Після створення проекту, відкривається сторінка управління проектами. Перед тим як розпочати роботу, необхідно створити спринт, а в ньому вже задачі які будуть виконувати у період цього спринту. Також присутня можливість обміну кодом між співробітниками в цій системі. Та багато інших функцій.

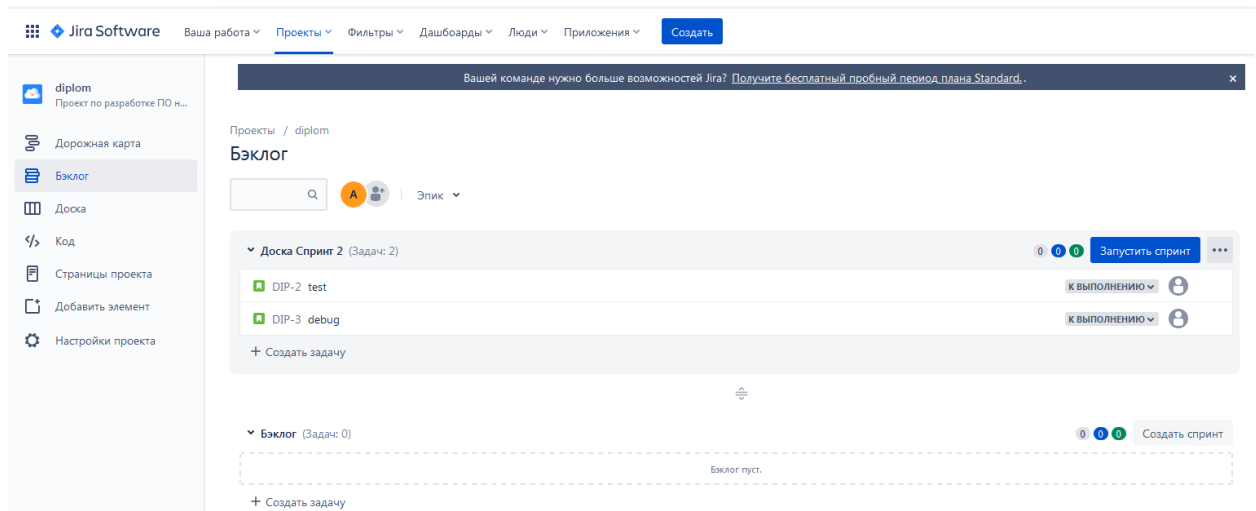


Рисунок 1.2 – Сторінка створення спринтів і задач у системі Jira

На рисунку 1.2 зображена сторінка для роботи в системі Jira. Після реєстрації у системі Zistemo, пропонується вибрати необхідний функціонал. Це може бути набір “Час перебування”, “Відстеження проектів” чи “Повна система”. Під час огляду, був вибраний набір “Повна система”. Після вибору набору, користувачу пропонують підключити інтеграції з іншими системами, і з тими які були згадані раніше теж, для обміну даними при необхідності перенести проекти та інше. Далі користувач потрапляє на сторінку управління, де є майже все. Можливість створити нового клієнта, кошторис, проект, рахунок чи витрату. Можливість відстеження всіляких даних по компанії, наприклад: відправлені рахунки чи кошториси, активні проекти, запити, облік відсутності робітників чи робітників у відпустці, та багато іншого. Також є журнал останніх дій.

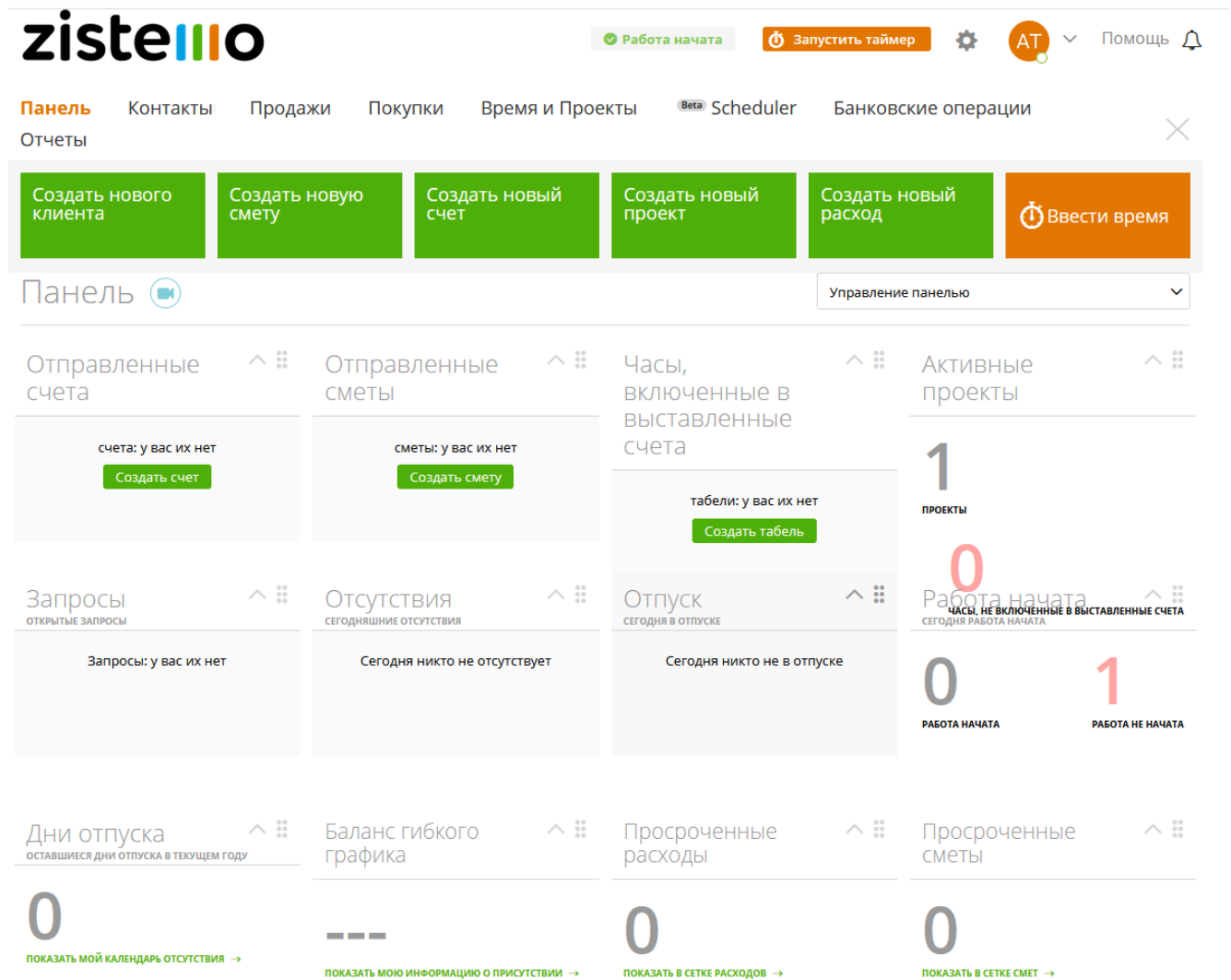


Рисунок 1.3 Головна сторінка Zistemo перша частина

На рисунку 1.3 зображена головна сторінка системи Zistemo яка містить головні функції. Якщо опуститися нижче, можна знайти велику кількість графіків та діаграм які інформують про рахунки та платежі, витрачений час, доходи і витрати та інше.

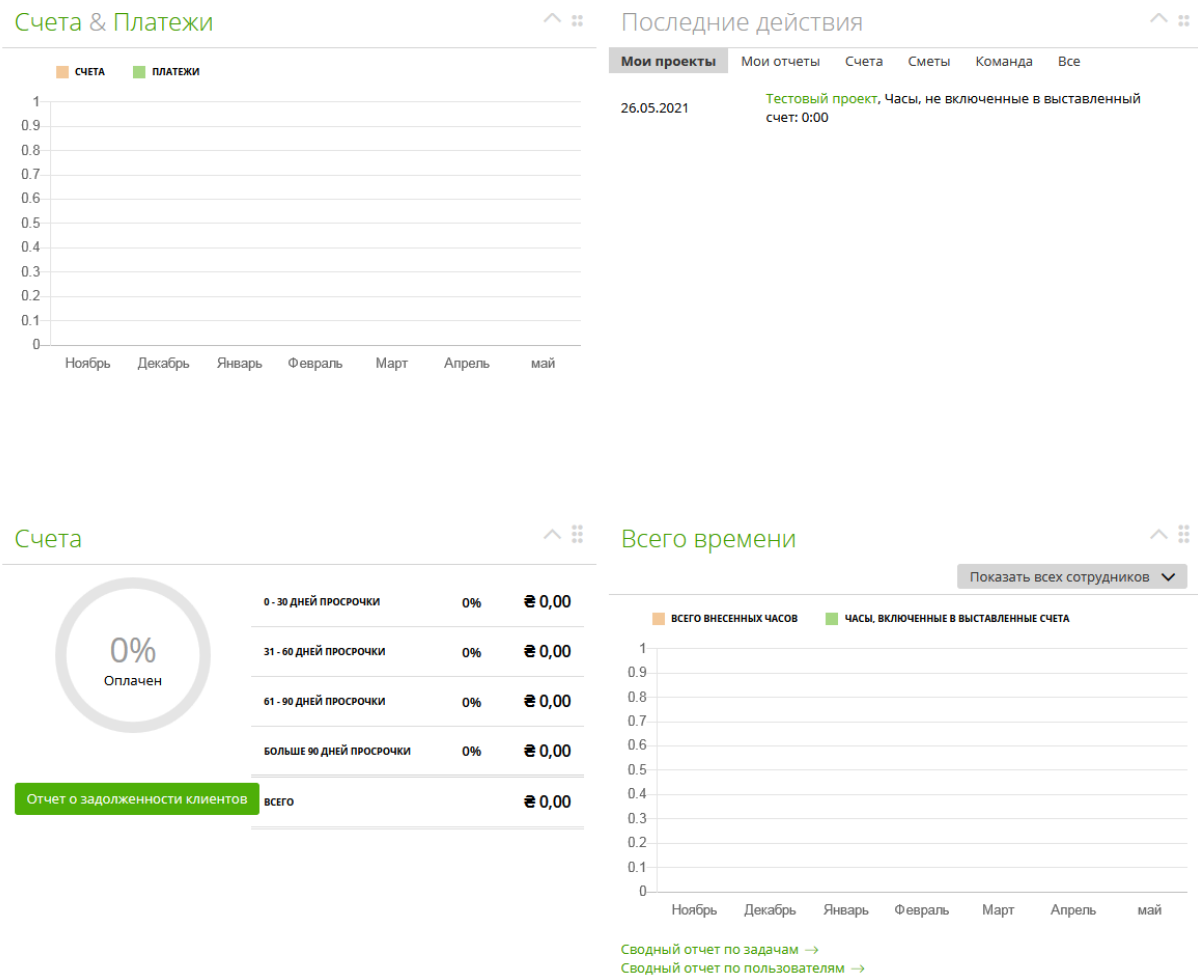


Рисунок 1.4 Головна сторінка Zistemo друга частина

На рисунку 1.4 зображена головна сторінка системи Zistemo яка містить графіки та діаграми з даними компанії. Підводячи підсумки, можливо зробити такі висновки. Існуючі системи контролю та управління проектами мають вражаючий функціонал. Деякі зі згаданих систем можуть майже повністю замінити програмне забезпечення для ведення обліку у компанії. Так наприклад в Zistemo можливі створювати доповіді чи графіки з діаграмами що стосуються бухгалтерів та дозволяють контролювати грошову складову компанії. Або ведення обліку співробітників не тільки як перелік, а й навіть рахувати витрачений час кожним з них, час який вони були відсутні та контроль відпусток. Усі переглянуті системи поділяють один головний недолік. Цей недолік, це нагромадженість функцій. Після реєстрації користувач потрапляє на головні

сторінки систем і перше що він баче це неймовірна кількість функцій та інформації. У цьому плані Trello здається кращим вибором, оскільки нема такого критичного перенавантаження інформацією як в інших аналогах. Однак і в Trello є свої недоліки. Один з головних, це первинне нерозуміння принципу робочого столу. Після реєстрації, на головній сторінці, було лише декілька кнопок навігації і головні в цій системі картки, які на перший погляд не надають ніякого розуміння для чого вони.

### 1.3 Використовувані технології

Ця система спроектована за парадигмою (пакет програм) LAMP [2] що розшифровується як:

- L – Linux, операційна система;
- A – Apache, веб-сервер;
- M – MySQL / MariaDB, система керування базами даних;
- P – PHP / Perl / Python, мова програмування для створення веб-додатків.

Цей комплекс програмного забезпечення має в собі усе необхідне для створення повністю функціонуючих веб-додатків. Звідси й впливає популярність цієї парадигми. В інтернеті існує багато схожих комплексів, які виконують таку саму роль, але лише LAMP, WAMP(windows замість Linux) та XAMPP [3] отримали значну популярність. Саме на XAMPP розроблена ця система. XAMPP це:

- X – Будь яка з 4 ОС;
- A – Apache;
- M – MySQL;
- P – PHP;
- P – Perl;

## 1.4 Технології бек-енд

Linux [4] - загальна назва UNIX-подібних операційних систем на основі однойменного ядра. Це один із найвидатніших прикладів розробки вільного (free) та відкритого (з відкритим кодом, open source) програмного забезпечення (software). На відміну від власницьких операційних систем (на кшталт Microsoft Windows та MacOS X), їхні вихідні коди доступні всім для використання, зміни та поширення абсолютно вільно (в тому числі безкоштовно).

Apache [5] HTTP-сервер - відкритий веб-сервер Інтернет для UNIX-подібних, Microsoft Windows, Novell NetWare та інших операційних систем.

Apache розроблюється та підтримується спільнотою розробників відкритого програмного забезпечення під керівництвом Apache Software Foundation.

Web-сервер Apache є самостійним, некомерційним, вільно розповсюджуваним продуктом. Продукт підтримує безліч можливостей, багато з яких реалізовані як скомпільовані модулі, які розширюють основні функціональні можливості. Вони різняться від серверної підтримки мов програмування до схем аутентифікації. Існують інтерфейси для підтримки мов програмування Perl, Python, Tcl і PHP.

MySQL [6] - вільна система керування реляційними базами даних. MySQL є рішенням для малих і середніх додатків. Входить до складу серверів WAMP, AppServ, LAMP і в портативні збірки серверів Денвер, XAMPP, VertrigoServ. Зазвичай MySQL використовується як сервер, до якого звертаються локальні або видалені клієнти, проте в дистрибутив входить бібліотека внутрішнього сервера, що дозволяє включати MySQL в автономні програми.

PHP [7] - скриптова мова програмування, була створена для генерації HTML-сторінок на стороні веб-сервера. PHP є однією з найпоширеніших мов, що використовуються у сфері веб-розробок (разом із Java, .NET, Perl, Python, Ruby). PHP підтримується переважною більшістю хостинг-провайдерів.

Smarty [8]— це компілюючий обробник шаблонів до PHP. Надає інструменти, які дозволяють домогтися відділення прикладної логіки й даних від відображення. Це дуже зручно в ситуаціях, коли програміст і верстальник шаблону працюють окремо.

## **1.5 Технології фронт-енд**

HTML [9] – стандартизована мова розмітки веб-сторінок у Всесвітній павутині. Код HTML інтерпретується браузером; отримана в результаті інтерпретації сторінка відображається на екрані монітора комп'ютера або мобільного пристрою. Елементи HTML є будівельними блоками сторінок HTML. За допомогою конструкцій HTML, зображення та інші об'єкти, такі як інтерактивні форми, можуть бути вбудовані у візуалізовану сторінку. HTML надає засоби для створення структурованих документів, позначаючи структурну семантику тексту, наприклад заголовки, абзаци, списки, посилання, цитати та інші елементи. Елементи HTML окреслені тегами, написаними з використанням кутових дужок.

CSS [10] - це спеціальна мова стилю сторінок, що використовується для опису їхнього зовнішнього вигляду. Самі ж сторінки написані мовами розмітки даних. CSS є основною технологією всесвітньої павутини, поряд із HTML та JavaScript. Найчастіше CSS використовують для візуальної презентації сторінок, написаних HTML, але формат CSS може застосовуватися до інших видів XML-документів.

JavaScript [11] (JS) — динамічна, об'єктно-орієнтована прототипна мова програмування. Реалізація стандарту ECMAScript. Найчастіше використовується для створення сценаріїв вебсторінок, що надає можливість на боці клієнта (пристрої кінцевого користувача) взаємодіяти з користувачем, керувати

браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд вебсторінки.

jQuery [12] — популярна JavaScript-бібліотека з відкритим кодом(фреймворк). Згідно з дослідженнями організації W3Techs, JQuery використовується понад половиною від мільйона найвідвідуваніших сайтів. jQuery є найпопулярнішою бібліотекою JavaScript, яка посилено використовується на сьогоднішній день.

Bootstrap [13] - вільний набір інструментів для створення сайтів і веб-додатків. Включає в себе HTML- і CSS-шаблони оформлення для типографіки, веб-форм, кнопок, міток, блоків навігації та інших компонентів веб-інтерфейсу, включаючи JavaScript-розширення.

## **1.6 Обґрунтування вибору технологій**

Головні критерії вибору технологій були безкоштовність, доступність та кількість інформації, для цих технологій, яку можливо знайти в інтернеті у вільному доступі. Далі, технології були підібрані для розробки системи за архітектурою MVC. MVC[14] або model – view – controller, архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення. Цей шаблон передбачає поділ системи на три взаємопов'язані частини: модель даних, вигляд (інтерфейс користувача) та модуль керування. Застосовується для відокремлення даних (моделі) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача. Мета шаблону гнучкий дизайн програмного забезпечення, який повинен полегшувати подальші зміни чи розширення програм, а також надавати можливість повторного використання окремих компонентів програми. Крім того використання цього шаблону у великих системах сприяє впорядкованості їхньої

структури і робить їх більш зрозумілими за рахунок зменшення складності. У рамках архітектурного шаблону модель–вигляд–контролер (MVC) програма поділяється на три окремі, але взаємопов'язані частини з розподілом функцій між компонентами. Модель (Model) відповідає за зберігання даних та їх структуру. Вигляд (View) відповідальний за представлення цих даних користувачеві, тобто інтерфейс програми. Контролер (Controller) керує компонентами, отримує сигнали у вигляді реакції на дії користувача (зміна положення курсора миші, натискання кнопки, ввід даних в текстове поле) і передає дані у модель.

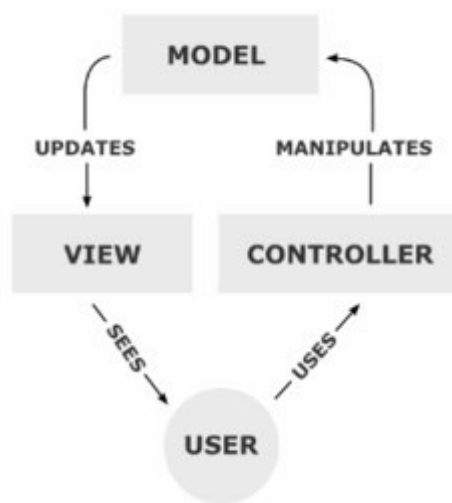


Рисунок 1.5 – Графічне зображення зв'язків в архітектурі MVC

На рисунку 1.5 зображено як саме повинна працювати система за архітектурою MVC.

ХАМРР, це платформа для локального хостингу серверів, яка включає в себе усі необхідні сервери, а саме: Apache MySQL PHP. ХАМРР це безкоштовне програмне забезпечення з яким навіть недосвідчений користувач зможе розібратися. Достатньо завантажити це ПЗ, встановити його та запустити. Далі достатньо натиснути кнопку старт навпроти Apache і через секунду чи дві сервер буде працювати на локальному комп'ютері. Щоб отримати доступ до будь якого сервера достатньо в браузері у поле для вводу URL ввести localhost і відкриється локальна сторінка з якої можливо керувати усією системою.

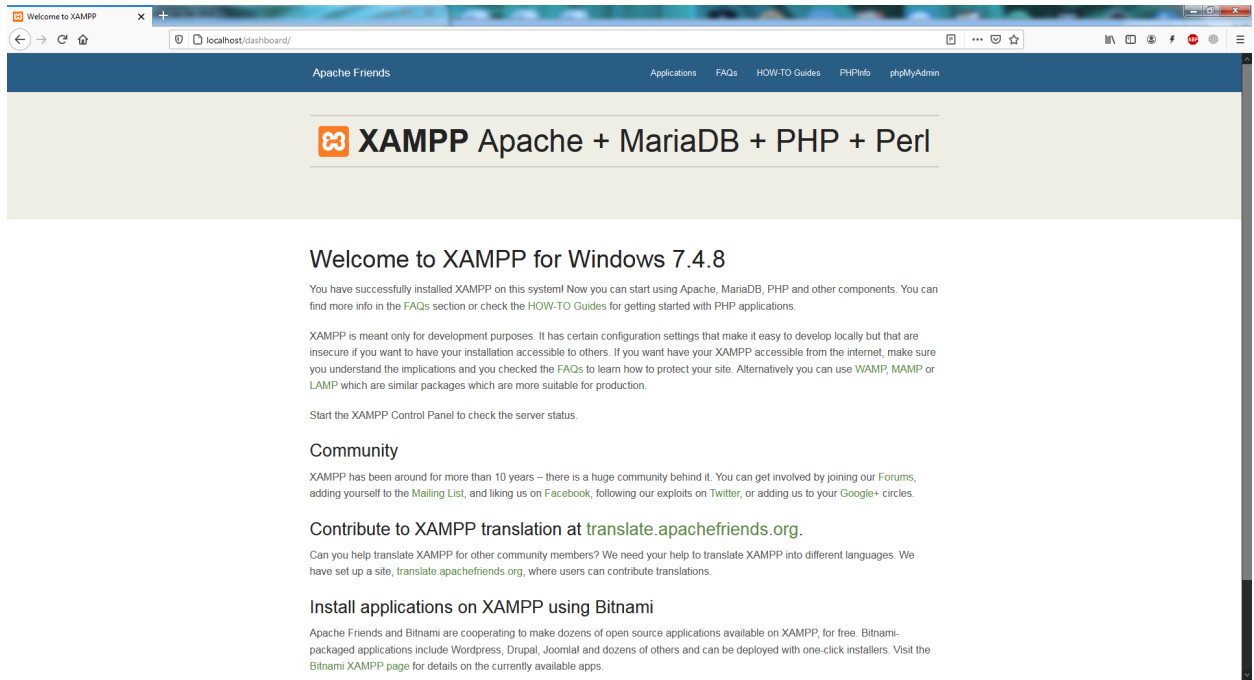


Рисунок 1.6 – Вигляд локальної сторінки localhost

На рисунку 1.6 зображена сторінка localhost, коли увімкнений сервер Apache.

Якщо увімкнути ще й MySQL сервер, натиснувши на кнопку phpMyAdmin, потрапляємо до СКБД MariaDB. Ця СКБД є інтерфейсом для SQL запитів. За допомогою цього веб-інтерфесу користувач може легко виконувати запити без знання мови SQL. Інтерфейс самостійно будує запити після збереження змін або при створенні нових таблиць.

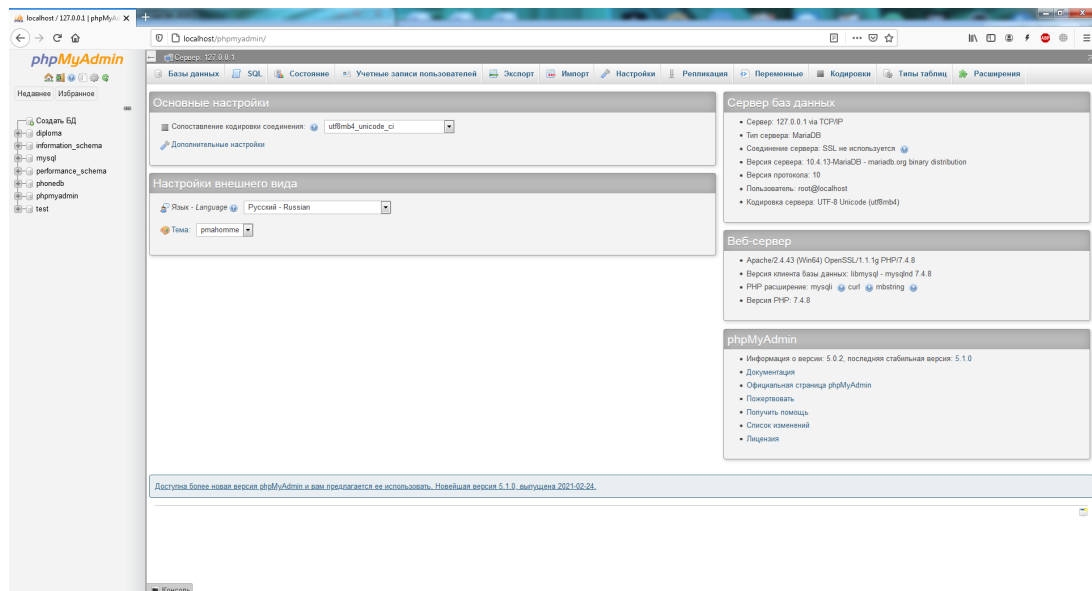


Рисунок 1.7 – Вигляд головної сторінки phpMyAdmin

На рисунку 1.7 зображена головна стрінка, на яку потрапляє користувач одразу після натискання кнопки phpMyAdmin. Важливо зазначити, що phpMyAdmin це веб-інтерфейс для керування MySQL сервером. Він дозволяє викотувати SQL запити натискаючи кнопки а ніж пишучи ці запити самостійно. Так наприклад, для створення нової таблиці та її заповнення достатньо обрати існуючу, чи створити нову, базу даних, у полі кількість столбців ввести потрібне значення та натиснути кнопку вперед. Далі заповнити параметри столбців і нова таблиця готова.

Зараз, мова запитів SQL, є лідируючою серед собі подібних. Це зумовлено тим, що завдяки SQL запитам можливо домогтися майже будь-яких опрецій з базами даних, а саме: створення нових таблиць, редагування кількості столбців в таблиці, налаштування параметрів столбців, вибірка, видалення столбців, оновлення даних та багато інших.

“Шаблонизатор” SMARTY дозволяє створювати файли-шаблони, які є модифікованими файлами HTML. У такі файли пишуть звичайний HTML код з використанням SMARTY тегів, які в свою чергу дозволяють використовувати всередині себе PHP змінні та де-які функції такі як, “for-each” та “if”.

В наш час PHP є однією з лідируючих мов для створення веб-додатків. А оскільки ця система розроблена на платформі XAMPP вибір пав саме на PHP. PHP не дуже складна мова, яка чітко вписується у архітектуру MVC, та справляється з усіма задачами які потребує ця архітектура для справної та налагодженої роботи усієї системи. Також, для взаємодії PHP з базами даних був використаний драйвер PDO.[15] Розширення Об'єкти даних PHP (PDO) визначає простий і узгоджений інтерфейс для доступу до баз даних в PHP. Кожен драйвер бази даних, в якому реалізований цей інтерфейс, може уявити специфічну для бази даних функціональність у вигляді стандартних функцій розширення. Але треба зауважити, що саме по собі розширення PDO не дозволяє маніпулювати доступом до бази даних. Щоб скористатися можливостями PDO, необхідно використовувати відповідний до конкретної бази даних PDO драйвер. PDO забезпечує абстракцію доступу до даних, це означає, що незалежно від того, яка конкретна база даних використовується, ви можете користуватися одними і тими ж функціями для виконання запитів і вибірки даних. PDO НЕ абстрагує саму базу даних, це розширення не переписує SQL-запити і не емулює відсутній в СКБД функціонал.

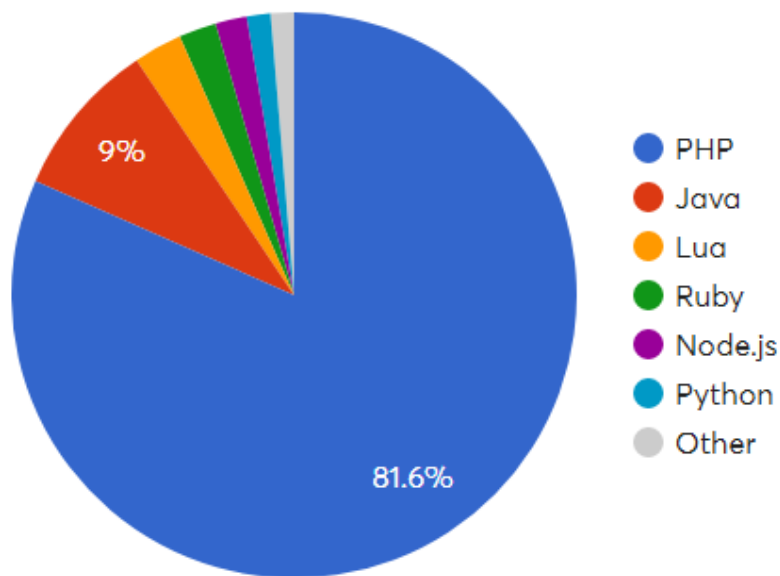


Рисунок 1.8 Популярність мов програмування для веб-розробок [16]

На рисунку 1.8 зображена діаграма в якій представлені мови програмування за популярністю серед веб-розробників. На цій діаграмі повністю лідирує PHP, оскільки майже 80% сайтів написані саме на PHP. Дані були взяті за січень 2019 року.

HTML та CSS є основою будь якого сайту. Можливо створювати невеликі сайти використовуючи лише ці дві мови. HTML є “скелетом” веб-додатку, завдяки HTML коду можна побачити структуру сторінки, а CSS форматує вигляд цієї сторінки так, щоб вона виглядала як завершена структура. Також CSS дуже гнучка мова стилів, що дозволяє для одого класу HTML об'єкта чи декількох груп задавати конкретні значення.

У системі, JavaScript(JS) використовується для функцій, які не можуть бути реалізованими іншою мовою, або можуть але потребують більше часових ресурсів, як при написанні цієї функції так і при її обробці на стороні клієнт-сервері. Наприклад оновлення виду сторінки без її перезавантаження.

JQuery доповнює можливості JS. Він спрощує синтаксис, додає нові функції та й в цілому спрощує розробку JS функцій.

Bootstrap корисний набір інструментів при створенні сайтів чи веб-додатків з невизначеним дизайном. Оскільки в Bootstrap є заздалегідь створені класи, із дизайном властивим саме Bootstrap, це дозволяє створювати системи з простим та зрозумілим дизайном який до того ж виглядає досить приємно. Зараз дизайн на основі Bootstrap`у є одним з стандартів веб дизайну, оскільки він був створений для веб-ресурсу Facebook. Переглянувши де-які сайти можливо знайти такі, які використовують саме Bootstrap дизайн.

## 2 ПРОЕКТУВАННЯ СИСТЕМИ

### 2.1 Перший етап проектування. Опис бази даних

Перший етап проектування системи, це визначення необхідних цілей. Головною метою було створення системи ведення обліку проектів і виконання задач у команді розробників з різним набором навичок, а також можливість коментування задач для більш детального опису, можливість ведення обліку часу витраченого на виконання завдання, можливість розподілу користувачів за ролями для обмеження доступу і можливість призначати користувачів на виконання задач відповідно їх навичкам. Наступний етап, це проектування бази даних цієї системи. Відповідно до постановлених цілей, була визначена наступна структура бази даних:

а) “users” – в цій таблиці зберігаються дані користувача. В таблиці є поля:

- 1) “id” – номер користувача. Це поле є первинним ключем;
- 2) “login” – унікальний ідентифікатор користувача в системі. Це поле є унікальним ключем оскільки в системі не можуть існувати два однакових ідентифікатора;
- 3) “password” – хеш паролю користувача, який необхідний для авторизації;
- 4) “email” – електронна пошта користувача;
- 5) “name” – ім`я користувача;
- 6) “role” – роль користувача. Можливі ролі – адміністратор, менеджер, користувач;
- 7) “user\_pic” – аватар користувача;
- 8) “active” – це поле відповідає за активність користувача, якщо значення 1 то користувач може увійти в систему, якщо 0, то не зможе увійти;

б) “projects” – в цій таблиці зберігається існуючі проекти;

- 1) “id” – Це поле є первинним ключем;
- 2) “name” – назва проекту;
- 3) “creation\_date” – дата створення проекту визначається при створенні проекту;
- 4) “status” – поточний стан проекту;
- 5) “description” – опис проекту;
- 6) “user\_id” – номер користувача відповідального за цей проект;

в) “sessions” – в цій таблиці зберігається хеш токени сесій користувачів, які необхідні для авторизації в системі. Поки запис існує, відповідний користувач може знаходитись в системі. Коли він виходить з системи, запис видаляється;

- 1) “user\_id” – номер користувача;
- 2) “user\_token” – хеш токен користувача;
- 3) “remember\_me” – чи був встановлений прапорець “запам’ятай мене” при останній авторизації;
- 4) “creation\_time” – час авторизації;

г) “skills” – в цій таблиці перераховані які навички присутні якому користувачу;

- 1) “user\_id” – номер користувача;
  - 2) “skill” – назва наички;
- д) “tasks” – в цій таблиці зберігаються задачі до проектів;
- 1) “id” – Це поле є первинним ключем;
  - 2) “name” – назва задачі;
  - 3) “description” – опис задачі;
  - 4) “project\_id” – номер проекту до якої відноситься ця задача;
  - 5) “user\_id” – номер користувача який виконує цю задачу;
  - 6) “creation\_date” – дата створення задачі;

7) “deadline\_date” – дата до якої потрібно завершити цю задачу;

8) “status” – поточний стан задачі;

е) “worklog” – в цій таблиці зберігаються дані про виконання задач користувачами;

1) “id” – Це поле є первинним ключем;

2) “task\_id” – номер задачі яка виконувалася;

3) “user\_id” – номер користувача який виконував задачу;

4) “description” – опис діяльності який міг залишити виконавець;

5) “start\_time” – час розпочатку виконання роботи;

6) “end\_time” – час призупинення виконання роботи;

7) “total\_time” – витрачений час за цю сесію.

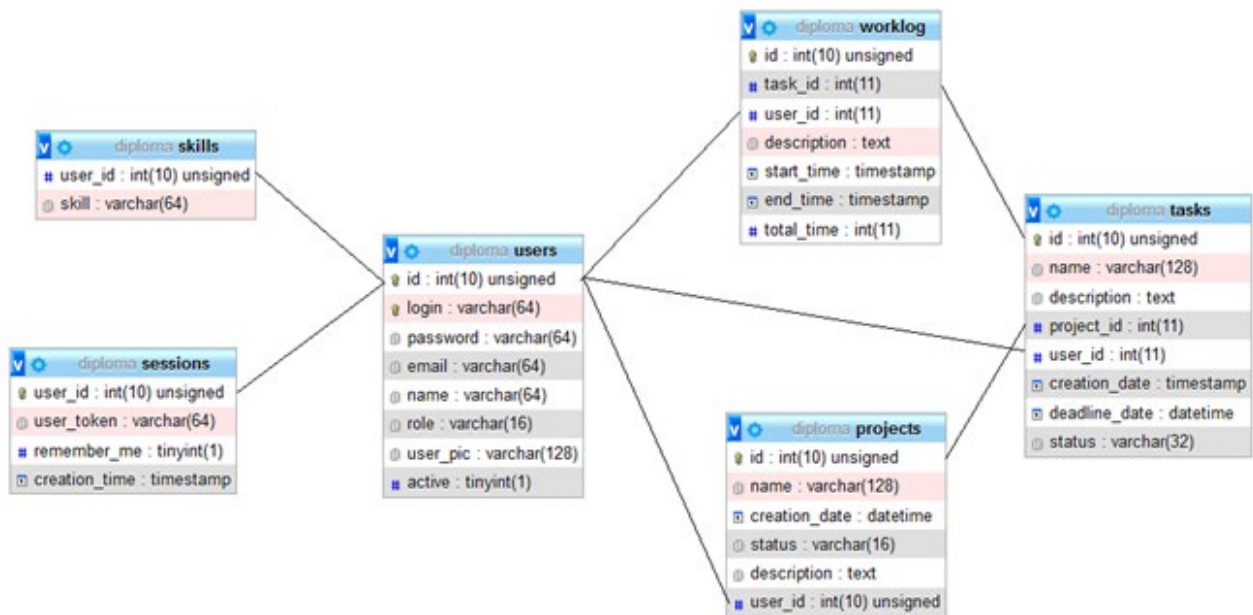


Рисунок 2.1 Структурна схема бази даних

На рисунку 2.1 знаходиться графічне зображення структурної схеми зв'язків таблиць у базі даних.

## 2.2 Другий етап проектування. Алгоритм роботи системи

Система поділяється на PHP класи, tpl файли шаблонів, CSS файли стилів та json скрипти. Усі функції поділяються на 2 типи, функції з GET запитам та функції з POST запитам. Якщо сервер отримує POST запит, він перенаправляє потік даних у файл actions.php де головний обробник функцій це конструкція switch. Якщо сервер отримає GET запит, він перенаправляє потік даних у файл index.php.

В системі реалізован поділ користувачів за ролями. Роль користувача передбачає доступ до функцій системи, які відповідають за виконання роботи. Роль менеджера надає доступ до тих самих функцій що й користувач, а також менеджмент проектами, задачами і призначення користувачів на проекти та задачі. Роль менеджера надає доступ до функцій менеджера, а також керування користувачами, тобто редагування, видалення, додавання користувачів та їх деактивація. Ця функція грає важливу роль в обмеженні доступу у системі, тому доступ до функцій в ролях був затверджений і не може бути змінений без модифікування коду. Так само, якщо потрібно створити нові ролі необхідно модифікувати код. Заздалегідь було вирішено, якщо система буде продаватися компаніям, у цих компаній буде можливість допрацювати та модифікувати систему по бажанню. Але важливо зауважити що навіть без допрацювання або модифікування коду, система вже є повноцінною та завершеною, вона має в собі усі необхідні функції для оптимальної роботи компанії.

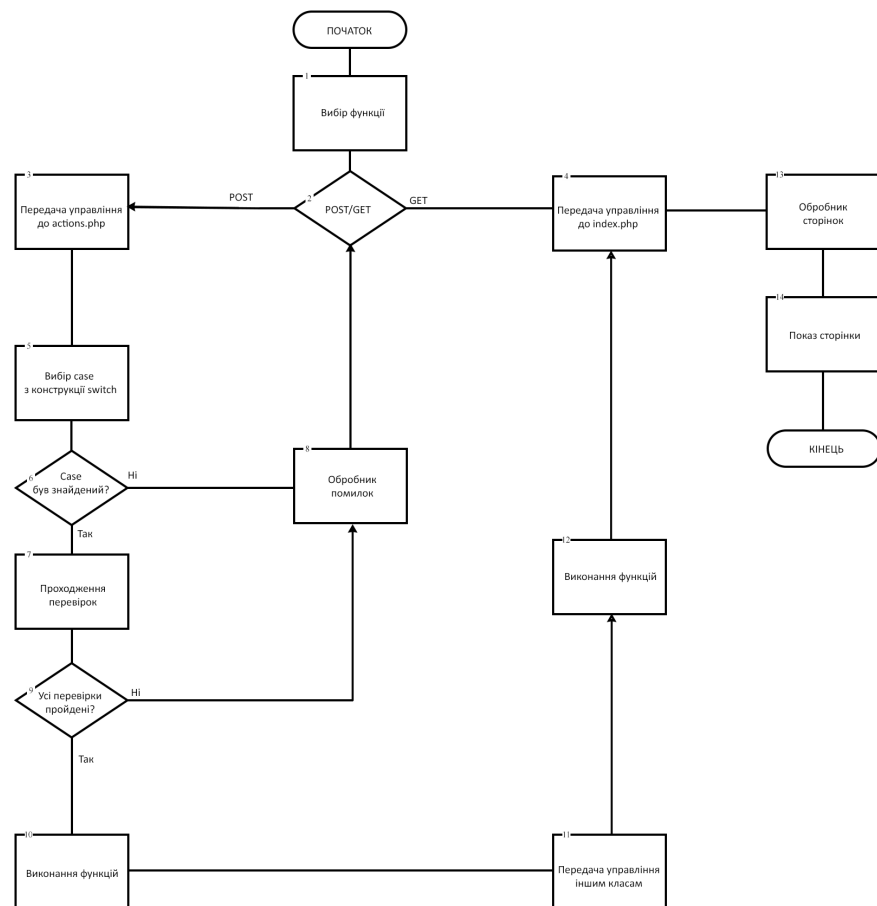


Рисунок 2.2 Алгоритм роботи системи

На рисунку 2.2 зображен головний алгоритм за яким працює система. Коли користувач натискає будь яку кнопку, система розпочинає роботу за цим алгоритмом. В першу чергу перевіряється який тип запиту у викликаній функції. Якщо це звичайний GET запит, то управління передається до класу index.php де обробник сторінок отримує передані користувачем дані і відображає вибрану сторінку. Якщо функція зроблена на базі POST запиту, то управління передається класу actions.php. Далі контролер actions.php перебирає конструкцію switch у пошуку потрібного case. Якщо такого не знайде, то управління з даними передається до index.php який відображає головну сторінку з повідомленням про помилку чи сторінку 404. Якщо case був знайдений, дані користувача проходять перевірку. Якщо вони не проходять перевірку, то управління з даними

передається до `index.php` який відображає головну сторінку з повідомленням про помилку. Після проходження перевірки, управління передається відповідним класам в яких виконуються потрібні функції, зберігаються нові дані, якщо такі є, або зберігає зміну даних. Після чого управління знов передається до `index.php` який відображає бажану сторінку або результат виконання функції. Винятками з цього алгоритму є функції яким не потрібне перезавантаження сторінки(аіах-запити).

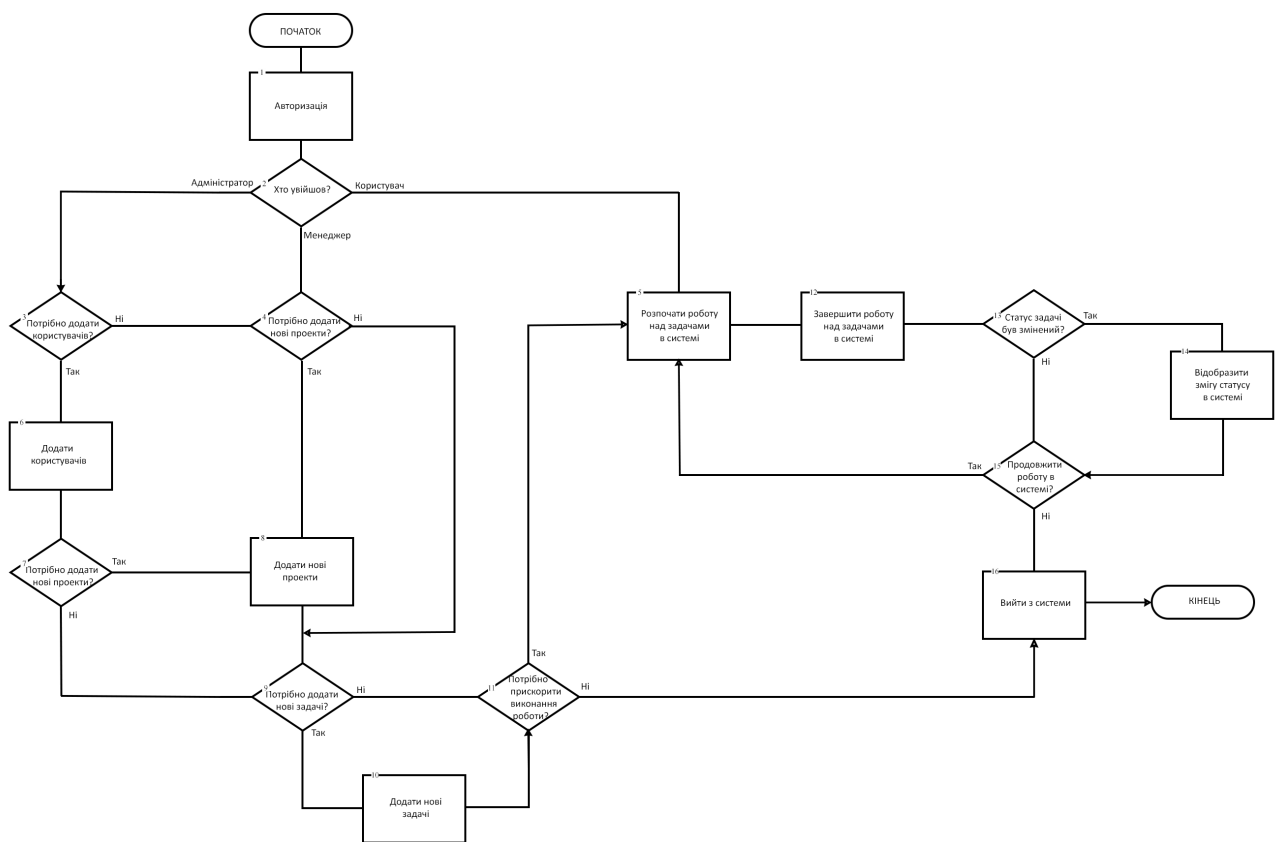


Рисунок 2.3 Алгоритм роботи в системі

На рисунку 2.3 зображений алгоритм за яким співробітники будуть працювати залежно від їх ролі в системі. Винятками з цього алгоритму є редагування даних(як редагування користувачів так і проектів та задач) та деактивація, чи активація, користувачів. Це зумовлено тим, що ці функції не є необхідними для роботи в системі, але несуть в собі важливу роль для умовної

компанії. Наприклад деактивація користувачів слугує можливістю надання відпустки співробітникам. Також редагування не є головною функцією оскільки це фактично є способом виправлення помилок, чи оновлення інформації, для користувачів, проектів та задач.

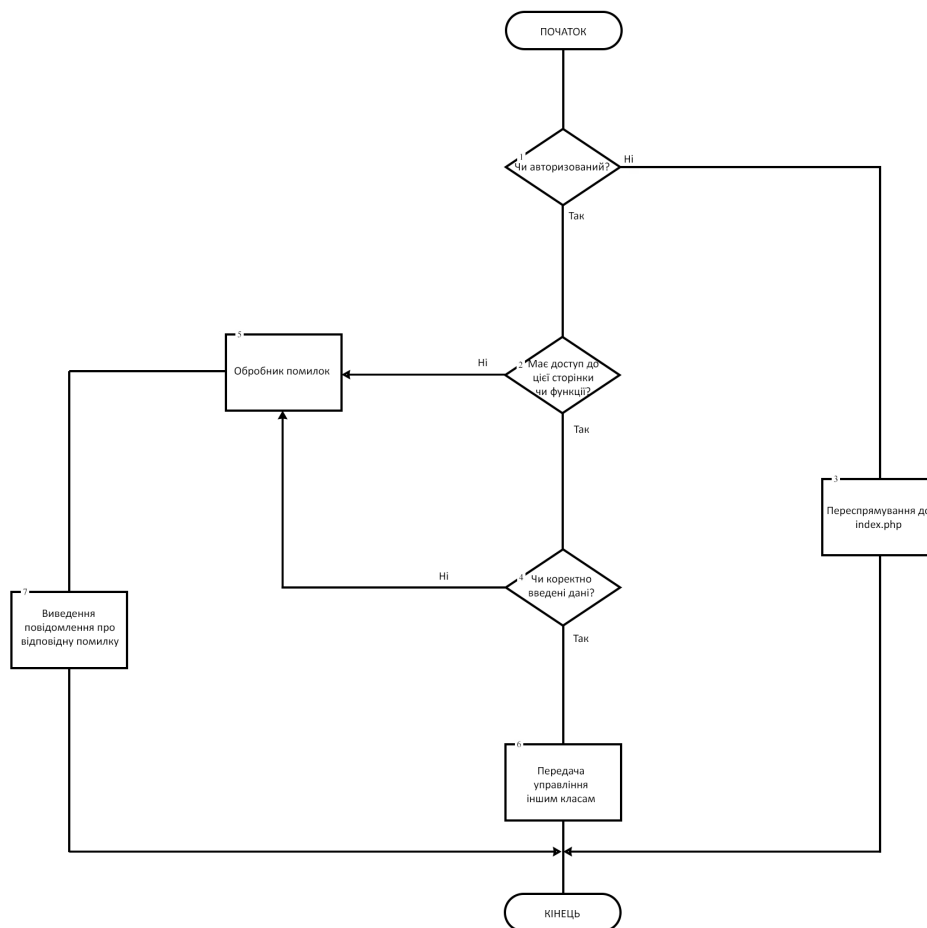


Рисунок 2.4 Основний алгоритм перевірки даних

На рисунку 2.4 зображен алгоритм перевірки даних, який використовує контролер actions.php при отриманні даних від користувача. У секції “Чи коректно введені дані” проводяться кілька перевірок. Наприклад чи існує певний ключ в переданому масиві даних, або чи є числом вміст ключа масива даних.

## 2.3 Програмна реалізація загальних функцій системи

Перша функція яку користувач побачить, це функція авторизації. На цій сторінці знаходиться форма для вводу логіна, пароля та чек-бокс “запомни меня”.

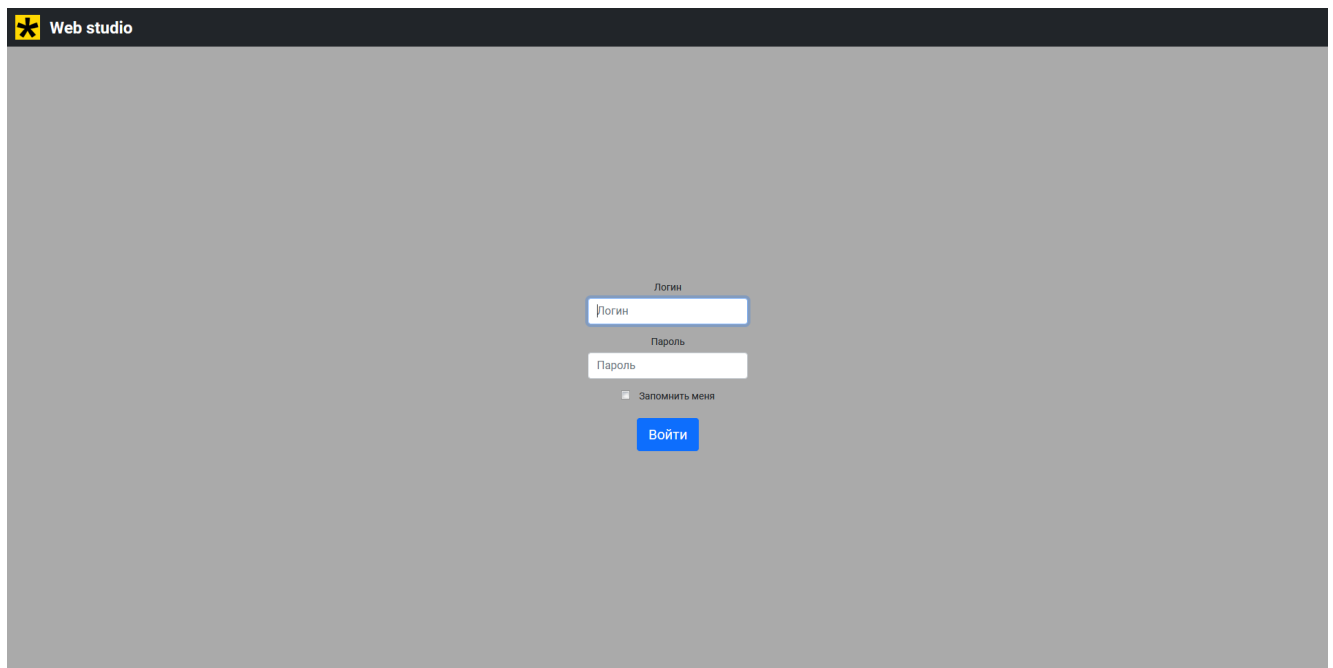


Рисунок 2.5 – Вигляд сторінки авторизації

На рисунку 2.5 зображена сторінка авторизації. Вона з’являється при переході на будь яку сторінку системи, якщо в браузері відсутній “cookie” що містить код сесії поточного користувача. Після натискання кнопки Войти, клієнт посилає введені користувачем дані на сервер де вони проходять перевірку на збіг із значеннями бази даних. Слід зауважити що на кожній сторінці присутній “header”. Його наповнення залежить від поточної сторінки та від ролі користувача який знаходиться у системі. Тільки у адміністратора присутня кнопка “Пользователи”.

### Лістинг 2.1 – Авторизація (actions.php)

```
case 'login':  
    if(empty($_POST['login']) or empty($_POST['password'])) {
```

```

        Pages::set_message('Пустой логин или пароль', false);
        header('Location: index.php?page=login');
    }
    $login = $_POST['login'];
    $password = $_POST['password'];
    $user = new User;
    if($user->login($login, $password)){
        header('Location: index.php');
    } else {
        Pages::set_message('Логин или пароль не верны!', false);
        header('Location: index.php?page=login');
    }
    break;

```

## Лістинг 2.2 – Авторизація (PHP та SQL запити)

```

public function login($login, $password) {
    $remember_me = (!empty($_POST['remember_me'])) ? '1' :
'0';

    $sth = db::i()->prepare("SELECT id, password FROM users
WHERE login = :login AND active = '1'");
    $res = $sth->execute( array('login' => $login) );
    if(!$res){
        db::er($sth->errorInfo());
    }
    $rows = $sth->fetchAll(PDO::FETCH_ASSOC);
    if(count($rows) == 0){
        return false;
    }
    if (password_verify($password, $rows[0]['password'])) {
        $this->set_session($rows[0]['id'], $remember_me);
        return true;
    } else {
        return false;
    }
}

```

У лістингу 2.1 перевіряється наявність логіну чи паролю. За відсутності одного чи іншого, з'явиться повідомлення “Пустой логин или пароль” і авторизація не відбудеться. Далі якщо пароль і логін дійсні, відбудеться авторизація, а якщо ні, з'явиться повідомлення “Логин или пароль не верны!”.

Якщо користувач успішно авторизувався він має можливість продовжити роботу в системі або вийти з системи.

## Лістинг 2.3 Функція виходу з системи (контроллер actions.php)

```

case 'logout':

```

```

        $user = new User;
        $user->logout();
        header('Location: index.php');
    break;

```

#### Лістинг 2.4 Функція виходу з системи (user.php)

```

public function logout() {
    if (empty($_COOKIE[USER_COOKIE_NAME])) {
        return;
    }
    $user_token = $_COOKIE[USER_COOKIE_NAME];
    $sth = db::i()->prepare("DELETE FROM sessions WHERE
user_token = :user_token");
    $res = $sth->execute(array(':user_token' =>
$user_token));
    if (!$res) {
        db::er($sth->errorInfo());
    }
    setcookie(USER_COOKIE_NAME, '', time() - 3600);
}

```

У лістингу 2.3 коли контролер отримує команду `logout`, він передає управління методу “`logout`” класу `user.php`, код якого наведено в лістингу 2.4, далі я метод видаляє запис сесії поточного користувача з бази даних та змінює “`cookie`” користувача на порожнє з негативним часом життя, тим самим знищуючи його з браузера.

Також в системі є функція відображення повідомлень в залежності від результату виконання, чи помилки при виконанні, тієї чи іншої функції системи. Повідомлення за замовчуванням з’являються у верхньому правому кутку вікна браузера.

#### Лістинг 2.5 Функція відображення повідомлень (pages.php)

```

public static function set_message($message = '', $status =
true) {
    setcookie(MESSAGE_COOKIE_NAME,
json_encode(array('message' => $message, 'status' => $status)),
time()+MESSAGE_COOKIE_LIFETIME);
    return true;
}

```

### Лістинг 2.6 Функція відображення повідомлень (JavaScript main.js)

```
function show_message() {
    if( $('#display-message').html().length ) {
        $('#display-message').slideToggle()
        setTimeout(function() {
            $('#display-message').slideToggle()
        }, 3000);
    }
}
```

У лістингу 2.5 наведений код методу “set\_message” класу pages.php при виклику якого, створюється “cookie” з текстом повідомлення і невеликим часом існування, після чого в main.js виконується відображення цього повідомлення, а точніше анімація розгортання зверху до низу і при закінченні часу існування, анімація згортання знизу до верху.

Наступна сторінка яку користувач побачить це сторінка проектів.

## 2.4 Програмна реалізація функцій проектів та задач

Функції проектів та задач спрямовані на надання можливості ведення обліку існуючих проектів та задач, створення нових проектів та задач, їх редагування та головна функція, ведення обліку часу затраченого на виконання тієї чи іншої задачі.

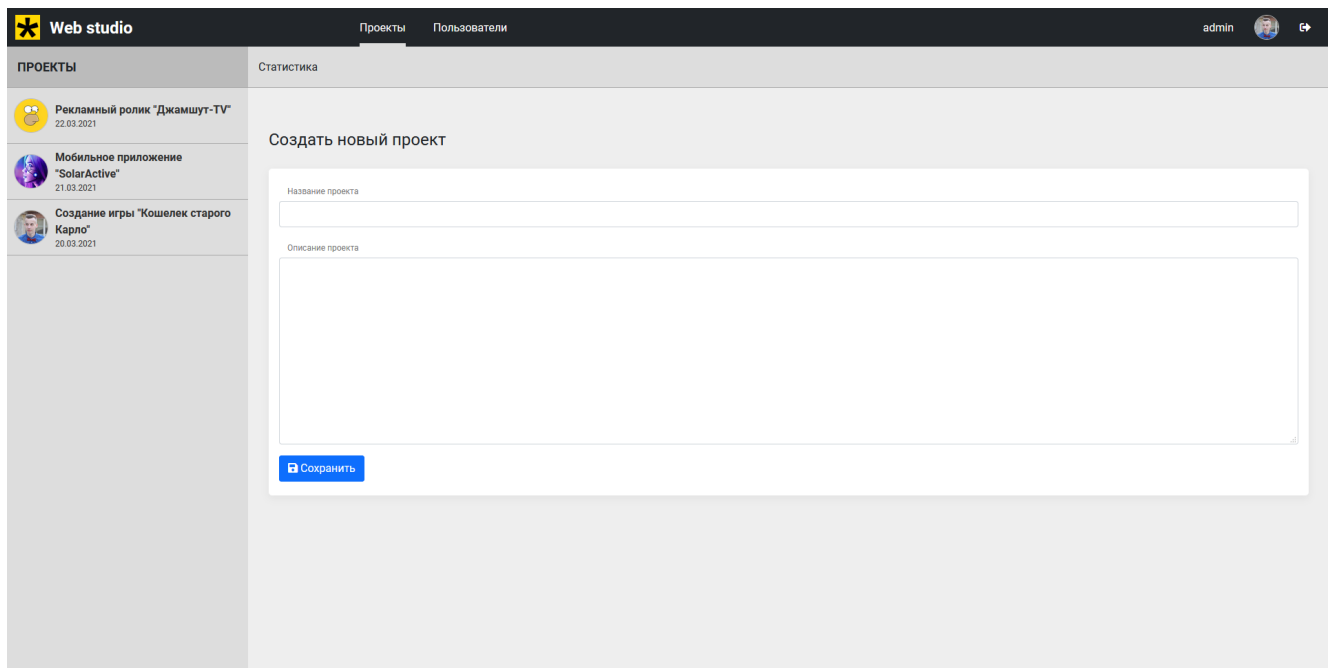


Рисунок 2.6 – Вигляд сторінки проектів

На рисунку 2.6 зображена сторінка проектів. Більшу частину екрану займає форма створення нового проекту. Ліву частину сторінки займає перелік проектів.

#### Лістинг 2.6 – Отримання переліку проектів (projects.php)

```
public static function get_list(){
    $sth = db::i()->prepare( "SELECT * FROM projects ORDER
BY creation_date DESC" );
    $res = $sth->execute( array() );
    if(!$res){
        db::er($sth->errorInfo());
    }
    $rows = $sth->fetchAll(PDO::FETCH_ASSOC);
    foreach($rows as $row ){
        $out[$row['id']] = $row;
    }
    return $out;
}
```

Натиснувши на будь який з проектів користувач потрапляє на сторінку цього проекту.

Наступна функція, це функція створення нового проекту.

### Лістинг 2.7– Створення нового проекту (projects.php)

```

public static function add_project( $name, $description,
$user_id ){
    $sth = db::i()->prepare( "INSERT INTO projects (name,
creation_date, description, user_id) VALUES (:name, NOW(),
:description, :user_id )" );
    $res = $sth->execute( array(':name' => $name,
':description' => $description, ':user_id' => $user_id) );
    if(!$res){
        db::er($sth->errorInfo());
    }
    $id = db::i()->lastInsertId();
    return $id;
}

```

У лістингу 2.6 метод отримує назву та опис проекту, яку написав користувач у відповідній формі, а також ід цього користувача, як ід того хто буде керувати цим проектом. Після чого отримні дані відправляються через SQL запит в базу даних сервера де створюється новий запис і зберігаються в нього.

### Лістинг 2.8 - Збереження проекту (actions.php)

```

case 'save_project':
    if( !$user_id or !User::allowed(['admin', 'manager'])
){
        break;
    }
    if( !array_key_exists('project_id', $_POST) ){
        break;
    }

    $project_id = $_POST['project_id'];
    $name = filter_var($_POST['name'],
FILTER_SANITIZE_STRING);
    //$description = filter_var($_POST['description'],
FILTER_SANITIZE_STRING);
    $description = htmlspecialchars($_POST['description']);
    if( $project_id != '' ){
        //edit
        if(!is_numeric($project_id)){
            break;
        }
        Projects::save_project($project_id, $name,
$description);
        header('Location:
index.php?page=projects&id='.$project_id);
    } else {
        //add
    }

```

```

        $new_id = Projects::add_project($name,
        $description, $user_id);
        header('Location:
index.php?page=projects&id='.$new_id);
    }
    $ret = array('status' => 'success');
    header('Content-Type: application/json');
    echo json_encode($ret);
    break;

```

### Лістинг 2.9 – Збереження проекту (PHP та SQL запити)

```

    public static function save_project( $id, $name, $description
    ){
        $sth = db::i()->prepare( "UPDATE projects SET name =
:name , description = :description WHERE id = :id" );
        $res = $sth->execute( array(':id' => $id, ':name' =>
$name, ':description' => $description) );
        if(!$res){
            db::er($sth->errorInfo());
        }
        return true;
    }
    public static function add_project( $name, $description,
$user_id ){
        $sth = db::i()->prepare( "INSERT INTO projects (name,
creation_date, description, user_id) VALUES (:name, NOW(),
:description, :user_id )" );
        $res = $sth->execute( array(':name' => $name,
':description' => $description, ':user_id' => $user_id) );
        if(!$res){
            db::er($sth->errorInfo());
        }
        $id = db::i()->lastInsertId();
        return $id;
    }

```

У лістингу 2.8 перевіряється доступ користувача до цієї функції, якщо є, то далі йде фільтрація даних користувача. Фільтрація необхідна для запобігання передачі можливих рядків коду, які при передачі на сервер можуть спричинити порушення цілісності даних. Якщо усі перевірки пройдені успішно, викликається метод `save_project`, якщо був переданий `id` проекту, або `add_project`, якщо `id` не був переданий, класу `Projects.php`. У лістингу 2.9 наведений код методів `save_project` та `add_project` класу `Project.php`. При їх виклику виконуються SQL запити для збереження змін в проекті чи збереженні нового проекту відповідно.

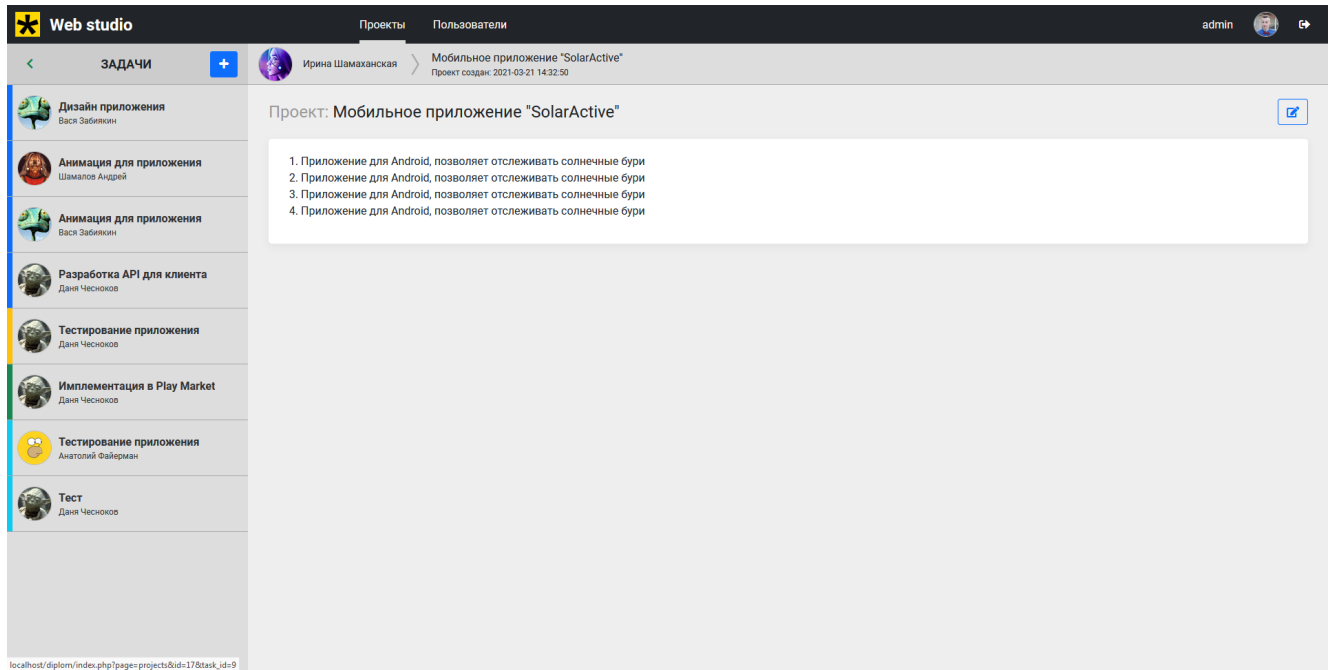


Рисунок 2.7 – Вигляд сторінки проекту

На рисунку 2.7 зображена сторінка вибраного проекту. Більшість простору на сторінці займає основна область на якій при переході знаходиться назва проекту та його опис. В правому верхньому кутку, над описом проекту, знаходиться кнопка редагування опису проекту. Натиснувши її, область з описом зміниться на редактор в який автоматично підставляється вже існуючий опис та назва проекту. Зліва знаходиться перелік задач до вибраного проекту. Картинка у кружечку це аватар користувача який виконує це завдання, кольорова стрічка зліва від аватару це стан задачі. Над цим списком знаходяться дві кнопки, стрілочка вліво та білий плюс на синьому фоні. Стрілочка повертає користувача на сторінку із списком проектів, а плюс заміняє основну область на форму для створення нового завдання. Важливо зауважити, усі форми поділяються на форми редагування та форми створення. Різниця між ними лише в переданих даних, у форму редагування передаються дані об'єкту що редагується і поля автоматично заповнюються існуючою інформацією, а у форму створення не передається дані і після натисання кнопки "Сохранить" клієнт посилає запит на сервер де

перевіряється доступ користувача до цієї функції і якщо доступ є, то проводиться запис до бази даних.

Рисунок 2.8 – Вигляд форми редагування задачі

На рисунку 2.8 зображена форма редагування задач, в якій окрім необхідних полів треба вибрати виконавця. У цій формі під областю редагування опису знаходяться поля пошуку по імені чи навичкам та кнопка, натиснувши на яку нижче з'явиться список усіх користувачів.

#### Лістинг 2.10 – Пошук користувачів за ім'ям (контролер actions.php)

```
case 'search_user_by_name':
    if( !$user_id or empty($_POST['name']) ) {
        break;
    }
    $name = filter_var($_POST['name'], FILTER_SANITIZE_STRING);
    $res = User::search_user_by_name($name);
    $ret = array('status' => 'success', 'users' => $res);
    header('Content-Type: application/json');
    echo json_encode($ret);
    break;
```

### Лістинг 2.11 – Пошук користувачів за ім'ям (PHP – MySQL запит)

```
public static function search_user_by_name($name){
    $sth = db::i()->prepare("SELECT id, login, name, user_pic
FROM users WHERE login LIKE :name OR name LIKE :name");
    $res = $sth->execute( array(':name' => '%'.$name.'%') );
    $rows = $sth->fetchAll(PDO::FETCH_ASSOC);
    $out = array();
    foreach($rows as $row ){
        $out[$row['id']] = $row;
    }
    return $out;
}
```

### Лістинг 2.12 – Пошук користувачів за ім'ям (JavaScript - JQuery)

```
function get_performers(data){
    $.ajax({
        method: "POST",
        url: "actions.php",
        data: data,
        dataType: 'json'
    })
    .done(function( res ) {
        if(res.status == 'success' && res.users ){
            $('#performer-list').empty()
            $('input[name="task_performer_id"]').val('')
            $.each(res.users, function(index, value) {
                var user_pic = value.user_pic ?
'upload/'+value.user_pic : 'assets/images/default-user-pic.png'
                $('#performer-list').append( $('<div
class="performer-item" data-userid="'+value.id+'"><div class="user-
pic"></div><div class="user-name-login"><div
class="user-name">'+value.name+'</div><div class="user-
login">'+value.login+'</div></div></div>'))
            });
        }
    })
}
```

У вище наведених лістингах наведено як саме відбувається пошук за ім'ям. У лістингу 2.10 контролер отримує введені користувачем дані. Далі йде валідація даних та передача управління методу класу Users. У лістингу 2.11 в методі search\_user\_by\_name відбувається SQL запит для вибору усіх користувачів у яких в імені частково є те що написав користувач у полі пошуку. У лістингу 2.12 використовується JQuery метод XHR запитів(Ajax). XHR (XMLHttpRequest) - API, доступний в скриптових мовах браузерів, таких як JavaScript. Використовує

запити HTTP або HTTPS безпосередньо до веб-сервера і завантажує дані відповіді сервера безпосередньо в викликаючий скрипт. Інформація може передаватися в будь-якому текстовому форматі, наприклад, в XML, HTML або JSON. Дозволяє здійснювати HTTP-запити до сервера без перезавантаження сторінки. В цьому випадку, коли метод отримує успішну відповідь від сервера, виконується метод done. Цей метод є “обіцянкою”, якщо метод запиту отримує успішну відповідь, він виконує код який знаходиться в середині цієї обіцянки. Результатом буде згенеровані блоки користувачів.

Новая задача

Название Задачи

Срок выполнения

Описание Задачи

← → B I U 12pt Paragraph

Р 0 WORDS. POWERED BY TINYMCE

Назначить исполнителя

Шам Поиск по навыкам Показать всех

Ирина Шамаханская klava Шамалов Андрей shaman

Сохранить

Рисунок 2.9 Результат поиска пользователей по имени

На рисунку 2.9 зображений результат пошуку користувачів по імені у формі редагування задачі. Якщо натиснути на будь яку задачу основна область зміниться на область управління цією задачею. У цій області можливо змінити стан задачі, редагувати цю задачу та розпочати роботу над цією задачею. Перед тим як розпочати роботу, можливо написати опис того що користувач збирається робити в наступну сесію роботи(спринт). Після натискання кнопки Начать. Після цього почнеться підрахунок часу, скільки користувач працював над цією задачею, а кнопка Начать змінить колір на жовтий і замість Начать буде Приостановить.

Якщо натиснути її, підрахунок завершиться, кнопка повернеться в базовий стан а у журналі праці з'явиться новий запис, в якому буде дата та час початку та кінця цієї сесії а також сумарний витрачений час на цю сесію.

### Лістинг 2.13 – Початок роботи (actions.php)

```

case 'start_work':
    if( !$user_id or empty($_POST['task_id']) ){
        break;
    }
    $task_id = filter_var($_POST['task_id'],
FILTER_SANITIZE_NUMBER_INT);
    $task = Projects::get_task($task_id);
    if(!empty($_POST['description'])){
        $work_description =
filter_var($_POST['description'], FILTER_SANITIZE_STRING);
    }else{
        $work_description = '';
    }
    if( $user['role'] == 'user' and $task['user_id'] !=
$user_id ){
        $ret = array('status' => 'error', 'message' => 'Вы
не можете выполнить эту задачу');
        header('Content-Type: application/json');
        echo json_encode($ret);
        die;
    }
    $work_id = Projects::start_work($task_id, $user_id,
$work_description);
    $ret = array('status' => 'success', 'work_id' =>
$work_id);
    header('Content-Type: application/json');
    echo json_encode($ret);
    break;

```

### Лістинг 2.14 – Початок роботи (PHP та SQL запити)

```

public static function start_work($task_id, $user_id,
$description = ''){
    $sth = db::i()->prepare( "UPDATE tasks SET status =
:status WHERE id = :id" );
    $res = $sth->execute( array(':status' => 'in_progress',
':id' => $task_id ) );
    if(!$res){
        db::er($sth->errorInfo());
    }
    $sth = db::i()->prepare( "INSERT INTO worklog (task_id,
user_id, description) VALUES (:task_id, :user_id, :description)" );
    $res = $sth->execute( array(':task_id' => $task_id,
':user_id' => $user_id, ':description' => $description ) );

```

```

    if (!$res) {
        db::er($sth->errorInfo());
    }
    $id = db::i()->lastInsertId();
    return $id;
}

```

## Лістинг 2.15 – Початок роботи (JavaScript - JQuery)

```

$('#submit-work').click(function() {
    var description = $('#work_description').val()
    var task_id = $(this).data('taskid')

    if ($(this).hasClass('btn-primary')) {
        $(this).removeClass('btn-primary')
        $(this).addClass('btn-warning')
        $(this).children('span').text('Приостановить')
        $(this).children('i').removeClass('fa-play-circle')
        $(this).children('i').addClass('fa-pause-circle')
        $('#work_description').attr("disabled", true);
        start_work(task_id, description)

    } else {
        var work_id = $('input[name="work_id"]').val()
        stop_work(work_id)
        $(this).removeClass('btn-warning')
        $(this).addClass('btn-primary')
        $(this).children('span').text('Начать')
        $(this).children('i').removeClass('fa-pause-
circle')

        $(this).children('i').addClass('fa-play-circle')
        $('#work_description').val('')
        $('#work_description').attr("disabled", false)
    }
})

function start_work(task_id, description) {
    $.ajax({
        method: "POST",
        url: "actions.php",
        data: { action: 'start_work', task_id: task_id,
description: description },
        dataType: 'json'
    })
    .done(function( res ) {
        if (res.status == 'success' && res.work_id) {
            $('input[name="work_id"]').val(res.work_id)
            return true
        } else {
            return false
        }
    })
}

```

У лістингу 2.13 перевіряється чи авторизований та перевірка дійсності задачі. Далі йде фільтрація даних які користувач залишив у полі опису. Наступний крок це перевірка чи може користувач виконати це завдання. У будь якому негативному випадку, система не дасть користувачу можливість розпочати це завдання, якщо результат позитивний, дані передаються в метод `start_work` класу `projects`. У лістингу 2.14 виконуються SQL запити для збереження часу початку праці та опису спринта. У лістингу 2.15 виконується аякс запит до сервера і при успішній відповіді запускається лічильник часу та кнопка Начать змінюється на Приостановить.

Важливо зауважити, що при створенні сторінок проектів та задач, форми створення чи редагування проектів та задач були написані як окремі шаблони в інших файлах.

#### Листинг 2.16 – HTML код форми редагування проектів

```
<div id="edit-project" class="">
  <div class="project-details-headline">
    {if $project_data['id'] == ''}
      <h4>Создать новый проект</h4>
    {else}
      <h4>Изменить проект</h4>
    {/if}
  </div>
  <div class="content-box">
    <form class="main-form" action="actions.php"
method="POST">
      <input type="hidden" name="action"
value="save_project">
      <input type="hidden" name="project_id"
value="{ $project_data['id'] }">
      <div class="row pb-3">
        <div class="col">
          <label for="project-name">Название
проекта</label>
          <input type="text" id="project-name"
name="name" class="form-control" value="{ $project_data['name'] }"
maxlength="100" autocomplete="off" required>
        </div>
      </div>
      <div class="row">
        <div class="col">
```

```

                                <label for="description">Описание
проекта</label>
                                <textarea class="form-control"
name="description" maxlength="2000" id="description"
rows="10">{$project_data['description']}</textarea>
                                </div>
                            </div>
                            <div class="row pt-3">
                                <div class="col d-flex justify-content-
between">
                                    {if $project_data['id'] != ''}
                                    <button class="btn btn-danger"
type="button" data-bs-toggle="modal" data-bs-target="#delete-
project-modal"><i class="fas fa-trash-alt"></i> Удалить
проект</button>
                                    {/if}
                                    <button id="submit-button" class="btn btn-
primary"><i class="fas fa-save"></i> Сохранить</button>
                                </div>
                            </div>
                        </form>
                    </div>
                </div>
            </div>

```

Якщо уважно переглянути код то можна помітити, що в коді присутні фрагменти які не є HTML кодом, наприклад рядок `{if $project_data['id'] == ''}`. Ці рядки, це фрагменти PHP коду які SMARTY використовує при створенні сторінки. Саме за допомогою SMARTY при відкриванні, по суті однієї і тієї ж форми, в залежності від переданих даних, а точніше в цьому випадку, id проекту, користувач бачить різний результат. Якщо id проекту був переданий, відкривається форма редагування цього проекту і поля автоматично заповнюються. Якщо ж id не був переданий, відкривається форма створення проекту і усі поля за замовчуванням порожні. Підключення SMARTY до окремих файлів шаблонів разом із підключенням змінних SMARTY виконується завдяки команді `assign`. Як приклад, перелік команд `assign` для сторінки проектів:

Лістинг 2.17 – Фрагмент коду контролера `Pages.php` що включає команди `assign` для сторінки проектів

```

$this->smarty->assign('status_list',
Projects::get_task_status_list());

```

```

$this->smarty->assign('task_data', $task_data);
$this->smarty->assign('worklog', $worklog);
$this->smarty->assign('is_current_work_open',
$хis_current_work_open);
$this->smarty->assign('total_time', $total_time);
$this->smarty->assign('user', $user);
$this->smarty->assign('projects_list', $projects_list);
$this->smarty->assign('project_data', $project_data);
$this->smarty->assign('section', $section);

```

Якщо переглянути лістинг 2.16 і згадати рядок `{if $project_data['id'] == ''}` то з лістингу 2.17 стає зрозумілим, що команда `$this->smarty->assign('project_data', $project_data);` необхідна для роботи форми редагування проєктів.

## 2.5 Програмна реалізація функцій користувачів

У адміністратора є можливість перейти на сторінку користувачі для управління ними.

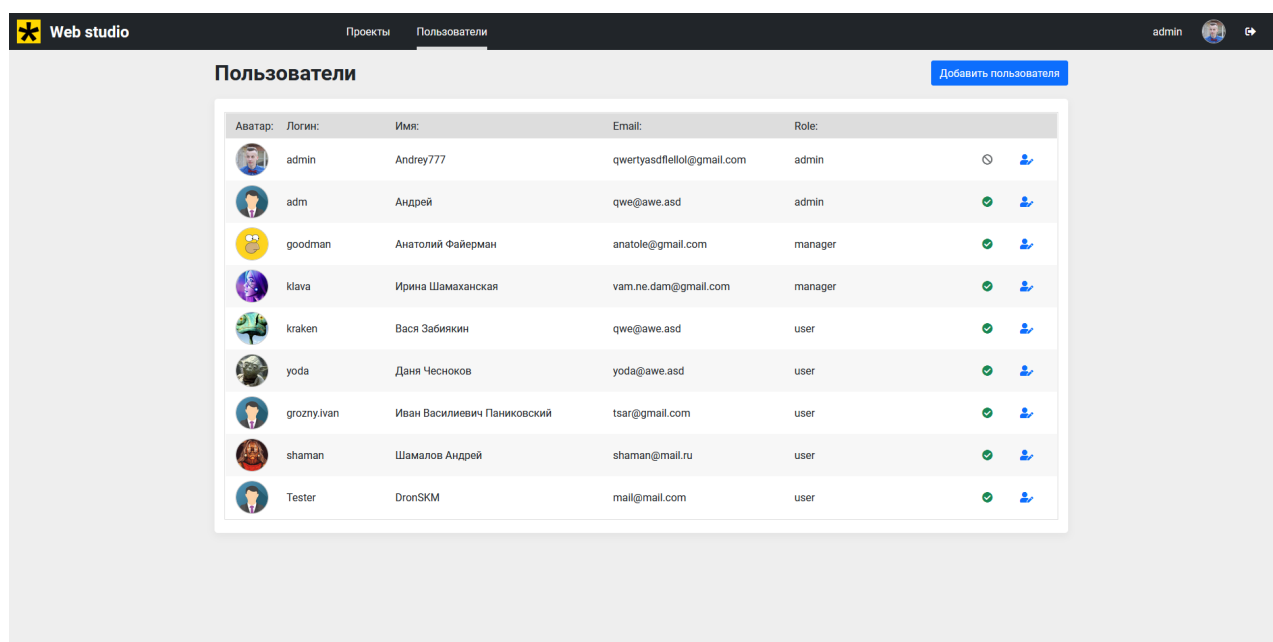


Рисунок 2.10 Вигляд сторінки користувачів

На рисунку 2.10 зображена сторінка користувачів. На цій сторінці знаходиться лише перелік користувачів. Справа зверху над цим переліком є кнопка для створення нового користувача. Навпроти кожного користувача, окрім їх даних, присутні 2 кнопки. Перша кнопка демонструє поточний стан користувача, тобто чи активний він, а при натисканні на неї цей статус змінюється на протилежний і так само змінюється картинка. Важливо зауважити, в цілях безпеки, функція деактивування себе була заблокована. Таким чином адміністратори не можуть випадково деактивувати себе. Користувач якого деактивовано автоматично виходить з системи і втрачає можливість авторизуватися до його активації. Друга кнопка замінює список на форму редагування користувача. Так само т

рапиться якщо натиснути кнопку створення нового користувача, але усі поля будуть за замовчуванням порожні.

#### Лістинг 2.18 – Функція відображення списку користувачів (user.php)

```
public static function get_users_list(){
    $sth = db::i()->prepare("SELECT * FROM users ORDER BY
role ASC");
    $res = $sth->execute( array() );
    if(!$res){
        db::er($sth->errorInfo());
    }
    $rows = $sth->fetchAll(PDO::FETCH_ASSOC);
    $out = array();
    foreach($rows as $row ){
        $out[$row['id']] = $row;
    }
    return $out;
}
```

#### Лістинг 2.19 – Збереження користувача (контроллер actions.php)

```
case 'save_user':
    if( !$user_id and !User::allowed('admin') ){
        break;
    }
    if(!array_key_exists('user-id', $_POST)){
        break;
    }
}
```

```

        if(empty($_POST['login']) or empty($_POST['name']) or
empty($_POST['email'])) {
            die("Недопустимые значения");
        }
        $roles = User::get_roles();
        if(empty($_POST['role']) and !in_array($_POST['role'],
$roles)){
            break;
        }

        $user_pic = '';
        $login = filter_var($_POST['login'],
FILTER_SANITIZE_STRING);
        $name = filter_var($_POST['name'],
FILTER_SANITIZE_STRING);
        $email = filter_var($_POST['email'],
FILTER_SANITIZE_EMAIL);
        $role = filter_var($_POST['role'],
FILTER_SANITIZE_STRING);

        if(!empty($_FILES['user_pic'])){
            if($_FILES['user_pic']['error'] != 4){
                if ( $_FILES['user_pic']['error'] != 0 ){
                    Pages::set_message('Ошибка загрузки файла',
false);

                    header('Location: index.php?page=users');
                    die;
                }
                if ($_FILES['user_pic']['size'] > 1000000) {
                    Pages::set_message('Превышен размер файла',
false);

                    header('Location: index.php?page=users');
                    die;
                }
                $path = $_FILES['user_pic']['name'];
                $ext = strtolower(pathinfo($path,
PATHINFO_EXTENSION));
                $valid_ext = array('jpg', 'jpeg', 'png', 'bmp',
'gif');
                if ( !in_array($ext, $valid_ext)){
                    Pages::set_message('Недопустимый формат
файла', false);

                    header('Location: index.php?page=users');
                    die;
                }
                $pic_name = User::create_random_name();
                $fileTmpName =
$_FILES['user_pic']['tmp_name'];
                $dst_file =
UPLOAD_FOLDER.'\\'.$pic_name.'.'.$ext;
                $didUpload = move_uploaded_file($fileTmpName,
$dst_file);

                if($didUpload == false){

```

```

        Pages::set_message('Ошибка сохранения
файла', false);

        header('Location: index.php?page=users');
        die;
    }
    $user_pic = $pic_name.'.'.$ext;
}

}

if(is_numeric($_POST['user-id'])){
    //edit user
    $password = '';
    if(!empty($_POST['password'])){
        $password = $_POST['password'];
    }
    if(!filter_var($_POST['email'],
FILTER_VALIDATE_EMAIL)){
        die('email is incorrect');
    }
    $user_id = $_POST['user-id'];

    $res = User::save_user(array('user-id' => $user_id,
'login' => $login, 'name' => $name, 'password' => $password, 'email'
=> $email, 'role' => $role, 'user_pic' => $user_pic));
    if($res){
        Pages::set_message('Изменения сохранены',
true);

        header('Location: index.php?page=users');
    } else {
        Pages::set_message('Ошибка сохранения данных',
false);

        header('Location: index.php?page=users');
    }
} else {
    //add user
    if(empty($_POST['password'])){
        Pages::set_message('Недопустимый пароль',
false);

        header('Location: index.php?page=users');
    }
    $password = $_POST['password'];
    $res = User::add_user(array('login' => $login,
'name' => $name, 'password' => $password, 'email' => $email, 'role'
=> $role, 'user_pic' => $user_pic));
    if($res){
        Pages::set_message('Пользователь создан',
true);

        header('Location:
index.php?page=users&action=edit&id='.$res);
    } else {
        Pages::set_message('Ошибка создания
пользователя', false);
    }
}

```

```

        header('Location: index.php?page=users');
    }
}

```

## Лістинг 2.20 – Збереження користувача (user.php)

```

public static function save_user($user_data) {
    if(empty($user_data['password'])){
        if($user_data['user_pic'] == ''){
            $sth = db::i()->prepare("UPDATE users SET login
= :login, name = :name, email = :email, role = :role WHERE id =
:id");
            $res = $sth->execute( array(':id' =>
$user_data['user-id'], ':login' => $user_data['login'], ':name' =>
$user_data['name'], ':email' => $user_data['email'], ':role' =>
$user_data['role']) );
        } else {
            $sth = db::i()->prepare("UPDATE users SET login
= :login, name = :name, email = :email, role = :role, user_pic =
:user_pic WHERE id = :id");
            $res = $sth->execute( array(':id' =>
$user_data['user-id'], ':login' => $user_data['login'], ':name' =>
$user_data['name'], ':email' => $user_data['email'], ':role' =>
$user_data['role'], ':user_pic' => $user_data['user_pic']) );
        }
    } else {
        if($user_data['user_pic'] == ''){
            $password_hash =
password_hash($user_data['password'], PASSWORD_BCRYPT);
            $sth = db::i()->prepare("UPDATE users SET login
= :login, password = :password, name = :name, email = :email, role =
:role WHERE id = :id");
            $res = $sth->execute( array(':id' =>
$user_data['user-id'], ':login' => $user_data['login'], ':password'
=> $password_hash, ':name' => $user_data['name'], ':email' =>
$user_data['email'], ':role' => $user_data['role']) );
        } else {
            $password_hash =
password_hash($user_data['password'], PASSWORD_BCRYPT);
            $sth = db::i()->prepare("UPDATE users SET login
= :login, password = :password, name = :name, email = :email, role =
:role, user_pic = :user_pic WHERE id = :id");
            $res = $sth->execute( array(':id' =>
$user_data['user-id'], ':login' => $user_data['login'], ':password'
=> $password_hash, ':name' => $user_data['name'], ':email' =>
$user_data['email'], ':role' => $user_data['role'], ':user_pic' =>
$user_data['user_pic']) );
        }
    }
    if(!$res){
        if($sth->errorInfo()[1] == 1062){
            return false;
        }else{

```

```

        db::er($sth->errorInfo());
    }
}
return true;
}

```

У лістингу 2.19 case “save\_user” виконує наступні дії при виклику. Спочатку перевіряється доступ поточного користувача до цієї функції. Далі введені цим користувачем дані проходять фільтрацію. Після чого починається процес збереження аватару користувача. Спочатку файл проходить перевірки розміру та формату. Якщо файл не пройде якусь з перевірок з’явиться повідомлення в якому буде сказано яка саме помилка. Далі файл перейменується на випадково сгенеровану послідовність символів і зберігається у папку upload де зберігаються усі аватри користувачів. Наступний крок, це перевірка чи це створення нового користувача чи редагування існуючого. Якщо редагування тоді введена адреса електронної пошти валідується і дані посилаються у клас user.php. А якщо створення, то відбувається все те ж що при редагуванні, окрім вмісту даних. При редагуванні передаються усі дані, а при створенні усі крім id оскільки в базі даних ще нема запису з цим користувачем. Далі у листингу 2.20 наведений код з SQL запитамі класа user.php. У цьому класі викликається метод “save\_user” в якому виконуються запити для збереження отриманих даних у базу даних. Якщо пароль не був переданий, це означає що зберігаються зміни при редагуванні без заміни паролю. Якщо пароль переданий це збереження нового користувача. У кожного варіанта є ще по 2 варіанти. Перший, це збереження користувача з базовим аватаром, який за замовченням присвоюється користувачеві якщо при створенні він не загрузив свій. А другий, це збереження користувача з власним аватаром. Після завершення з’явиться повідомлення відповідно до результату.

#### Лістинг 2.21 – Деактивувати користувача (actions.php)

```

case 'set_user_inactive':
    if( !$user_id and !User::allowed('admin') ){
        break;
    }
    if(!is_numeric($_POST['userid'])){
        $ret = array('status' => 'error', 'message' => 'Не
коректний id пользователя!');
    }

```

```

    } else {
        $user_id = $_POST['userid'];
        User::set_inactive($user_id);
        $ret = array('status' => 'success');
    }
    header('Content-Type: application/json');
    echo json_encode($ret);
break;

```

## Лістинг 2.22 – Деактивувати користувача (PHP та SQL запити)

```

public static function set_inactive($user_id){
    $sth = db::i()->prepare("UPDATE users SET active = '0'
WHERE id = :id");
    $res = $sth->execute( array(':id' => $user_id) );
    if(!$res){
        db::er($sth->errorInfo());
    }
    $sth = db::i()->prepare("DELETE FROM sessions WHERE
user_id = :user_id");
    $res = $sth->execute( array(':user_id' => $user_id) );
    if(!$res){
        db::er($sth->errorInfo());
    }
    return true;
}

```

## Лістинг 2.23 – Деактивувати користувача (JS - JQuery)

```

$('.user-status').click(function(){
    var userid = $(this).data('userid');
    var obj = $(this)
    if($(this).hasClass('text-danger')){
        var action = 'set_user_active'
    } else if($(this).hasClass('text-success')){
        var action = 'set_user_inactive'
    } else {
        return false
    }
    $.ajax({
        method: "POST",
        url: "actions.php",
        data: { action: action, userid: userid},
        dataType: 'json'
    })
    .done(function( msg ) {
        console.log( "Data Saved: " + msg.status)

        if(obj.hasClass('text-danger')){
            obj.addClass('text-success')
            obj.removeClass('text-danger')
            obj.children('.fas').addClass('fa-check-
circle')

```

```

        obj.children('.fas').removeClass('fa-stop-
circle')
        $('#display-message').html('<span
class="message-success">Пользователь активирован</span>')
        show_message()
    } else if(obj.hasClass('text-success')){
        obj.removeClass('text-success')
        obj.addClass('text-danger')
        obj.children('.fas').removeClass('fa-check-
circle')
        obj.children('.fas').addClass('fa-stop-circle')
        $('#display-message').html('<span
class="message-success">Пользователь деактивирован</span>')
        show_message()
    }
    });
})

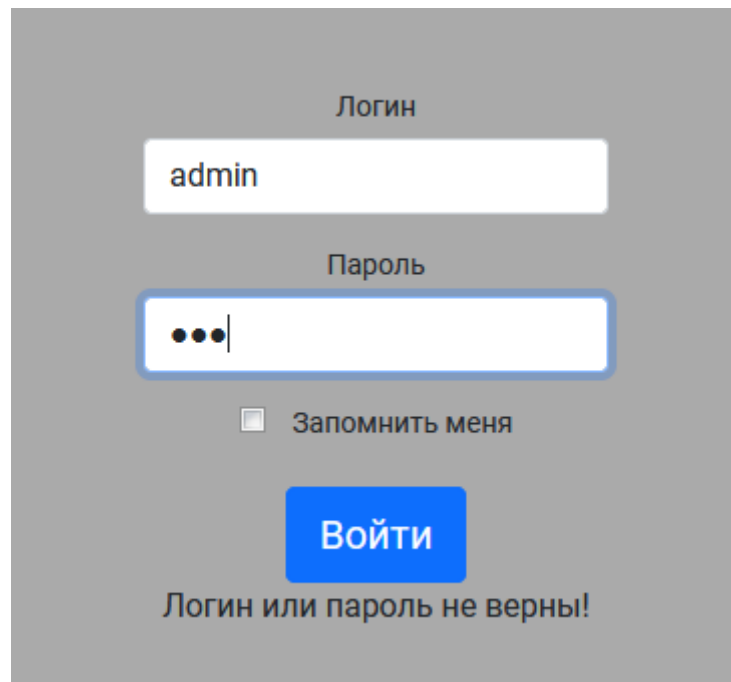
```

У лістингу 2.21 перевіряється чи є користувач адміністратором, якщо є, то перевіряє id користувача. Якщо id не існує то видасть помилку, якщо ж існує то викликається метод `set_inactive` класу `User`. У лістингу 2.22 вібдувається запит який оновлює стан користувача а також видаляє поточний запис сесії користувача, якщо такий існує. У лістингу 2.23 клієнт посилає аякс запит на сервер і при позитивній відповіді замінює знак статусу користувача на протилежний.

## 3 ТЕСТУВАННЯ РОЗРОБЛЕНОЇ СИСТЕМИ

### 3.1 Тестування авторизації

Перший важливий ряд тестів який був проведений, це тестування механізму авторизації. Спочатку було перевірено що трапиться при спробі авторизуватися з неправильним логіном чи паролем.



The image shows a login interface on a gray background. At the top, the label "Логин" (Login) is centered. Below it is a white input field containing the text "admin". Underneath the login field is the label "Пароль" (Password). Below the password label is a white password input field with three black dots and a cursor, indicating it is empty. Below the password field is a checkbox with the text "Запомнить меня" (Remember me). Below the checkbox is a blue button with the white text "Войти" (Login). At the bottom, the error message "Логин или пароль не верны!" (Login or password is incorrect!) is displayed in red.

Рисунок 3.1 Результат спроби авторизуватися з неправильним логіном чи паролем

Як видно на рисунку 3.1 така спроба завершується повідомленням “Логин или пароль не верны!”. Так само трапиться якщо не вводити пароль чи логін, оскільки порожній рядок також рахується як не дійсні дані. Далі біло перевірено, що трапиться якщо після успішної авторизації скопіювати cookie поточної сесії та вставити його після виходу з системи.

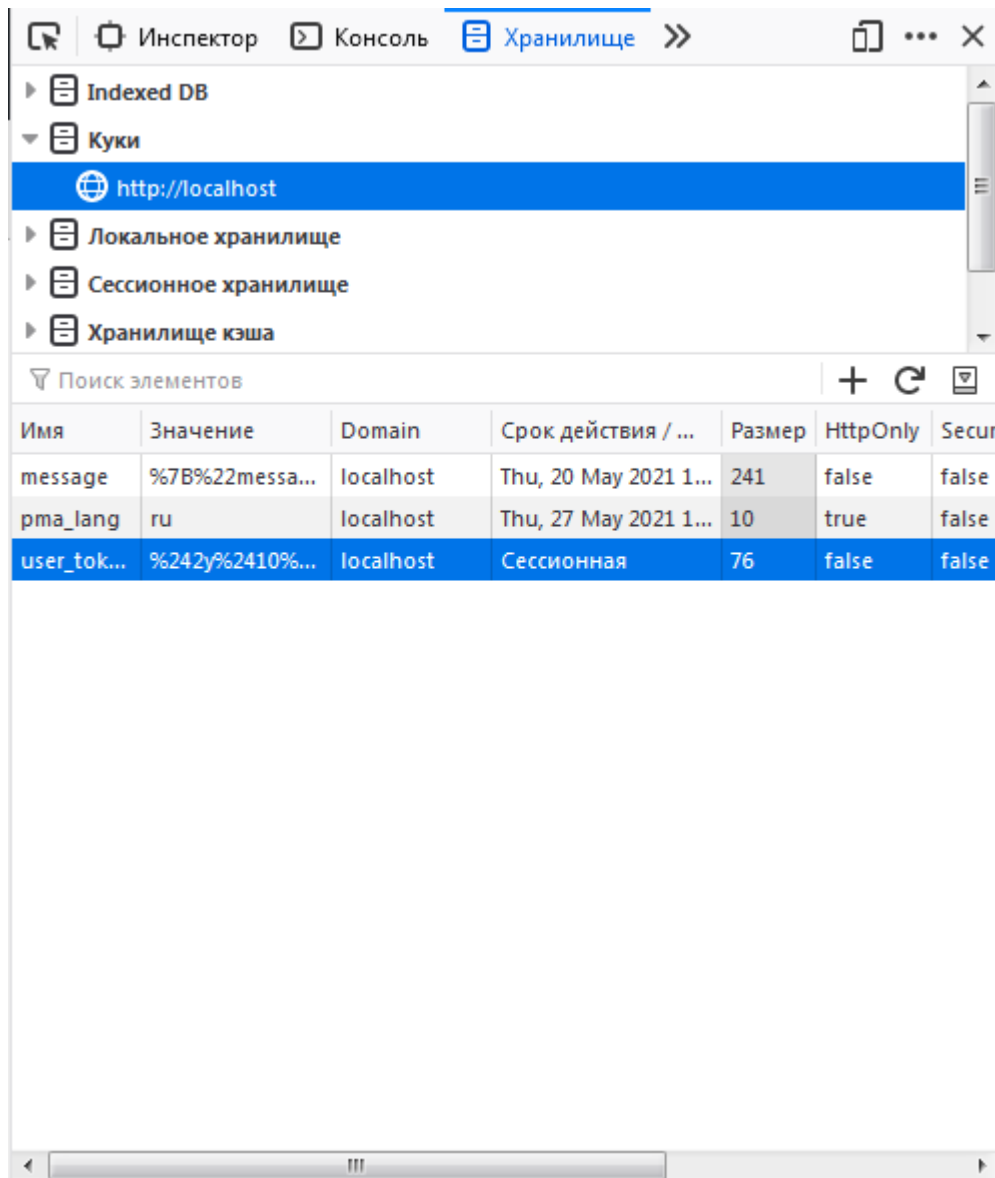


Рисунок 3.2 Cookie-файл поточної сесії

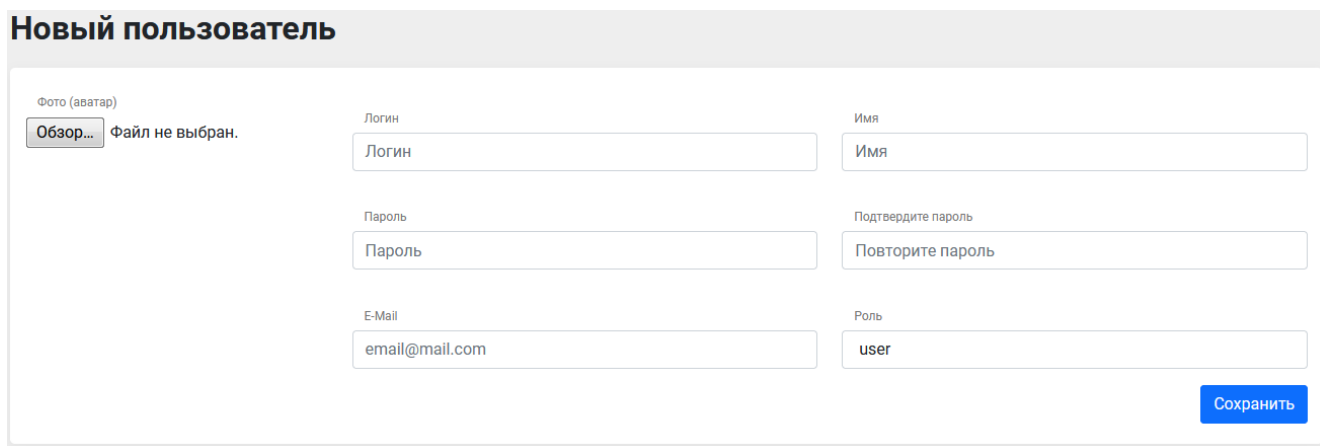
На рисунку 3.2 виділений той самий cookie-файл який відповідає за поточну сесію в системі. Якщо скопіювати хеш токен, видалити його та перезавантажити сторінку, то користувач потрапить на сторінці авторизації. Після цього можливо у порожнє поле, де раніше був хеш токен, вставити раніше скопійований хеш. Якщо так зробити та перезавантажити сторінку, знов відкриється головна сторінка системи. Так само можливо повністю видалити cookie-файл, оновити сторінку і побачити що потрібна повторна авторизація. Далі достатньо створити новий cookie-файл, назвати його user\_token та вставити раніше скопійований хеш

токен в поле Значение і після перезавантаження сторінки користувач буде авторизований.

Після цього було перевірено, що трапиться якщо авторизуватись в акаунт який вже авторизований з іншого пристрою. Якщо таке трапиться, то користувач який був авторизований першим, автоматично скасує свою авторизацію і потрапить на сторінку авторизації.

### 3.2 Тестування функцій взаємодії з користувачами

Перша головна функція взаємодії з користувачами яка була тестована, це функція створення нового користувача.



The image shows a web form titled "Новый пользователь" (New User) for creating a new account. The form is organized into three rows of input fields. The first row contains a photo upload section with a button "Обзор..." and the text "Файл не выбран.", a login field with the label "Логин" and placeholder "Логин", and a name field with the label "Имя" and placeholder "Имя". The second row contains a password field with the label "Пароль" and placeholder "Пароль", and a confirmation password field with the label "Подтвердите пароль" and placeholder "Повторите пароль". The third row contains an email field with the label "E-Mail" and placeholder "email@mail.com", and a role field with the label "Роль" and placeholder "user". A blue "Сохранить" (Save) button is located at the bottom right of the form.

Рисунок 3.3 Вигляд форми створення нового користувача

На рисунку 3.3 зображена форма створення нового користувача. Перший тест був спрямований на перевірку можливості створити користувача з повторюваним логіном.

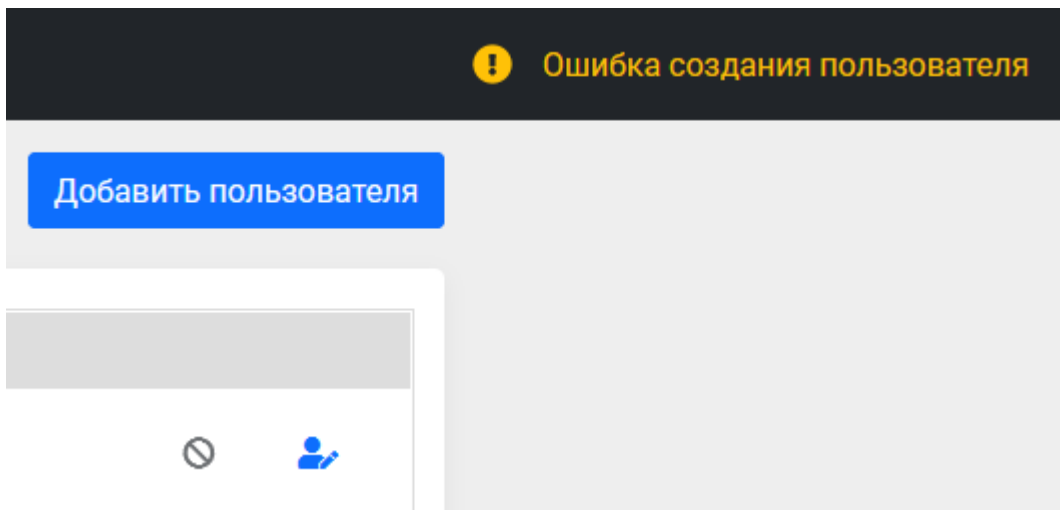


Рисунок 3.4 Повідомлення про помилку при створенні нового користувача

На рисунку 3.4 видно результат створення користувача з повторюваним логіном. Подібне повідомлення буде з'являтися при спробі зберегти користувача з аватаром, розмір якого перевищує 2мб, але замість “Ошибка создания пользователя” буде “Превишен размер файла”.

Далі було перевірено, що трапиться якщо вміст полей Пароль та Подтвердите пароль буде різним.

Рисунок 3.5 Перевірка на збіг вмісту полів паролей

Результат бачно на риунку 3.5, під полем Подтвердите пароль з'явиться надпис “Пароли не совпадают!”.

Остання перевірка була спрямована на виявлення можливості створити користувача з не валідною електронною поштою. Важливо зауважити, що валідна електронна пошта, це адреса яка відповідає зарзку: \*@\*.\*, де \* - будь який набір символів.

The screenshot shows a registration form with the following fields and values:

- Логин** (Login): Maximilian
- Имя** (Name): MadMax
- Пароль** (Password): [masked with dots]
- Подтвердите пароль** (Confirm password): [masked with dots]
- E-Mail**: asdf123.com
- Роль** (Role): user

A blue border highlights the E-Mail field. Below it, a tooltip message reads: "Пожалуйста, введите адрес электронной почты." (Please, enter the email address).

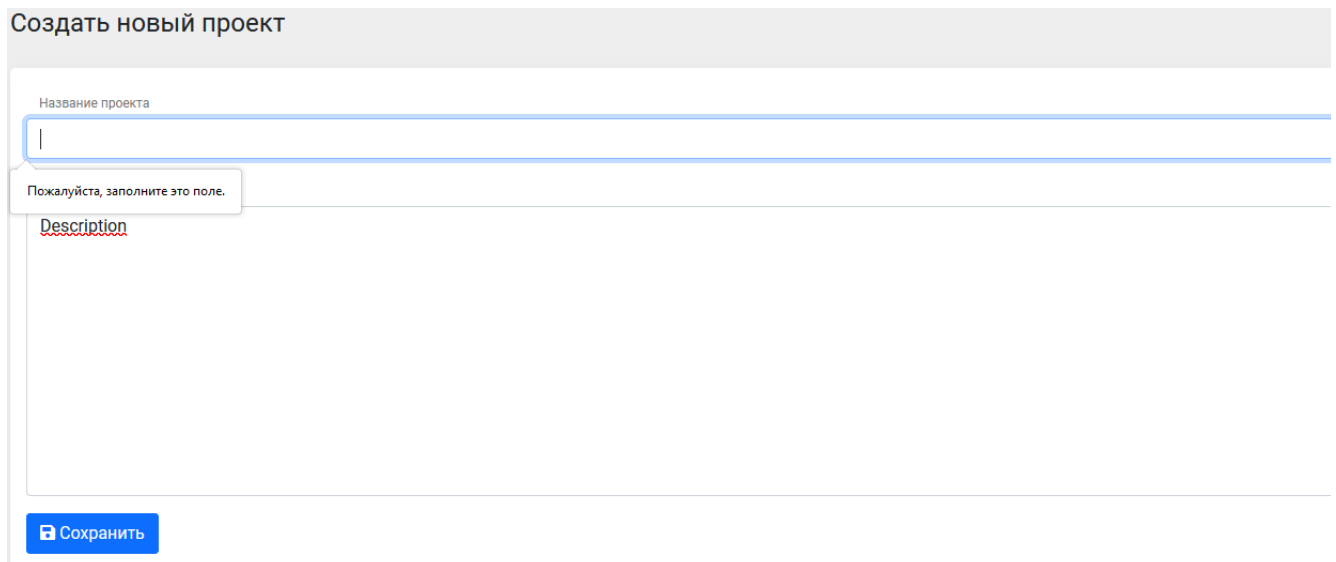
A blue button labeled "Сохранить" (Save) is located at the bottom right of the form.

Рисунок 3.6 Спроба створити користувача з не валідною поштою

Як видно на рисунку 3.6 при спробі створити користувача з не валідною електронною поштою з`явиться повідомлення під полем "E-Mail" з текстом "Пожалуйста, введите адрес электронной почты".

### 3.3 Тестування функцій взаємодії з проектами

Метою першої перевірки було виявлення можливості створювати проекти без назви.



Создать новый проект

Название проекта

Пожалуйста, заполните это поле.

Description

Сохранить

Рисунок 3.7 Спроба створити проект без назви

На рисунку 3.7 зображен результат спроби збереження проекту без назви. Результат це повідомлення “Пожалуйста, заполните это поле”. Результат був очікуваним оскільки у цього поля є параметр “required”. Якщо спробувати зберегти проект без назви при редагуванні існуючого проетку буде такий же результат.

Наступна перевірка була спрямована на виявлення вразливостей при збереженні даних користувача, а саме, що трапиться якщо в поля імені чи опису додати HTML теги.

Название проекта

"/>Мобильное приложение "SolarActive"

Описание проекта

↶ ↷ **B** *I* U ~~S~~ 12pt Paragraph

1. Приложение для Android, позволяет отслеживать солнечные бури"/>  
 2. Приложение для Android, позволяет отслеживать солнечные бури  
 3. Приложение для Android, позволяет отслеживать солнечные бури  
 4. Приложение для Android, позволяет отслеживать солнечные бури

OL » LI

Удалить проект

Рисунок 3.8 Приклад даних які можуть зламати форму

На рисунку 3.8 зображені дані які при збереженні у систему зламають зовнішній вигляд сторінки.

Проект: "/>Мобильное приложение "SolarActive"

1. Приложение для Android, позволяет отслеживать солнечные бури"/>  
 2. Приложение для Android, позволяет отслеживать солнечные бури  
 3. Приложение для Android, позволяет отслеживать солнечные бури  
 4. Приложение для Android, позволяет отслеживать солнечные бури

Рисунок 3.9 Результат збереження проекту з HTML тегами в назві та опису

На рисунку 3.9 видно, проект зберігся коректно та відображає HTML теги в назві та опису. Це сталося завдяки HTML сутностям (HTML entities)[17]. HTML сутність це послідовність символів яка замінює ключові HTML символи, такі як лапки.

### 3.4 Тестування функцій взаємодії з задачами

Перший ряд перевірок був спрямований на виявлення можливостей створення пустої задачі.

Новая задача

Название Задачи Срок выполнения

Описание Задачи

12pt Paragraph

Назначить исполнителя

Поиск по имени пользователя Поиск по навыкам Показать всех

Необходимо выбрать исполнителя

Сохранить

Рисунок 3.10 Спроба створити пусту задачу

На рисунку 3.10 зображен результат спроби збереження пустої задачі. Перший результат, це червоне повідомлення над кнопкою збереження, яке повідомляє про необхідність вибрати виконавця для цієї задачі.

Новая задача

Название Задачи

Срок выполнения

Пожалуйста, заполните это поле.

< >

**B** *I* U ~~S~~

12pt ▾ Paragraph ▾

≡ ≡ ≡ ≡

☰ ☷ ☷ ☷

A ▾ ▾ I<sub>x</sub>

{ }

< >

Р

Назначить исполнителя

Заб

Поиск по навыкам

Показать всех

Вася Забиякин  
kraken

Сохранить

Рисунок 3.11 Спроба створити задачу без назви

На рисунку 3.11 видно, що під полем “Название Задачи” з’явилося повідомлення “Пожалуйста заполните это поле”. Так само як у формі збереження проектів, поле назви задачі є обов’язковим.

# Новая задача

Название Задачи

проектирование основных функций

Срок выполнения

Описание Задачи

<

>

B

I

U

S

12pt

Paragraph

[List Icons]

[Text Color Icon]

[Background Color Icon]

[Link Icon]

[Image Icon]

[Video Icon]

ВсПнВтСрЧтПтСб

1

2345678

9101112131415

16171819202122

23242526272829

3031

P

0 WORDSPOWERED BY TINY

Назначить исполнителя

ЗабПоиск по навыкам

Показать всех

Avatar

Вася Забиякинkraken

Сохранить

Рисунок 3.12 Спроба створити задачу без терміну виконання

На рисунку 3.12 зображен результат спроби збереження задачі без терміну виконання. Як результат, з'являється таке ж повідомлення для обов'язкових полей та відкривається календар в якому необхідно вибрати термін виконання задачі. Після заповнення усіх необхідних полей, задача може бути створена.

Далі було перевірено, чи можуть користувачі виконувати чужі задачі. Кожному користувачеві видно весь список задач і відкривається кожна задача, але лише над тими задачами, на які користувач був призначений, можливо розпочати роботу чи змінити їх стан.

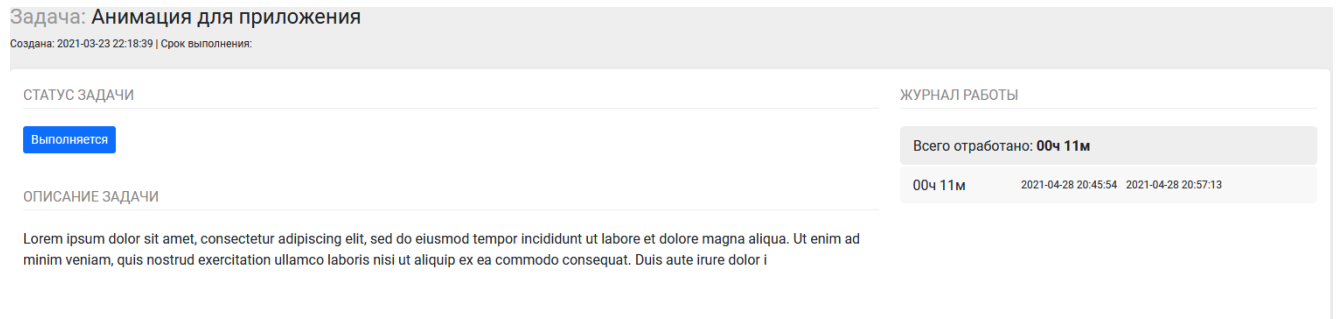


Рисунок 3.13 Вигляд чужої задачі

На рисунку 3.13 зображений вигляд задачі іншого виконавця. Остання перевірка взаємодії з задачами, це перевірка виконання роботи. Якщо розпочати роботу над задачею, тобто запустити лічильник в системі, та вийти з системи, лічильник не зупиниться. Далі можливо знову авторизуватися в систему і побачити що кнопка “Приостановить” змінилася на “Завершить”, тобто можливо завершити, раніше не завершено, роботу над задачею.

## РАБОЧИЙ ПРОЦЕСС

Опишите над чем работаете

⏸ Приостановить

## ЖУРНАЛ РАБОТЫ

Всего отработано: **00ч 22м**

|         |                     |                     |   |
|---------|---------------------|---------------------|---|
| 00ч 00м | 2021-03-29 18:45:56 | 2021-03-29 18:46:49 | ▼ |
| 00ч 21м | 2021-03-25 19:27:49 | 2021-03-25 19:49:19 | ▼ |

Рисунок 3.14 Розпочата робота над задачею до виходу з системи

На рисунку 3.14 зображений стан сторінки одразу після натискання кнопки “Начать” на сторінці задачі поточного користувача.

## РАБОЧИЙ ПРОЦЕСС

Опишите над чем работаете

🔒 Завершить

## ЖУРНАЛ РАБОТЫ

Всего отработано: **00ч 22м**

|         |                                            |   |
|---------|--------------------------------------------|---|
| 🕒       | 2021-05-21 21:00:27                        | ▼ |
| 00ч 00м | 2021-03-29 18:45:56    2021-03-29 18:46:49 | ▼ |
| 00ч 21м | 2021-03-25 19:27:49    2021-03-25 19:49:19 | ▼ |

Рисунок 3.15 Стан задачі через декілька хвилин після виходу з системи та повторної авторизації

Як видно на рисунку 3.15 в журналі роботи з'явився запис в якому зазначена дата та час початку виконання задачі. Після натискання кнопки “Завершить” кнопка знов зміниться на “Начать” та після перезавантаження сторінки поточний запис у журналі роботи зміниться на звичайний запис попередньої роботи.

## ВИСНОВКИ

В сфері it-розробок, інтернет технології займають одну з головних ролей. Саме зараз, коли весь світ почав підключатися до інтернету і люди почали проявляти великий інтерес до нього, підприємці помітили в цьому можливість покращити доступність інформації, щоб кожен хто має доступ до інтернету, міг знайти їх сайт і ознайомитись з необхідною інформацією. Саме через це в інтернеті можливо побачити все більше нових сайтів з різноманітним функціоналом.

У цій роботі був проведений огляд двох схожих систем управління проектами та задачами, які використовуються зараз у різних галузях. Також представлені їх функції, можливості та чим вони відрізняються.

В роботі також наведено розробку своєї системи управління, яка наслідує де-які функції аналогів, вдосконалює їх та робить їх більш простими для розуміння будь-якому користувачеві. Також інтегрована можливість управління користувачами в самій системі.

За допомогою цієї системи можна оптимізувати роботу з різними задачами в різних галузях шляхом раціонального розподілення виконавців на задачі. Це допоможе компаніям прискорити процес виконання їх задач чи замовлень що призведе до зменшення потрібного часу на виконання цільного проекту, а тому з'явиться більше часу на інші проекти.

## ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

- 1 - Вікіпедія, онлайн енциклопедія, SCRUM.URL:  
<https://ru.wikipedia.org/wiki/SCRUM>;
- 2 - Вікіпедія, онлайн енциклопедія, LAMP. URL:  
<https://uk.wikipedia.org/wiki/LAMP>;
- 3 - Вікіпедія, онлайн енциклопедія, XAMPP. URL:  
<https://ru.wikipedia.org/wiki/XAMPP>;
- 4 - Вікіпедія, онлайн енциклопедія, Операційна система Linux. URL:  
<https://uk.wikipedia.org/wiki/Linux>;
- 5 - Вікіпедія, онлайн енциклопедія, Apache HTTP Server. URL:  
[https://ru.wikipedia.org/wiki/Apache\\_HTTP\\_Server](https://ru.wikipedia.org/wiki/Apache_HTTP_Server);
- 6 - Вікіпедія, онлайн енциклопедія, MySQL. URL:  
<https://ru.wikipedia.org/wiki/MySQL>;
- 7 - Вікіпедія, онлайн енциклопедія, мова програмування PHP. URL:  
<https://ru.wikipedia.org/wiki/PHP>;
- 8 - Вікіпедія, онлайн енциклопедія, Smarty. URL:  
<https://uk.wikipedia.org/wiki/Smarty>;
- 9 - Вікіпедія, онлайн енциклопедія, мова розмітки HTML. URL:  
<https://ru.wikipedia.org/wiki/HTML>;
- 10 - Вікіпедія, онлайн енциклопедія, мова стилів CSS. URL:  
<https://uk.wikipedia.org/wiki/CSS>;
- 11 - Вікіпедія, онлайн енциклопедія, скриптова мова JavaScript. URL:  
<https://uk.wikipedia.org/wiki/JavaScript>;
- 12 - Вікіпедія, онлайн енциклопедія, JQuery фреймоврк. URL:  
<https://uk.wikipedia.org/wiki/JQuery>;
- 13 - Вікіпедія, онлайн енциклопедія, фреймворк Bootstrap. URL:  
[https://ru.wikipedia.org/wiki/Bootstrap\\_\(%D1%84%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA\)](https://ru.wikipedia.org/wiki/Bootstrap_(%D1%84%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA));

14 - Вікіпедія, онлайн енциклопедія, Model-View-Controller. URL: <https://ru.wikipedia.org/wiki/Model-View-Controller>;

15 – Офіційний сайт PHP, введення у PDO. URL: <https://www.php.net/manual/ru/intro.pdo.php>;

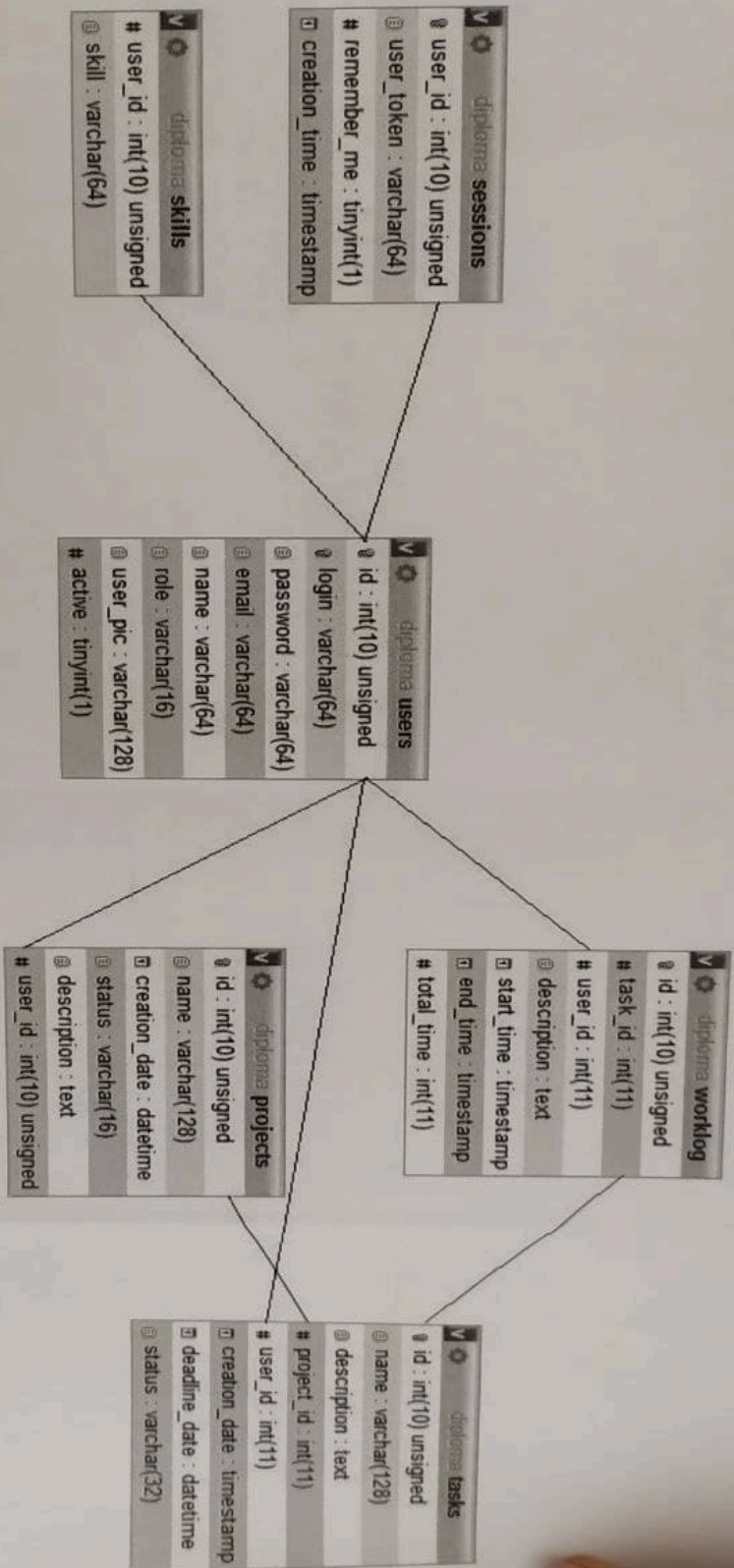
16 – Статистика популярності мов програмування. URL: <https://kenwood-bt.ru/info/statistika-populjarnosti-jazykov-programmirovanija/>;

17 – Офіційний сайт PHP, функція HTML сутностей htmlentities. URL: <https://www.php.net/manual/ru/function.htmlentities.php>;

Mozilla Developer Network - сховище документації і навчальний ресурс для веб-розробників. URL: <https://developer.mozilla.org/ru/>;

Сучасний підручник JavaScript. URL: <https://learn.javascript.ru/>;

Керівництво по PHP, офіційний сайт PHP. URL: <https://www.php.net/manual/ru/index.php>;



|            |                 |             |       |      |                                                                                                        |         |         |         |         |
|------------|-----------------|-------------|-------|------|--------------------------------------------------------------------------------------------------------|---------|---------|---------|---------|
| № п/п      | Датум           | № документа | Полн. | Лист | Разробка інвентаризації для ведення обліку, здійснення у відсотковій співпраці з сільськогосподарських | Звітн.  | Варіант | Місяць  | Місяць  |
| Розробка   | Територія д. т. | № документа | Полн. | Лист |                                                                                                        | Датум 1 | Датум 2 | Датум 3 | Датум 4 |
| Ліквідація | Класифікація    | № документа | Полн. | Лист |                                                                                                        | Датум 1 | Датум 2 | Датум 3 | Датум 4 |
| Т. копія   | № документа     | Полн.       | Лист  | Лист |                                                                                                        | Датум 1 | Датум 2 | Датум 3 | Датум 4 |
| Н. копія   | № документа     | Полн.       | Лист  | Лист |                                                                                                        | Датум 1 | Датум 2 | Датум 3 | Датум 4 |
| Звітн.     | № документа     | Полн.       | Лист  | Лист | Датум 1                                                                                                | Датум 2 | Датум 3 | Датум 4 | Датум 5 |





