

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

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

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ УПРАВЛІННЯ ВІДДАЛЕНОЮ
ЛАБОРАТОРІЄЮ RELDES
SOFTWARE FOR THE RELDES REMOTE LABORATORY CONTROL

Виконав: студент(ка) 4 курсу, групи КНТ-227
Спеціальності 122 Комп'ютерні науки
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Комп'ютерні науки

Терещенко О. В.

(прізвище та ініціали)

Керівник Пархоменко А. В.

(прізвище та ініціали)

Рецензент Морщавка С.В.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ІРЕ, ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 122 Комп'ютерні науки
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
“ ” 2021 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Терещенко Олени Віталіївни

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення для управління віддаленою лабораторією RELDES. Software for the RELDES remote laboratory control.

керівник проєкту (роботи) Пархоменко Анжеліка Володимирівна, к.т.н., доцент

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “30” березня 2021 року №103

2. Строк подання студентом проєкту (роботи) 17 травня 2021 року

3. Вихідні дані до проєкту (роботи) рекомендована література, завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Проєктування програмного забезпечення. 3. Конструювання програмного забезпечення. 4. Керівництво програміста. 5. Керівництво користувача.

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

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Пархоменко А.В., доцент		
Нормоконтроль	Дейнега Л.Ю., ст. викладач		

7. Дата видачі завдання “ 28 ” лютого 2020 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання
2	Аналіз предметної області	2-3 тижні	Розділ 1
3	Розробка архітектури програми	4 тиждень	Розділ 2
4	Розробка програми	5-7 тижні	Розділ 3
5	Розробка програмної документації	8 тиждень	Розділи 4,5
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	9 тиждень	ПЗ, Додатки
7	Захист роботи	10 тиждень	

Студент(ка)

(підпис) Терещенко О. В.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис) Пархоменко А.В
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 76 с., 37 рис., 6 табл., 2 дод., 39 джерел.

ВІДДАЛЕНА ЛАБОРАТОРІЯ, ЧЕРГА, ЕКСПЕРИМЕНТ, МЕТОДОЛОГІЯ ПОВТОРНОГО ВИКОРИСТАННЯ, ВЕБЗАСТОСУНОК, ФРЕЙМВОРК.

Об'єкт дослідження – процес управління функціонуванням віддаленої лабораторії.

Предмет дослідження – методи та засоби організації черги для виконання віддалених експериментів.

Мета роботи – розробка програмного забезпечення управління чергою користувачів для зручного виконання експериментів віддаленої лабораторії RELDES.

Матеріали, методи та технічні засоби: структурне та об'єктно-орієнтоване програмування, мова програмування PHP, фреймворк Laravel, JavaScript, мова розмітки HTML, персональний комп'ютер з процесором Intel Core i5 під управлінням операційної системи Microsoft Windows 10.

Результати. Створено модуль черги для віддаленої лабораторії RELDES, який може бути також впроваджено в інші віддалені лабораторії на основі методології повторного використання.

Висновки. Розроблено програмний застосунок, який надає можливість користувачу займати чергу для проведення експериментів віддаленої лабораторії RELDES та стежити за її станом.

Галузь використання – віддалені лабораторії.

ABSTRACT

Explanatory note to the diploma qualification work of the bachelor: 76 pages, 37 figures., 6 tables, 2 appendixes, 39 sources.

REMOTE LABORATORY, QUEUE, EXPERIMENT, REUSE METHODOLOGY, WEB APPLICATION, FRAMEWORK.

The object of research is the process of controlling the functioning of a remote laboratory.

The subject of research are methods and tools of queuing to perform remote experiments.

Materials, methods and technical tools: structural and object-oriented programming, PHP programming language, Laravel framework, JavaScript, HTML markup language, personal computer with Intel Core i5 processor running Microsoft Windows operating system 10.

The goal of the work is a development of a software for managing the queue of users for the convenient execution of experiments at the RELDES remote laboratory.

Results. A queue module has been created for the RELDES remote laboratory, which can also be implemented in other remote laboratories based on the reuse methodology.

Conclusions. The software application has been developed that allows the user to queue for RELDES remote laboratory experiments and to monitor its status.

Field of application is remote laboratories.

ЗМІСТ

	С.
ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК	8
ВСТУП	9
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ	10
1.1 Огляд існуючих віддалених лабораторій	10
1.2 Огляд варіантів організації черг віддалених лабораторій	12
1.3 Особливості віддаленої лабораторії RELDES	16
1.4 Використання методології повторного використання для розробки віддаленої лабораторії	19
1.5 Постановка задач дипломної роботи	21
2 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	22
2.1 Призначення та аналіз вимог до програмного забезпечення	22
2.2 Розробка архітектури програмного забезпечення	23
2.3 Вибір інструментарію розробки	24
2.4 Висновки до розділу 2	31
3 КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	32
3.1 Організація спільного середовища розробки	32
3.2 Розгортання проєкту в середовищі віртуалізації	36
3.3. Алгоритм функціонування програми	38
3.4. Опис особливостей програмної реалізації	39
4 КЕРІВНИЦТВО ПРОГРАМІСТА	45
4.1 Призначення програми та умови виконання програми	45
4.2 Характеристика програми	45
4.3 Звертання до програми	45
4.4 Вхідні та вихідні дані	46
4.5 Додавання експерименту	46
5 КЕРІВНИЦТВО КОРИСТУВАЧА	48
5.1 Призначення програми	48
5.2 Умови виконання програми	48

5.3 Виконання програми.....	48
ВИСНОВКИ.....	50
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ	51
ДОДАТОК А Текст програми.....	55
ДОДАТОК Б Слайди презентації	69

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

IDE	– Integrated Development Environment;
HTML	– HyperText Markup Language;
LaaS	– Laboratory as a Service;
MVC	– Model-View-Controller;
RELDES	– Remote Laboratory for Design of Embedded Systems;
БД	– база даних;
ВЛ	– віддалена лабораторія;
ПЗ	– програмне забезпечення.

ВСТУП

На сьогоднішній день віддалені лабораторії (ВЛ) набувають дуже великої популярності через світову пандемію та тотальний перехід на дистанційне навчання, коли відсутня можливість роботи в реальних лабораторіях навчального закладу.

Треба відзначити, що важливою перевагою є те, що при неспроможності придбати коштовне обладнання для створення або оновлення власних лабораторій, навчальний заклад має чудову альтернативу використовувати ВЛ інших розробників по всьому світу.

Для коректної організації доступу до обладнання лабораторії необхідно передбачити множинний доступ, тому на стороні сервера повинна бути реалізована система черги, яка дозволила б користувачу без зайвих перешкод отримати доступ до обладнання. Під час очікування доступу до обладнання, будь-який користувач повинен мати можливість стежити за станом черги.

Серед великої різноманітності віддалених лабораторій слід зазначити розробку кафедри програмних засобів Національного університету «Запорізька політехніка» - віддалену лабораторію RELDES. Тут користувачі мають змогу виконувати експерименти на платах Arduino з різноманітними модулями, а також проводити дослідження на віртуальній та фізичній моделях розумного будинку. Лабораторія RELDES постійно удосконалюється та модернізується, наразі потребує удосконалення підсистема управління чергою на виконання експериментів, тому тема роботи є актуальною.

Мета роботи – розробка програмного забезпечення управління чергою користувачів для зручного виконання експериментів віддаленої лабораторії RELDES.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд існуючих віддалених лабораторій

Мережа Інтернет розвивається з досить великою швидкістю. Завдяки цьому зв'язок між віддаленими один від одного комп'ютерними пристроями став більш надійним, швидкісним та дешевшим. Саме це дозволило реалізувати ідею створення віддаленого керування обладнанням реальної лабораторії.

На сьогоднішній день віддалені лабораторії набувають дуже великої популярності. Вони представляють собою програмно-апаратний комплекс, який дозволяє проводити певні експерименти без безпосереднього контакту з реальним обладнанням. Викладач або студент за наявності комп'ютера з доступом до мережі Інтернет може з'єднатися з лабораторією перебуваючи будь-якій точці земної кулі та продемонструвати або провести необхідні експерименти для виконання лабораторних або практичних робіт.

WebLab-Deusto – ВЛ, розробка інженерного факультету іспанського Університету Деусто. Вона спрямована на розширення експериментального навчання за рахунок використання і розвитку віддалених лабораторій. WebLab-Deusto використовується університетами зокрема інженерними факультетами та середніми школами з різних країн світу, а також у багатьох дослідницьких проєктах [1]. Головну сторінку вебсайту лабораторії представлено на рисунку 1.1.

Electromeeet Remote Labs – розробка Інженерного Технологічного Інституту (Австралія). Вона дозволяє студентам отримати доступ до широкого спектру інженерного програмного забезпечення та підключеного обладнання в режимі реального часу [2]. Сторінку з доступними лабораторіями зображено на рисунку 1.2.

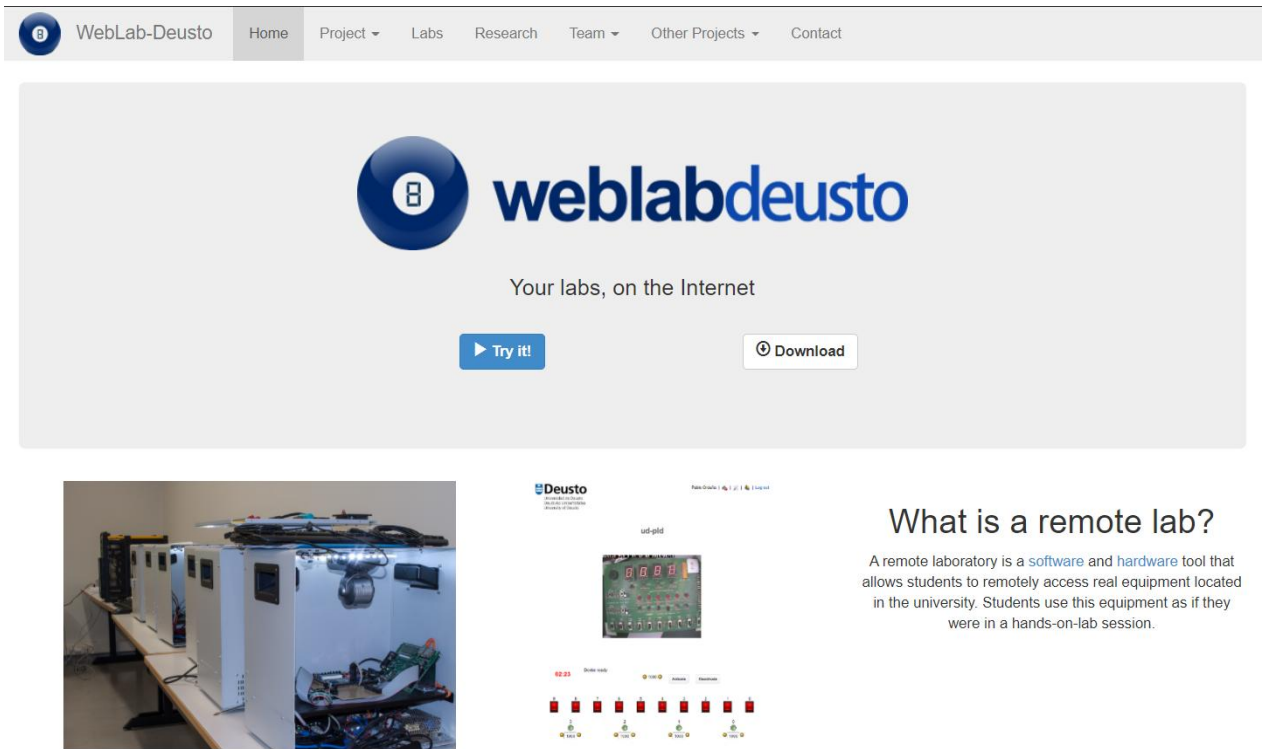


Рисунок 1.1 – Головна сторінка вебсайту ВЛ WebLab-Deusto

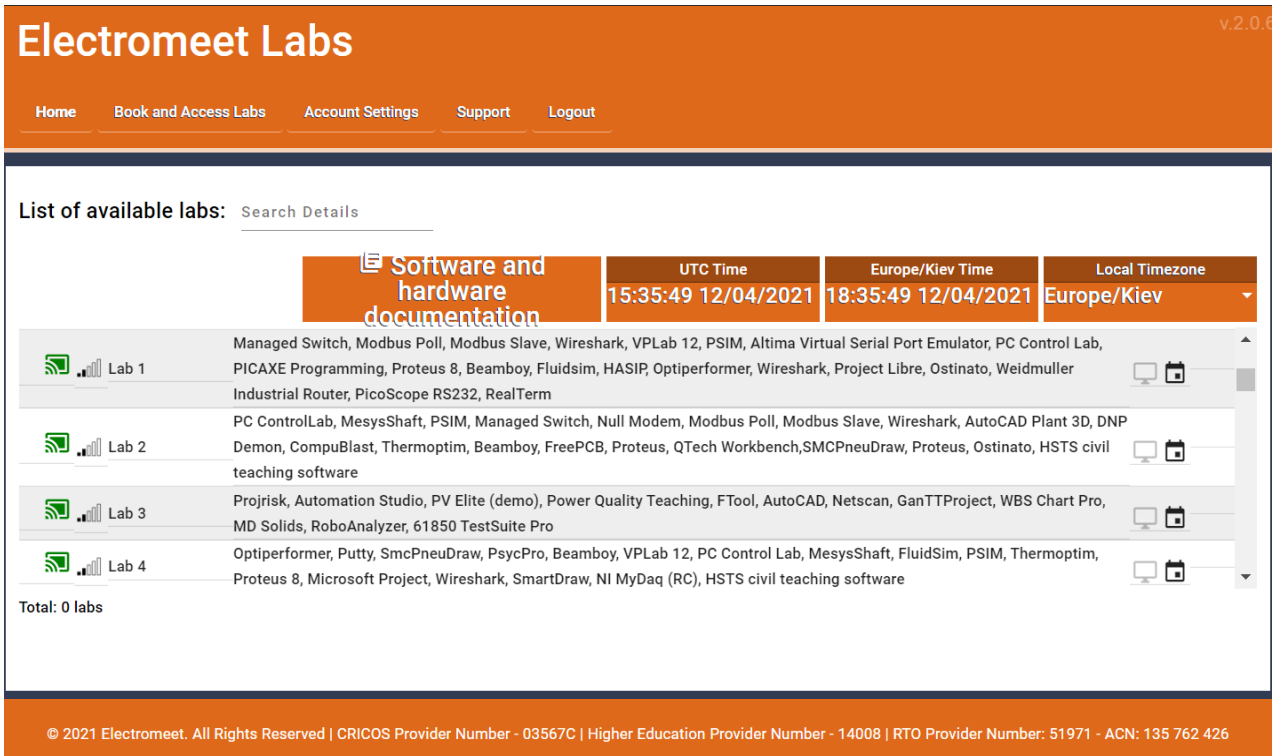


Рисунок 1.2 – Сторінка з доступними лабораторіями на вебсайті ВЛ Electromeet Remote Labs

GOLDI (Grid of Online Lab Devices Ilmenau) - гібридна лабораторія, розроблена на кафедрі інтегрованих систем зв'язку Технологічного університету Ільменау (Німеччина). Вона надає набір інструментів, що підтримує всі етапи розв'язку складних завдань керування (наприклад, у галузях машинобудування, робототехніки, телекомунікацій) [3]. Головну сторінку вебсайту лабораторії зображено на рисунку 1.3.

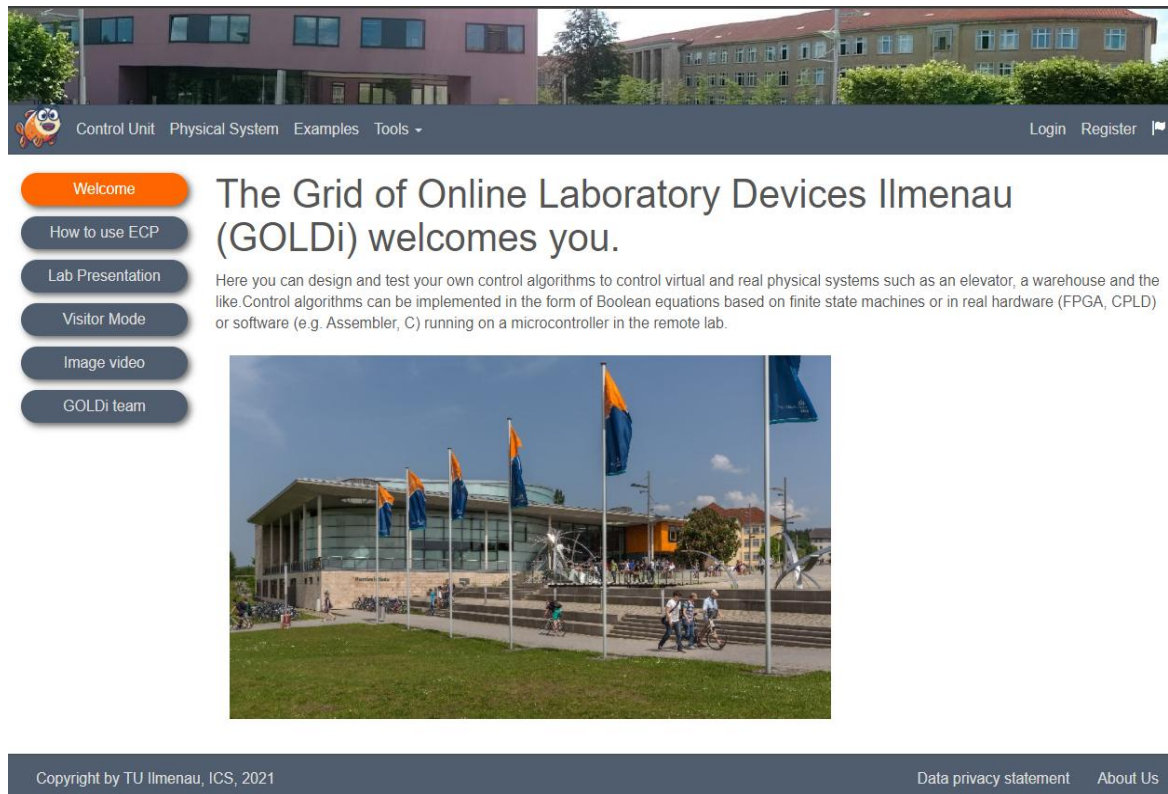


Рисунок 1.3 – Головна сторінка вебсайту гібридної лабораторії GOLDI

Таким чином, ВЛ дуже популярні на сьогоднішній день через пандемію, тому це напрямок, який має розвиватися далі.

1.2 Огляд варіантів організації черг віддалених лабораторій

У ВЛ WebLab-Deusto адміністрування доступу до експериментів відбувається на основі черги пріоритетів. Платформа автоматично показує

студентам доступність і позицію в черзі (рис. 1.4), а також дає контроль над експериментом, коли приходить черга студента [4].



Рисунок 1.4 – Позиція в черзі на експеримент ВЛ WebLab-Deusto

У ВЛ Electromee Remote Labs для виконання експериментів користувачу спочатку необхідно зарезервувати час користування обладнанням. Користувачу надається зручна форма резервування (рис. 1.5) де він може обрати дату та час виконання експерименту. Мінімальний час резервування 1 година, за один раз користувач може зарезервувати обладнання на 4 години.

Lab Schedule - UK-Lab 3
 Book your time clicking at empty timeslots

UTC Time

08:01:22 19/03/2021

Europe/Kiev Time

10:01:22 19/03/2021

Local Timezone

Europe/Kiev ▼

<

>

today

Mar 14 – 20, 2021

	Sun 3/14	Mon 3/15	Tue 3/16	Wed 3/17	Thu 3/18	Fri 3/19	Sat 3/20
12am				12:00 - Reserved Lab	12:00 - Reserved Lab		
1am				1:00 - Reserved Lab	1:00 - Reserved Lab		
2am				2:00 - Reserved Lab	2:00 - Reserved Lab		
3am					3:00 - Reserved Lab		
4am							
5am							
6am							
7am							
8am							
9am							
10am			10:00 - Reserved Lab				
11am			11:00 - Reserved Lab			11:00 - Reserved Lab	

Рисунок 1.5 – Форма для резервування експериментів ВЛ Electromeeet Remote Labs

У лабораторії GOLDI існує три різні варіанти запуску експериментів.

Механізм прямого запуску. Користувач може почати експеримент одразу. Цей варіант можливий тільки в тому випадку, якщо обрані пристрої ніким не використовуються. Час експерименту має обмеження на 30 хвилин, але якщо жоден інший користувач не очікує апаратного забезпечення він буде автоматично збільшуватися.

Механізм запуску на основі черги. Запуск на основі черги забезпечує додаткові функціональні можливості в порівнянні з прямим запуском. Після вибору експериментів користувача буде додано до спеціальної системи черги (рис. 1.6), яка динамічно змінює пріоритети користувачів в залежності від поточної завантаженості лабораторії. Він завжди намагається отримати максимально можливе навантаження.

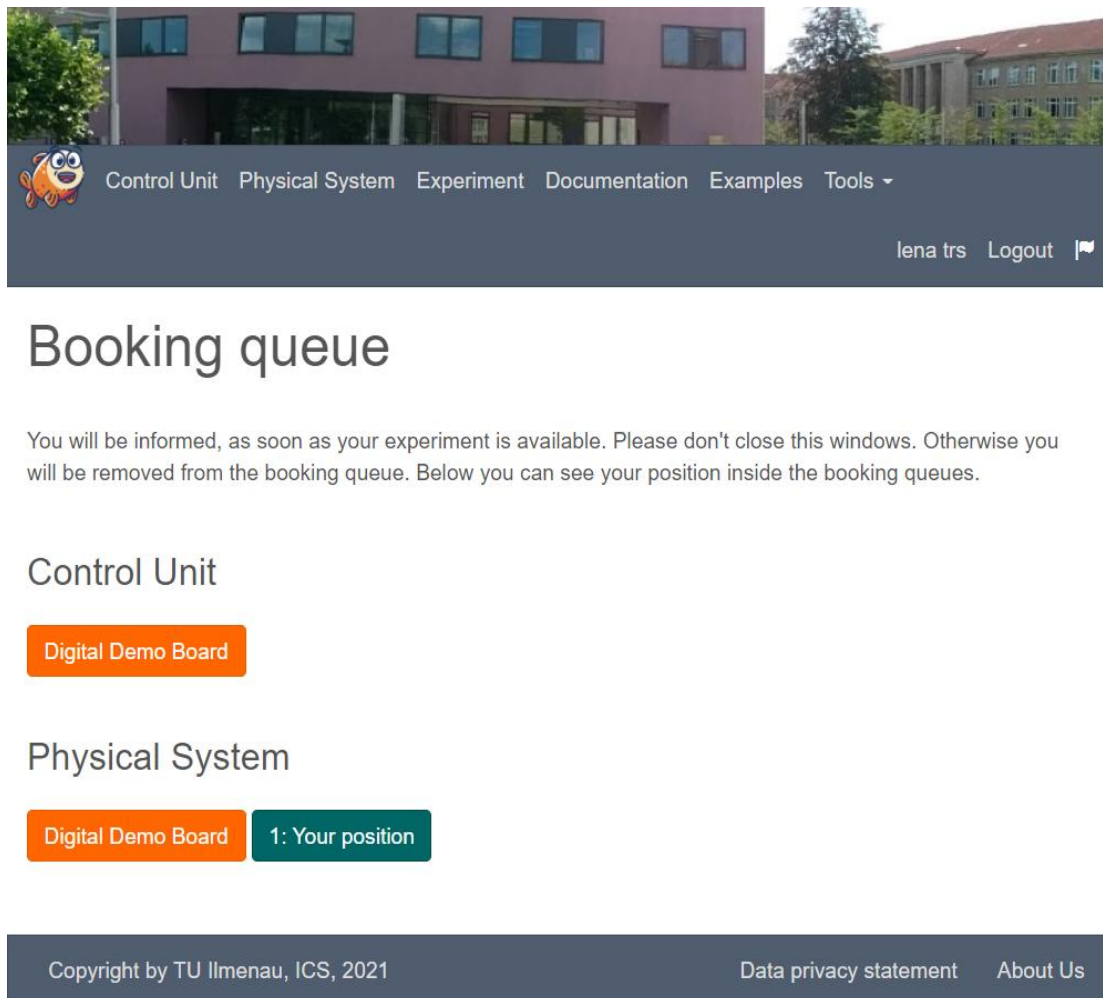


Рисунок 1.6 – Черга на виконання експериментів гібридної лабораторії
GOLDI

Попередньо зарезервований механізм запуску. У цьому випадку користувач може заздалегідь зарезервувати конкретний експеримент на певний часовий інтервал (рис. 1.7). Тоді обране обладнання буде заблоковано протягом усього зарезервованого часу. Крім того, викладачі можуть скористатися цією функцією, щоб отримати ексклюзивне резервування для групи студентів для спільної роботи над попередньо налаштованим експериментом [5].

Як показали дослідження існує два типи організації черг на основі попереднього резервування та в режимі онлайн. Таким чином, організація черги це важлива задача, яка дозволяє організовувати доступ користувачів та задати дисципліну проведення експериментів.

Control Unit Physical System Experiment Documentation Examples Tools ▾ Elena Teresh Logout

45
Duration (minutes) 04/15/2021 Reserve Delete Help My reser...

Control Unit: Digital Demo Board [Real]

			New reservation		
	13:00	14:00	15:00	16:00	17:00
Thu 15 April					

Physical System: Digital Demo Board [Virtual]

Copyright by TU Ilmenau, ICS, 2021 Data privacy statement About Us

Рисунок 1.7 – Резервування експерименту гібридної лабораторії GOLDI

1.3 Особливості віддаленої лабораторії RELDES

Віддалена лабораторія RELDES є розробкою кафедри програмних засобів Національного Університету «Запорізька Політехніка». Вона надає віддалений доступ до експериментального обладнання на основі платформ Arduino[6].

RELDES - це зручне середовище для тестування і швидкого створення прототипів вбудованих систем на основі багаторазово використовуваних апаратних і програмних компонентів. На сьогоднішній день реалізовано 4 експерименти на платах Arduino та керуючого обладнання - виконавчі пристрої, ультразвуковий датчик відстані, світлофор і рідкокристалічний дисплей. Всі специфікації, схеми експериментів і програмний код (шаблони для використання експериментів) доступні на сайті ВЛ [7].

Схема архітектури лабораторії представлена на рисунку 1.8. Для доступу до експериментів користувачеві необхідно зареєструватися. Доступ до сторінки експериментів закритий для незареєстрованих користувачів [8].

Для реєстрації в системі реалізована звичайна процедура заповнення стандартних полів: Ім'я користувача, Пароль, Електронна пошта, Ім'я (рис. 1.9). У разі успішного входу користувач отримує електронний лист з посиланням для завершення процесу реєстрації. Після підтвердження користувач зможе використовувати API RELDES зі своїм індивідуальним ключем API [7].



Рисунок 1.8 – Архітектура віддаленої лабораторії RELDES

RELDES

Arduino Arduino IDE Raspberry PI OpenHAB REIoT Розумний будинок та Інтернет речей Українська Вхід

Реєстрація

Заповніть наступні поля для реєстрації

ім'я користувача

Необхідно заповнити «Username».

Email

пароль

Необхідно заповнити «Password».

Реєстрація

© SWED 2021

Рисунок 1.9 – Форма реєстрації у віддаленій лабораторії RELDES

Користувачі можуть завантажити свій програмний код в мікроконтролер на платі Arduino через вебклієнт. За допомогою Web-інтерфейсу користувач може створити власний код або завантажити файл з вихідним кодом. Сервер лабораторії компілює отриманий вихідний код. Після компіляції згенерований код завантажується в мікроконтролер на платі Arduino. Якщо в програмі є помилки, сервер генерує і повертає результати компіляції. Таким чином, у користувача є можливість створювати і редагувати свій програмний код прямо в вебінтерфейсі клієнта без необхідності в інструментах розробки на персональному комп'ютері користувача[8].

На рисунку 1.10 зображено вебінтерфейс сторінки експерименту. Вкладка «Експерименти» на верхній панелі меню дозволяє швидко перемикатися між експериментами. Головна частина поділена на дві області. У правій області знаходиться редактор коду, а в лівій – відображено відео експерименту. Нижче розташовано поле для завантаження файлу з кодом та час відведений на експеримент.

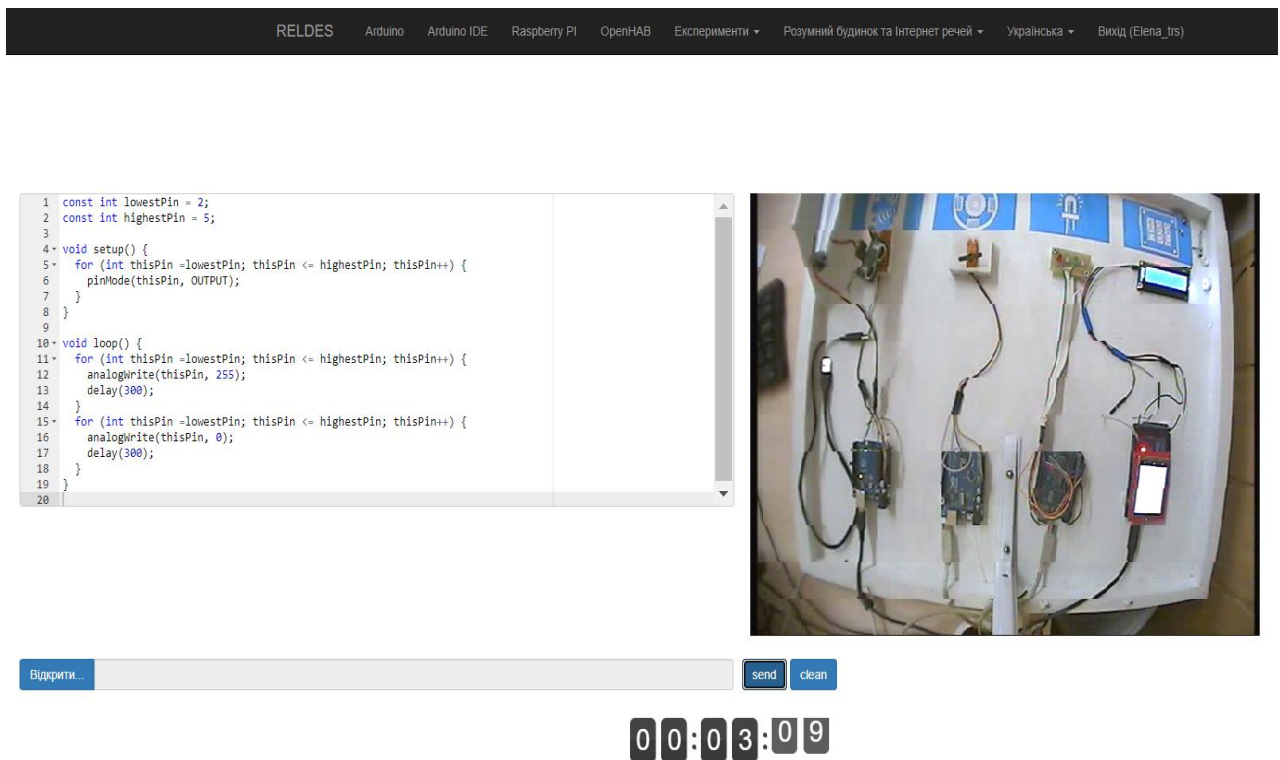


Рисунок 1.10 – Вебінтерфейс експерименту віддаленої лабораторії RELDES

В процесі модернізації лабораторії виникла така проблема, що механізм роботи черги реалізовано, але відсутня її візуалізація, тому тема роботи є актуальною.

1.4 Використання методології повторного використання для розробки віддаленої лабораторії

Створення ВЛ з нуля дуже складний та кропіткий процес, який потребує великих фінансових та часових витрат. На сьогоднішній день методологія повторного використання набуває великої популярності у багатьох сферах нашого оточення. Рішення, що придатні для повторного використання, дають можливість тиражувати і впроваджувати окремі частини системи, не витрачаючи час на їх розвиток. Існує багато успішних проєктів, що дуже швидко розвиваються завдяки повторному використанню існуючого програмного забезпечення, одним із яскравих прикладів є лабораторія WebLab-Deusto.

Розробка лабораторії, яка складатиметься з незалежних модулів сприятиме повторному використанню окремих компонентів та значно спростить її створення [7]. Існує парадигма *Laboratory as a Service*, яка використовується для розробки та впровадження модульних ВЛ з високим рівнем абстракції та віртуалізації. Вона ґрунтується на модульній концепції ВЛ та передбачає надання всіх лабораторних функцій та компонентів у «файлі опису служби» як набір абстрактних послуг. Метою модульних ВЛ є перетворення лабораторій на модульні компоненти з метою полегшення технічного обслуговування, повторного використання та взаємозамінності компонентів програмно [9].

Також існує парадигма *Smart Device*, вона переглядає традиційну архітектуру клієнт-сервер, на яку покладаються багато реалізацій ВЛ. Основними відмінностями між існуючими реалізаціями та смарт-пристроями - це, по-перше, повне роз'єднання між сервером та клієнтом, а по-друге, представлення сервера як набір чітко визначених служб та функціональних можливостей, що забезпечують взаємодію.

Парадигма *Smart Device* визначає одну загальну специфікацію, спільну для всіх смарт-пристроїв. Повторне використання загальної специфікації та роз'єднання клієнт-сервер полегшує більшість проблем, з якими стикаються розробники, коли клієнтська програма повинна бути адаптована до нових операційних систем або платформ, чи якщо клієнтська програма повинна бути інтегрована в інші середовища, або просто якщо додаткові функції додані до сервера. Крім того, взаємодія з існуючими програмами та послугами й їх повторне використання стає можливим, коли лабораторії мають спільну специфікацію.

Специфікація *Smart Device* визначає зв'язок та інтерфейси між клієнтом та сервером, і надає достатню інформацію для генерації клієнтських застосунків або повторного використання існуючих клієнтських програм. Оскільки специфікація є загальною для багатьох смарт-пристроїв, клієнтські

програми не є щільно пов'язаними з одним сервером, що заохочує повторне використання [10].

Отже, в останні роки стає актуальним питання про новий підхід до розвитку ВЛ, який спирається на методологію повторного використання. Така методологія сприяє швидкій розробці ВЛ, значно полегшує подальшу підтримку лабораторії. Вона зменшує кількість проблем при переносі системи на інші платформи або при розширенні функціоналу системи.

1.5 Постановка задач дипломної роботи

На сьогоднішній день ВЛ стають дуже популярними, але їх розробка це дуже складний і кропіткий процес. Одним із складних компонентів в реалізації є черга, для її організації існує два підходи на основі попереднього резервування та в режимі онлайн.

Методологія повторного використання значно полегшує розробку, підтримку та подальше розширення функціоналу ВЛ, якщо необхідний компонент є у вільному доступі.

Тому, створений для ВЛ RELDES модуль черги може бути використаний розробниками інших ВЛ.

Мета роботи – розробка програмного забезпечення управління чергою користувачів для зручного виконання експериментів віддаленої лабораторії RELDES.

Для досягнення поставленої мети потрібно вирішити наступні задачі:

- виконати аналіз вимог та розробку архітектури програмного забезпечення;
- обрати інструментарій розробки програми;
- організувати спільне середовище розробки та середовище віртуалізації;
- виконати програмну реалізацію програмного застосунку;
- розробити програмну документацію.

2 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Призначення та аналіз вимог до програмного забезпечення

Проведення експериментів у лабораторії RELDES передбачає роботу з реальним обладнанням, одже декілька користувачів не можуть виконувати один і той же експеримент одночасно. Тому, для коректної організації доступу до обладнання лабораторії потрібно розробити програмне забезпечення, яке дозволить студентам та викладачам раціонально використовувати час, відведений на виконання експериментів та підвищити ефективність своєї роботи.

Функціональні та нефункціональні вимоги представлено на рис. 2.1. Розробка ПЗ з дотриманням цих вимог сприятиме ефективній модернізації ВЛ.

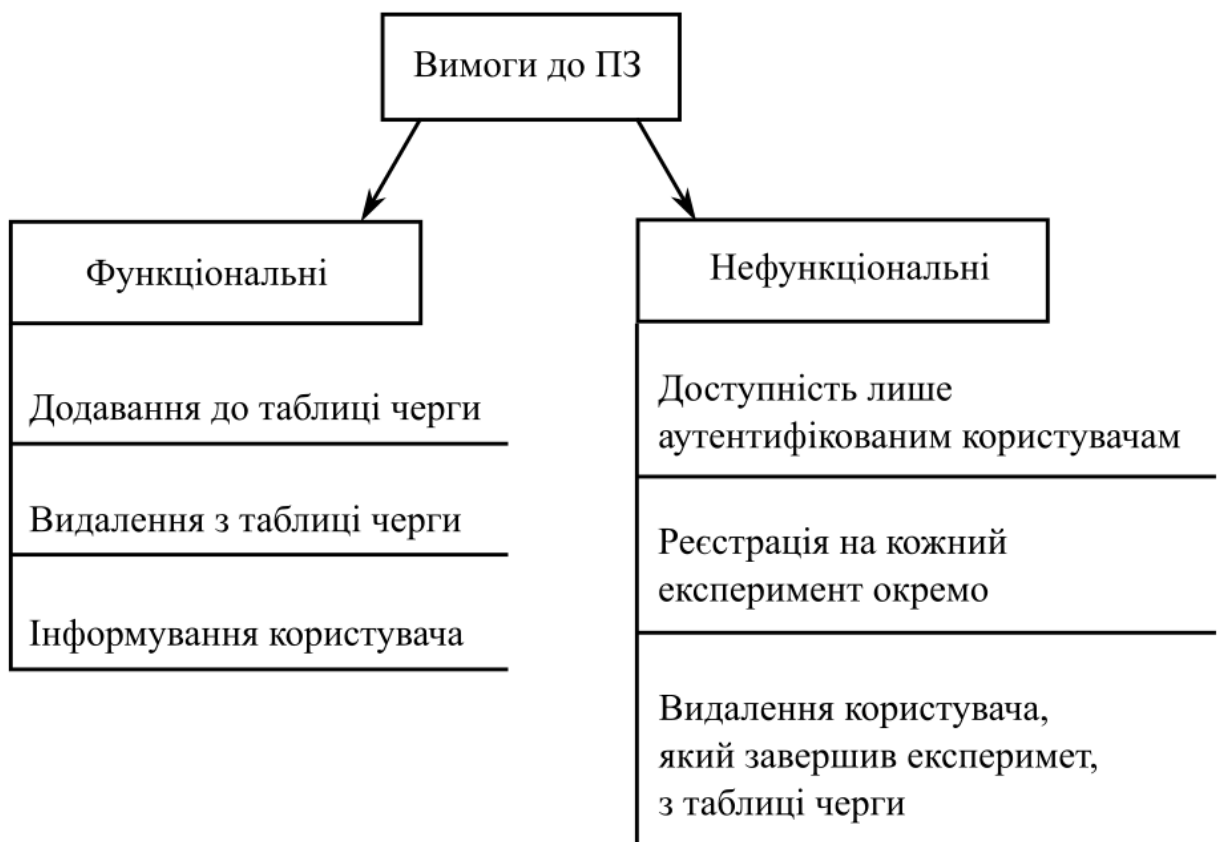


Рисунок 2.1 – Вимоги до програмного забезпечення

Для організації коректної роботи користувачів ПЗ повинне мати функціонал додавання до черги. Оскільки лабораторія RELDES передбачає реєстрацію користувачів на сайті, то ставати до черги можуть лише аутентифіковані користувачі.

Користувачі, які завершили експеримент, повинні звільняти чергу, тому ПЗ повинне мати функціонал видалення користувача з таблиці черги.

ВЛ RELDES має чотри експерименти, отже на кожний експеримент повинна бути передбачена окрема реєстрація.

Користувач повинен мати можливість стежити за станом черги, тому до ПЗ необхідно додати функціонал інформування користувача.

Таким чином, в процесі розробки необхідно:

- на сторінці з експериментом лабораторії RELDES додати кнопку реєстрації на експеримент для користувачів, які ввійшли в свій обліковий запис. Для користувача який вже зареєстрований на експеримент, кнопка має бути прихована, як тільки участь користувача в черзі закінчиться вона повинна з'явитися. При реєстрації користувача на експеримент слід додавати відповідний запис у таблицю бази даних;
- розробити задачу для черги, яка буде проходитися по таблиці кожного експерименту з чергою і виставляти статус активного експерименту, а також буде видаляти тих людей у яких закінчився час виконання експерименту. Задача повинна виводити в консоль який користувач закінчив такий експеримент.

2.2 Розробка архітектури програмного забезпечення

Виходячи з описаних вище вимог, розроблюване ПЗ повинне мати найбільш простий та інтуїтивно зрозумілий інтерфейсу вигляді кнопки та панелі з інформацією.

Оскільки кожен експеримент повинен мати окрему чергу, було прийняте рішення використовувати Redis.

Redis - це сховище структур даних в пам'яті, що використовується в якості бази даних. Завдяки структурам даних, які знаходяться в Redis можна без перешкод реалізувати черги за принципом FIFO (First In – First Out) [11].

На рисунку 2.2 зображено архітектуру ПЗ. Після того як користувач стане у чергу, робиться відповідний запис у таблицю БД, а вже потім для більш зручного доступу до даних та роботи з ними додаються до черг Redis (для кожного експерименту своя черга).

Завдяки використанню Redis реалізація черги стає більш зручною та швидкою.

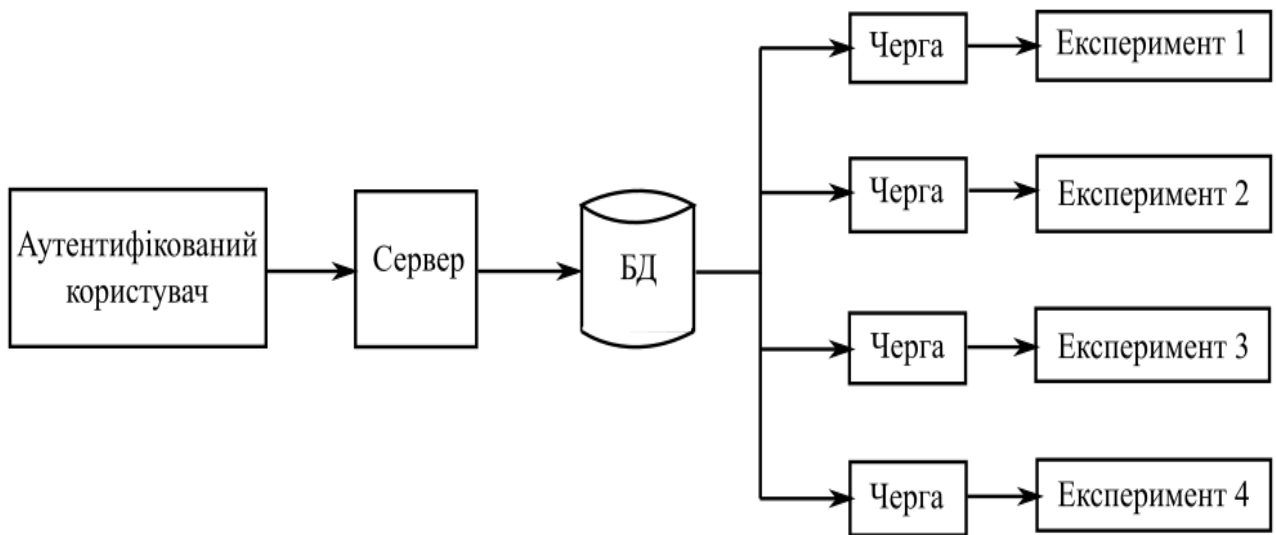


Рисунок 2.2 – Архітектура системи

2.3 Вибір інструментарію розробки

2.3.1 Вибір мови програмування

На сьогоднішній день існує досить велика кількість мов для розробки вебзастосунків. Далі буде розглянуто 3 найпопулярніші мови.

PHP - одна з найпопулярніших скриптових мов програмування, яка інтенсивно використовується для розробки вебпроектів [12]. Вона має велике співтовариство та безліч готових рішень. Завдяки динамічній типізації та

стандартним бібліотечним функціям для роботи з вебсайтами забезпечує високу швидкість написання проєктів.

Ruby – динамічна, високорівнева мова програмування, що інтерпретується. Завдяки фреймворку Ruby on Rails здобув широку популярність серед веброзробників. Основними принципами Ruby on Rails є максимальне використання механізмів повторного використання, що дозволяють мінімізувати дублювання коду в застосунках [13].

Python - одина з найпопулярніших і простих для вивчення мов програмування, що має досить широкий спектр застосування. Свою популярність у розробці вебзастосунків здобула завдяки таким фреймворкам, як Django, Pyramid, TurboGears та інші [14].

У таблиці 2.1 подано порівняльну характеристику мов розробки.

Таблиця 2.1 – Порівняння мов розробки

Критерій порівняння	Ruby	Python	PHP
Велика кількість фреймворків для веброзробок	-	+	+
Відсутність необхідності підключення бібліотек	-	-	+
Розширена підтримка баз даних	-	+	+
Об'єктно-орієнтований підхід	+	+	+
Висока продуктивність	+	-	+

Оскільки лабораторія RELDES ревізована на мові PHP, було прийнято рішення обрати саме цю мову для виконання дипломного проєкту.

Розробка мови PHP належить Расмусу Лердорфу, перша версія якої вийшла у 1995 році. Мова, створена ним, зуміла здобути популярність у всьому світі. Найголовнішою її задачею є забезпечення ефективного зв'язку вебресурсу з сервером та базами даних. PHP дуже гарно поєднується з

HTML-кодом, що значно полегшує розробку вебзастосунків. За своїм синтаксичним наповненням PHP схожа на мови Cі та Perl. Почати роботу можна з оператора PHP без опису змінних. Щоб мова виконувала код, його потрібно помістити всередину обмежуючих тегів `<? php?>`. Імена змінних, які укладають в подвійні лапки, містять на початку знак `$`. Процесор PHP обробляє HTML-файли так, що неможливо визначити це звичайний HTML-файл або його отримано в результаті виконання скрипта [15].

До суттєвих переваг мови можна віднести:

- високу швидкість роботи і, відповідно, загальну продуктивність ресурсів;
- простий синтаксис, і внаслідок цього простоту освоєння;
- відмінну сумісність і переносимість - php-коди працюють однаково добре з різними платформами;
- високу гнучкість, ємність і функціональність - PHP-програму можна складати окремо від розробки вебсторінки, після чого поєднати;
- багатозадачність і широкі можливості - створення будь-яких вебзастосунків, блогів, гостьових книг, інтернет-магазинів, сайтів, робота з заголовками, pdf-документами, базами даних, електронною поштою тощо;
- набір тексту коду і його редагування можна здійснювати в будь-якому текстовому або HTML-редакторі [16].

PHP має безліч можливостей. Серед них управління зображеннями за допомогою бібліотеки GD, створення віддалених підключень за допомогою cURL, обробка файлової системи, управління сесіями, призначеними для користувача, використання класу SimpleXML, який читає і записує файли XML через об'єкти, підтримка файлів cookie і сесій, оболонка виконання з командного рядка, стиснення ZIP-архівів, використання Ajax [17].

2.3.2 Вибір фреймворку

За допомогою нативної мови PHP можна створити повноцінний вебзастосунок, але на сьогоднішній день все більше веброзробників використовують PHP-фреймворки. Існує декілька причин використання фреймворків:

- фреймворки значно прискорюють розробку. Наприклад, PHP-фреймворк позбавляє від необхідності писати запити до баз даних. У ньому реалізовані базові функції CRUD, які необхідні для роботи з базами даних;
- написані на фреймворках застосунки легко масштабуються;
- проєкт, розроблений на фреймворку, підтримувати легше, ніж на нативном PHP. Код фреймворків лаконічний, тому з ним дуже просто працювати;
- в PHP-фреймворках використовується модель MVC, яка значно а спрощує розробку;
- застосунки на фреймворками краще захищені, ніж застосунки на чистому PHP;
- у фреймворках реалізований принцип DRY. Це дозволяє розробникам писати менше коду[18].

На сьогоднішній день існує декілька шаблонів проєктування програмного забезпечення: Presentation–abstraction–control (PAC), MVC, Naked objects та ін. Серед них найпопулярнішим є MVC, тому було обрано фреймворки, які використовують саме на цей шаблон.

Далі будуть розглянуті три найпопулярніші PHP-фреймворки, на яких було реалізовано ВЛ RELDES. Перша версія лабораторії RELDES була реалізована на мові PHP з використанням фреймворку Codeigniter.

Codeigniter – фреймворк для розробки вебзастосунків, набір інструментів для створення вебсайтів з використанням PHP. Його мета полягає в тому, щоб значно прискорити розробку проєктів. Фреймворк надає

широкий набір бібліотек для вирішення часто затребуваних завдань, а також простий інтерфейс і логічну структуру доступу до цих бібліотек. CodeIgniter дозволяє зосереджуватися на проєктах, мінімізуючи кількість коду, який необхідно написати для вирішення кожного конкретного завдання [19].

Пізніше у реалізації лабораторії використовувався фреймворк Yii.

Yii – високоефективний фреймворк, заснований на компонентній структурі PHP. Він дозволяє максимально застосувати концепцію повторного використання коду і може істотно прискорити процес веброзробки. Yii фреймворк для вебпрограмування загального призначення, який може бути використаний для розробки практично будь-яких вебзастосунків. Завдяки своїй легковажності і наявності просунутих засобів кешування, Yii особливо підходить для розробки застосунків з великим потоком трафіку, таких як портали, форуми, системи управління контентом, системи електронної комерції тощо [20].

На сьогоднішній день використовується фреймворк Laravel, базові функції якого дозволяють суттєво прискорити процес розробки.

Laravel - гнучкий PHP-фреймворк з відкритим вихідним кодом, який працює швидше, безпечніше і дешевше, ніж багато інших подібних фреймворків. Крім того, Laravel не вимагає використання плагінів для введення функціоналу на сайт, але дає можливість безпосередньо кодувати ці параметри у вихідному коді. Це забезпечує простоту і велику ефективність при реалізації всього проєкту [21].

У таблиці 2.2 подано порівняльну характеристику PHP-фреймворків.

Оскільки на даний момент лабораторія RELDES побудована на фреймворку Laravel, було прийнято рішення використовувати саме його.

Laravel – один із найпопулярніших та безкоштовних PHP фреймворків з відкритим кодом для розробки вебзастосунків. Його розробив Тейлор Отвелл як спробу представити більш просунуту альтернативу фреймворку CodeIgniter. Перша бета-версія Laravel стала доступною у червні 2011, а вже через місяць було випущено Laravel 1.

Таблиця 2.2 – Порівняння PHP-фреймворків

Критерій порівняння	Codeigniter	Yii	Laravel
Легкий у вивченні	+	-	+
Постійне оновлення	-	+	+
Швидка інтеграція з великою кількістю додаткових бібліотек	-	+	+
MVC-архітектура вебзастосунків	+	+	+
Вбудовані механізми для аутентифікації, авторизації, реєстрації користувачів	-	+	+

На сьогоднішній день актуальною стабільною версією є Laravel 8. Серед головних особливостей фреймворку можна виділити наступні:

- інформативна документація – документація Laravel дуже добре структурована, кожній інструкції та процесу присвячена окрема стаття, де розповідаються принципи їх роботи та демонструються приклади використання;
- міграції – контроль версій для структури таблиць БД. У кожному файлі міграції міститься або сама структура таблиць, або зміни їх структури. Завдяки великій кількості методів управління міграціями, з ними можна проводити певні маніпуляції: відкат окремих міграцій, скидання всіх та інші;
- валідатори – спеціальні конструкції, що дозволяють виконувати перевірку даних на основі різноманітних готових правил. Також Laravel дозволяє створювати власні правила та повідомлення про помилки;
- шаблонізатор Blade - власний шаблонізатор з набором своїх директив;
- можливість розширення базового функціоналу – у фреймворку Laravel існує можливість розширення базового функціоналу шляхом встановлення додаткових пакетів [22].

2.3.3 Вибір середовища розробки

Мова PHP передбачає можливість розробки в будь-якому текстовому редакторі, але для великих проєктів доцільно використовувати IDE. На сьогоднішній день існує безліз IDE для розробки PHP-проєктів серед них NetBeans, phpDesigner, PHPStorm, Eclipse PDT та ін.

Для виконання дипломної роботи було обрано PHPStorm (рис. 2.3). IDE було розроблено компанією JetBrains. У середовищі є багато корисних функцій, які відмінно підходить для роботи з PHP, HTML і JS [23]. PhpStorm має три найголовніші переваги.

Розумний редактор PHP-коду. IDE забезпечує розумне автодоповнення, підсвічування синтаксису, розширені налаштування форматування, перевірку помилок на льоту, згортання фрагментів коду, підтримку мовних комбінацій і багато інших можливостей для продуктивної розробки. Автоматичний рефакторинг відповідає за порядок у коді і допомагає легко і безпечно виконувати глобальні налаштування проєкту.

Аналіз якості коду. Сотні вбудованих інспекцій перевіряють код прямо в режимі редагування, а також дозволяють проаналізувати весь проєкт на наявність помилок і проблем в структурі коду. Для більшості інспекцій доступні швидкі виправлення, які дозволяють відкоригувати код миттєво.

Зручна навігація і швидкий пошук. PhpStorm допомагає швидко переміщатися по коду й економити час при роботі над великими проєктами. В один клік можна перейти до визначення будь-якого методу, функції, змінної або знайти їх використання в проєкті [24].

PhpStorm платне середовище, але має 30-денний тестовий період. Також компанія JetBrains надає безкоштовну ліцензію для студентів і викладачів. Воно дуже зручне у використанні та має безліч переваг, тому було обрано саме його.

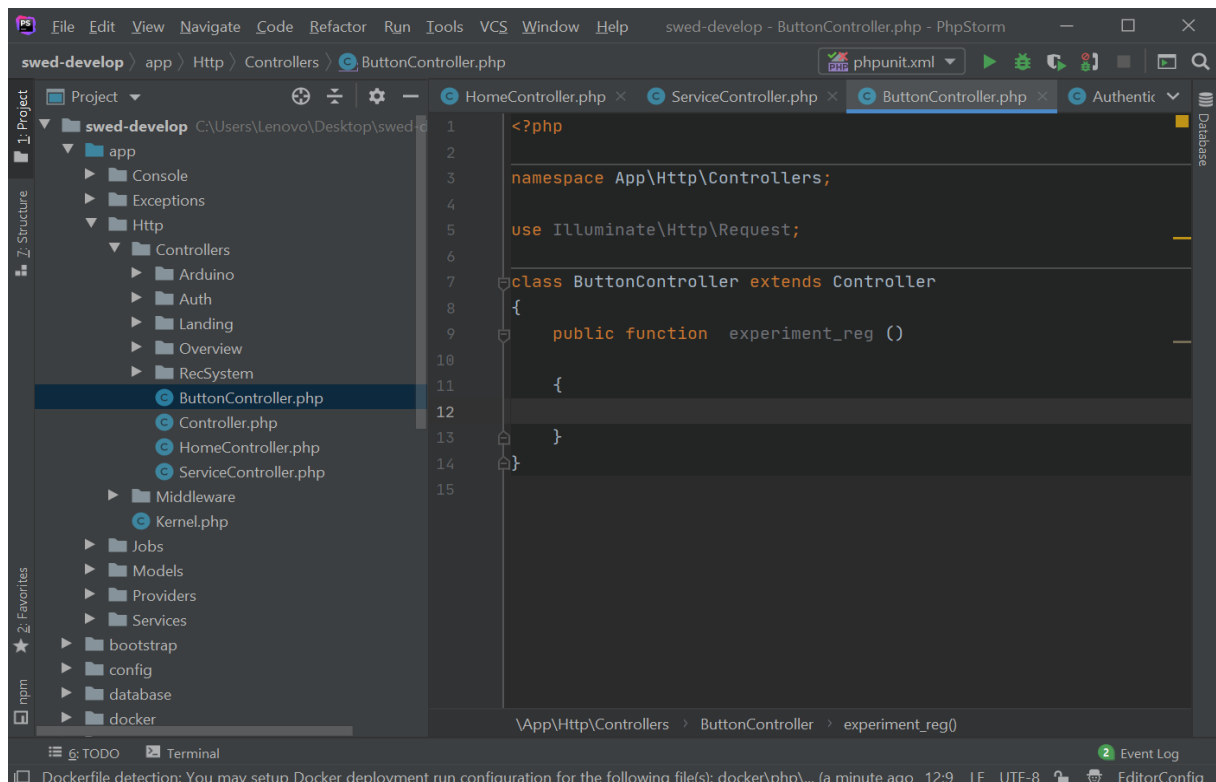


Рисунок 2.3 – Інтерфейс IDE PHPStorm

2.4 Висновки до розділу 2

Проаналізувавши призначення та вимоги до ПЗ, було розроблено його архітектуру. Для програмної реалізації було проаналізовано мови розробки вебзастосунків та обрано мову PHP з використанням фреймворку Laravel. Через зручність у використанні та значні переваги в якості середовища розробки було обрано PhpStorm.

3 КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Організація спільного середовища розробки

Як відомо, для розробки проєктів командою є сенс використовувати системи управління програмними проєктами. До найбільш популярних на сьогоднішній день відносять Jira, Asana, Trello та ін.

Jira - платформа для управління проєктами, завданнями і відстеження помилок (рис. 3.1). Вона заснована на методології Agile. Jira доступна в вебверсії та у вигляді десктопного застосунку. Платформа Jira являє собою гнучкий інструмент, в якому компанії можуть адаптувати і змінювати під свої потреби функціональний зовнішній вигляд сервісу. До кожного проєкту можна визначити права доступу для учасників. Структура Jira складається з трьох елементів: проєкт, завдання і підзадачі. Проєкт - основний елемент платформи, в якому зберігаються завдання і інформація по роботі над програмою. Завдання Jira - це структуровані інструменти для управління проєктом [25].

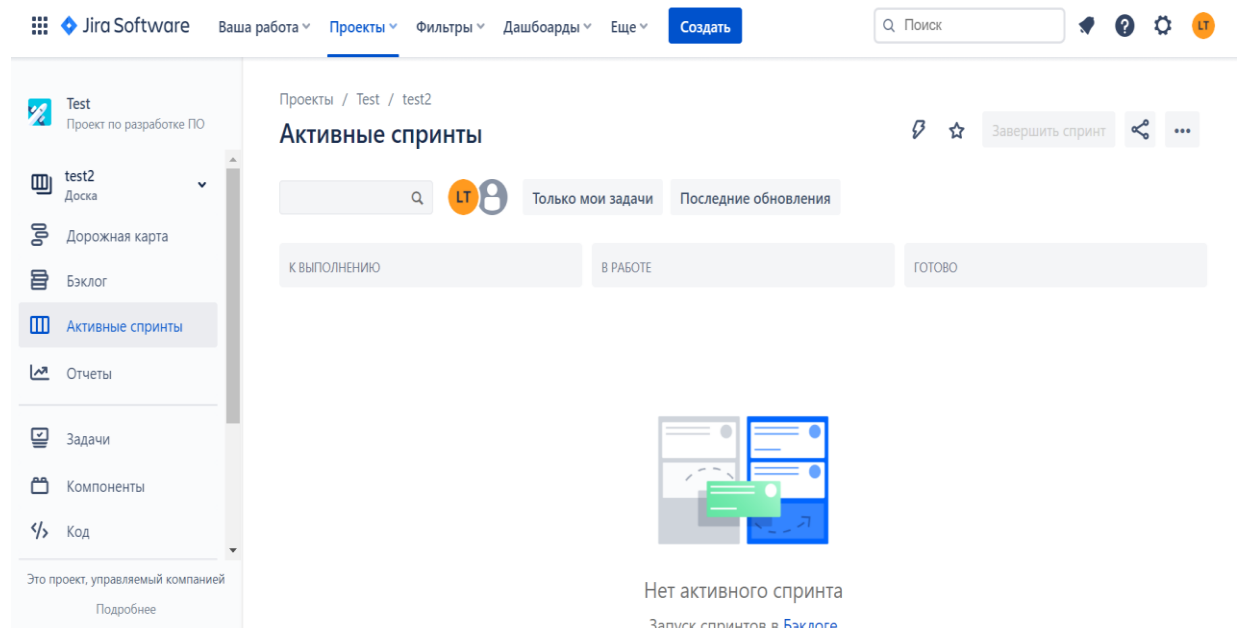


Рисунок 3.1 – Інтерфейс платформи для управління проєктами Jira

Asana - рішення для спільного управління проектами і завданнями (рис. 3.2), яке дозволяє командам обмінюватися даними, планувати, організовувати і відслідковувати процес виконання завдань, над яким працюють члени команди. Вона була створена для планування і координації завдань, розподілу їх на окремі задачі і призначення їх членам команди. Це рішення дозволяє адмініструвати всі завдання, створюючи окрему екосистему для кожного проекту. Розподіл великого проекту на окремі завдання, а їх на завдання, дозволяє кожному працівнику чітко відповідати за свою зону відповідальності. Користувач створює свій особистий простір, в якому він оглядає всі свої завдання і проекти. Таким чином робота виконується чітко, злагоджено і без затримок. А завдяки вбудованій системі зв'язку члени команди можуть легко обмінюватися інформацією, даними та повідомляти про статус роботи. [26].

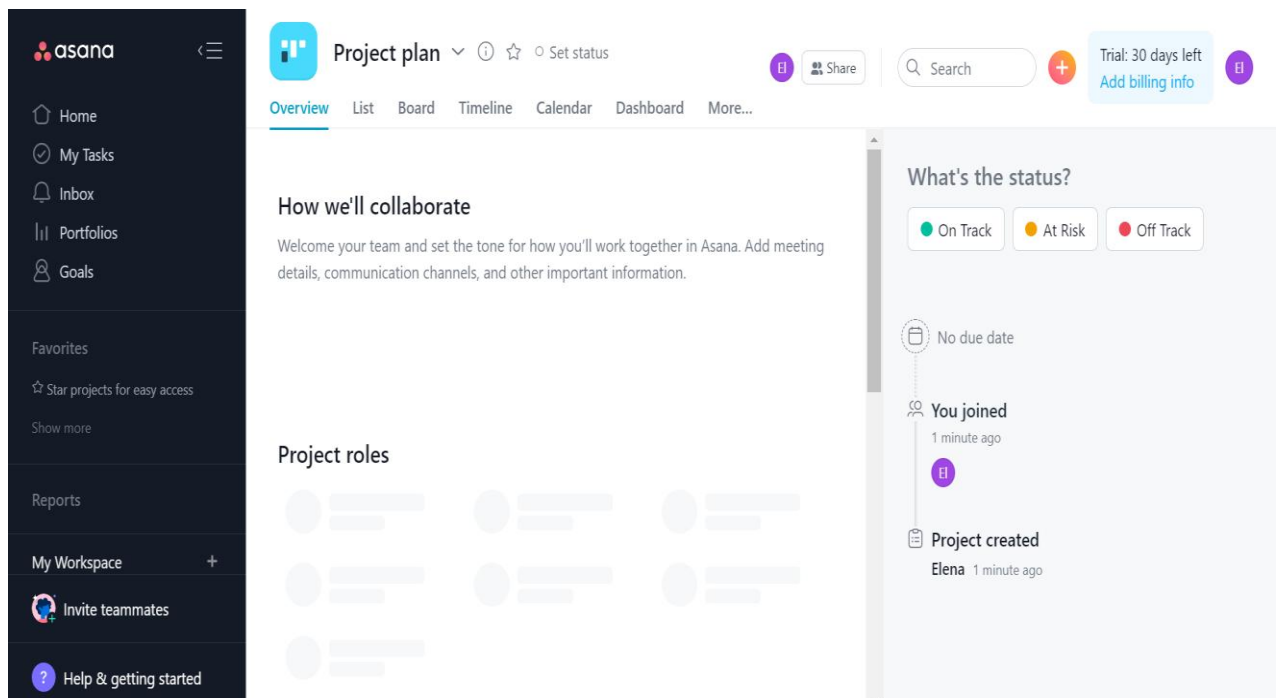


Рисунок 3.2 – інтерфейс системи управління проектами Asana

Trello - інструмент для командної роботи, управління проектами і збільшення особистої продуктивності. Він дозволяє працювати над

проєктами як індивідуально, так і в команді. Інструмент дає можливість підключити до роботи над завданнями і проєктом сторонніх виконавців [27].

Trello є більш простою та зручною системою, її використовують члени команди розробника лабораторії RELDES. Тому при виконанні завдань на дипломну роботу було прийнято рішення використовувати систему управління проєктами Trello. Вона заснована на парадигмі керування проєктами, відомої як Kanban.

Методологія Kanban - це система постановки завдань, при якій всі етапи проєкту візуалізуються на спеціальній дошці. Для спрощення контролю над проєктом робочий процес візуалізують на дошці, поділеної на колонки. Кожна колонка - це поточний стан робіт. Класичний варіант дошки (рис. 3.3) включає в себе три колонки: "Що треба зробити", "В процесі виконання", "Виконано". Безпосередньо завдання відображають в Kanban-картках, де можна прочитати їх опис, рівень важливості та додаткову інформацію. Коли завдання завершує певний етап, картку з її описом переносять у відповідну колонку. Поглянувши на дошку, можна відразу зрозуміти, яка ситуація з проєктом. Кожне завдання по мірі виконання розміщується у відповідній колонці [28].

Саме на основі Kanban дошок побудован інтерфейс Trello (рис. 3.4). Такий підхід дозволяє в досить зручному форматі переглядати поставлені задачі та слідкувати за процесом їх виконання.

Картки Trello мають безліч можливостей. Вони призначені для обговорень, голосувань, завантаження файлів і даних. Є можливість задавати дедлайни, призначати текстові та колірні мітки. Щоб до будь-якого завдання привласнити виконавця, необхідно в картці вибрати його зі списку робочої команди або просто перетягнути на завдання аватар виконавця. Важливо зауважити, що всі учасники команди бачать зміни в реальному часі і можуть спостерігати стану окремих виконавців - онлайн або офлайн [27].

Що треба зробити	В процесі виконання	Виконано
Завдання 4 Завдання 6 Завдання 5	Завдання 3	Завдання 1 Завдання 2

Рисунок 3.3 – Схема класичної дошки Kanban

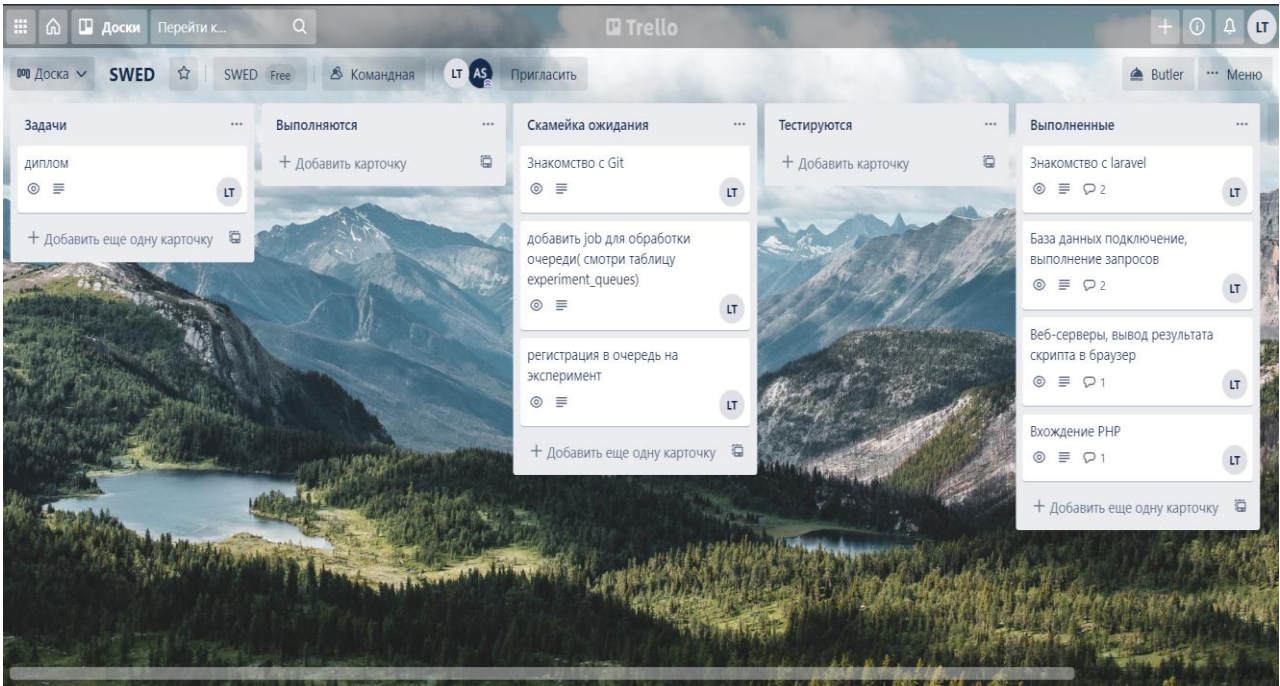


Рисунок 3.4 – Интерфейс системы Trello

3.2 Розгортання проєкту в середовищі віртуалізації

Дуже часто виникають проблеми на тлі того, що середовище розробки значно відрізняється від середовища, де буде працювати кінцевий застосунок. Для того щоб створити необхідне оточення для розробки застосунку та уникнути конфліктів з основним середовищем доцільно використовувати програмне забезпечення для створення ізольованих середовищ розробки, таких як Vagrant, Nanobox, Docker та ін.

Vagrant - це інструмент, що дозволяє уніфікувати середовище розробки і середовище де цей код буде розгортатися, використовуючи технологію віртуалізації. Основою Vagrant є файл з повністю налаштованим та готовим до використання віртуальним середовищем. Основа Vagrant'a - це box'и. Box - файл з повністю налаштованим та готовим до використання віртуальним середовищем [29, 30].

Nanobox – платформа, яка автоматизує створення ізольованих середовищ для місцевих та виробничих застосувань. При локальній розробці Nanobox забезпечує інфраструктуру застосунку всередині віртуальної машини. Будь-які зміни, внесені у код проєкту, відображаються у віртуальному середовищі. Після побудови та тестування коду локально, Nanobox забезпечує та розгортає таку ж саму інфраструктуру на виробничій платформі [31]. Іншими словами Nanobox - платформа для розробників, що дозволяє зосередитися на коді, а не на налаштуванні, за рахунок спрощення налаштування середовища застосунку [32].

Docker - програмне забезпечення для автоматизації розгортання і управління застосунками в середовищі віртуалізації на рівні операційної системи, воно дозволяє «упакувати» застосунок з усім його оточенням і залежностями в контейнер, а також надає середовище з управлінням контейнерами [33].

Оскільки Docker [34] є найпопулярнішою програмною платформою, то для виконання дипломної роботи була використана саме ця платформа.

На рисунку 3.5 зображено інтерфейс застосунку Docker Desktop for Windows, у якому запущено проєкт ВЛ RELDES.

У джерелі [35] сказано, що контейнери надають схожий з віртуальними машинами рівень ізоляції, але завдяки правильному залученню низькорівневих механізмів основної операційної системи роблять це зі значно меншим навантаженням на ресурси.

Щоб розгорнути проєкт лабораторії RELDES було використано інструментальний засіб Docker Compose. Він входить до складу Docker та призначений для вирішення завдань, пов'язаних з розгортанням проєктів.

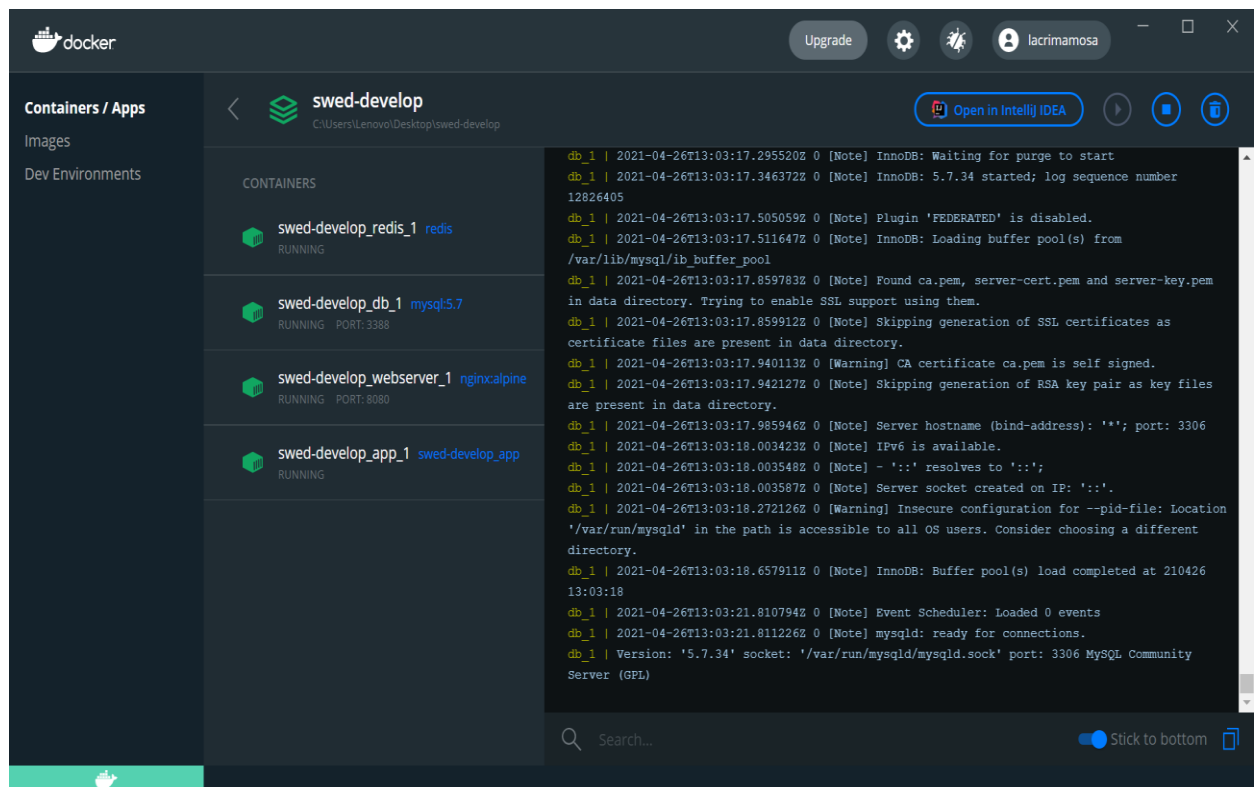


Рисунок 3.5 – інтерфейс застосунку Docker Desktop for Windows

Зазвичай для забезпечення функціонування вебпроєктів використовується декілька сервісів. Наприклад, вебсайт, який виконуючи аутентифікацію користувачів, підключається до бази даних. Такий проєкт може складатися з двох сервісів. Перший забезпечує роботу самого вебсайту, другий відповідає за підтримку бази даних. Для таких проєктів доцільно

використовувати Docker Compose. Саме ця технологія дозволяє за допомогою однієї команди запускати декілька сервісів [36].

3.3. Алгоритм функціонування програми

Розроблене ПЗ функціонує з дотриманням наступного алгоритму дій.

Етап 1. Початок роботи програми. На цьому етапі користувач натискає кнопку «Register».

Етап 2. Програма відправляє запит на додання користувача до черги.

Етап 3. Програма розраховує час початку експерименту.

Етап 4. Програма створює екземпляр задачі.

Етап 5. Програма розраховує час завершення експерименту.

Етап 6. Програма додає користувача до черги.

Етап 7. Програма виводить користувачу інформацію: довжина черги, час початку і час завершення експерименту.

Етап 8. Програма запускає задачу.

Етап 9. Задача шукає користувача, який завершив експеримент.

Етап 10. Якщо користувача, що завершив експеримент, знайдено, то програма переходить до етапу 11. Якщо такого користувача не знайдено, то програма переходить до етапу 12.

Етап 11. Задача видаляє знайденого користувача з черги.

Етап 12. Задача чекає 10 секунд та переходить до етапу 9.

Етап 13. Завершення роботи програми.

Схема даного алгоритму показана на рисунку 3.6.

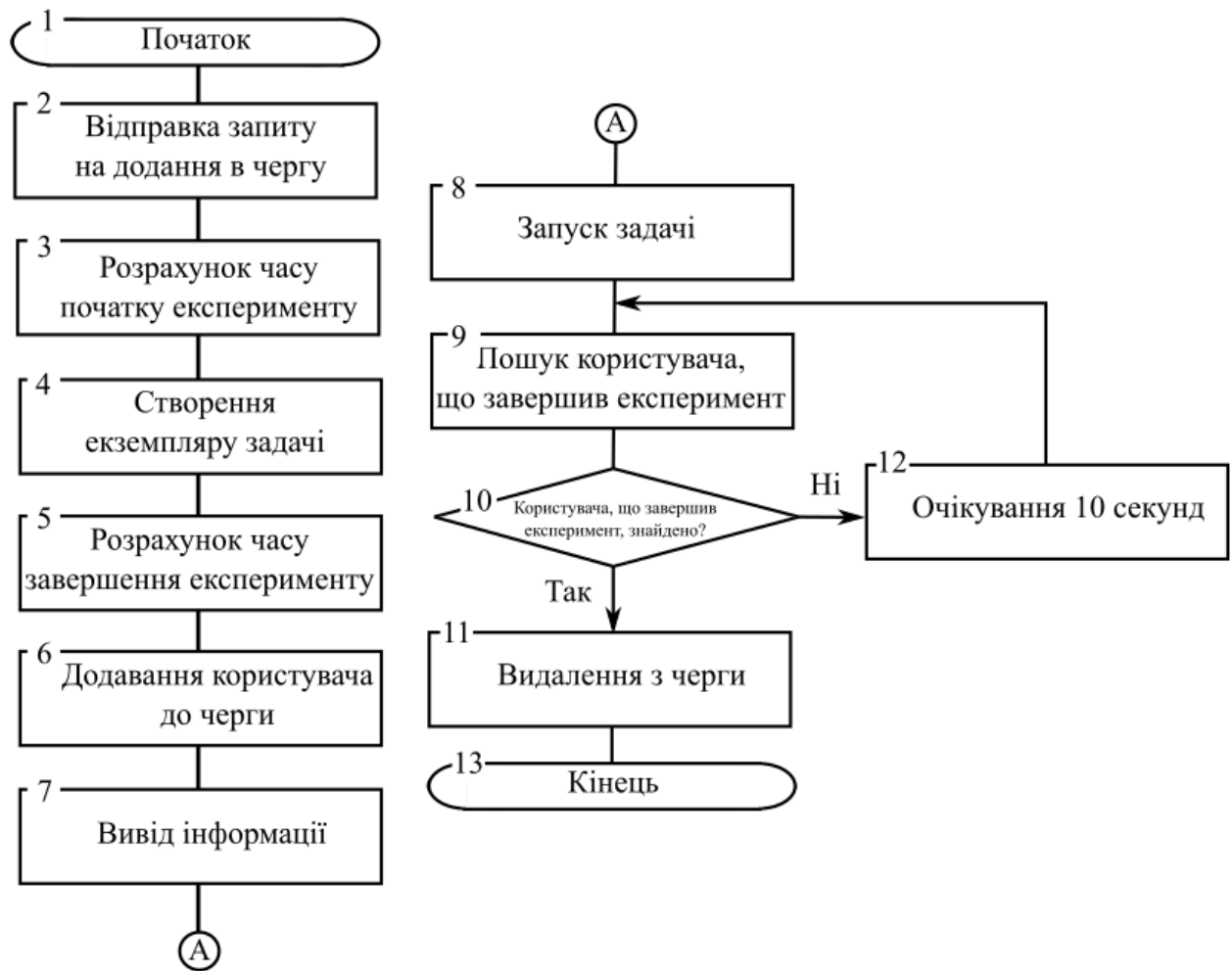


Рисунок 3.6 – Схема алгоритму роботи ПЗ

3.4. Опис особливостей програмної реалізації

Першим кроком програмної реалізації ПЗ було створення кнопки «Register» (рис. 3.7). Згідно із завданням, цю кнопку повинні бачити лише аутентифіковані користувачі. Для визначення контенту для аутентифікованих користувачів та для гостей сайту шаблонізатор Blade має спеціальні директиви. При створенні кнопки використовувалась директива `@auth ... @endauth`, весь контент, який розміщено в середині, буде відображатися лише для аутентифікованих користувачів.

Директива `@if(!$queue)...@ endif` показує користувачу кнопку за умови, якщо він не стоїть у черзі на цей експеримент.

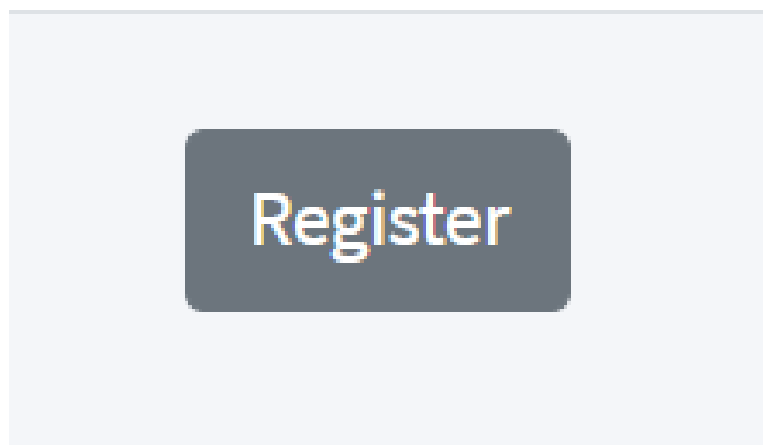


Рисунок 3.7 – Розроблена кнопка

Також було використано директиву `@csrf` для створення поля введення прихованого токена. Це робиться для запобігання CSRF (cross-site request forgery) атак. Laravel автоматично генерує токен CSRF для кожної активної користувальницької сесії, керованої застосунком. Цей токен використовується для перевірки того, що аутентифіцирований користувач є особою, яка фактично виконує запити до застосунку. Оскільки цей токен зберігається в сеансі користувача і змінюється кожен раз при повторному створенні сеансу, шкідливий застосунок не може отримати до нього доступ [37].

Після натискання кнопки «Register» виконується AJAX-запит методом POST. За відправку запиту відповідає функція `ajax` (`{url: '/experiments/register', method: 'POST', data: data}`). Вона приймає аргументи:

- `url: '/experiments/register'` - уніфікований покажчик ресурсу;
- `method: 'POST'` – метод запиту;
- `data: data` – дані користувача.

Технологія AJAX в основному використовується для створення асинхронних запитів до сервера. Асинхронний запит - це такий запит, який виконується в фоновому режимі і не заважає користувачеві взаємодіяти зі сторінкою [38]. При використанні AJAX немає необхідності оновлювати кожен раз всю сторінку, так як оновлюється тільки її конкретна частина. Це набагато зручніше, так як не доводиться довго чекати [39]. Отже для

відправки запиту при натисканні кнопки було доцільно використовувати технологію AJAX.

Після відправки запиту маршрутизатор реєструє цей запит та викликає метод `registerForExp (Request $request)` класу `DescriptionController`, який приймає аргумент `$request` - екземпляр поточного HTTP-запиту. Він створює та зберігає модель, після чого викликає методи `_timeStart(ExperimentQueue $eq)` та `_addQueue (ExperimentQueue $eq, $username, $timeStart)` класу `QueueService`.

Метод `_timeStart` приймає аргумент `$eq` – створена модель, та розраховує час початку експерименту. Метод `_addQueue` приймає аргументи: `$eq` – створена модель, `$username` – ім'я користувача та `$timeStart` – час початку експерименту. Він створює екземпляр задачі `ExperimentJob` в яку передає модель, ім'я користувача та час експерименту. Після чого, під'єднує до задачі `Redis`.

Далі метод перевіряє на який експеримент була реєстрація та додає користувача до черги цього експерименту. Після цього запускає задачу `ExperimentJob`.

Задача `ExperimentJob` містить метод `handle()` класу `ExperimentJob`, який видаляє з черги користувача, що завершив експеримент та виводить його ім'я у косоль.

У класі `Menu` розроблені методи які отримують інформацію для користувача про стан черги для виводу у бокову панель (рис. 3.8). Розроблені методи класу `Menu` наведені у таблиці 3.1.

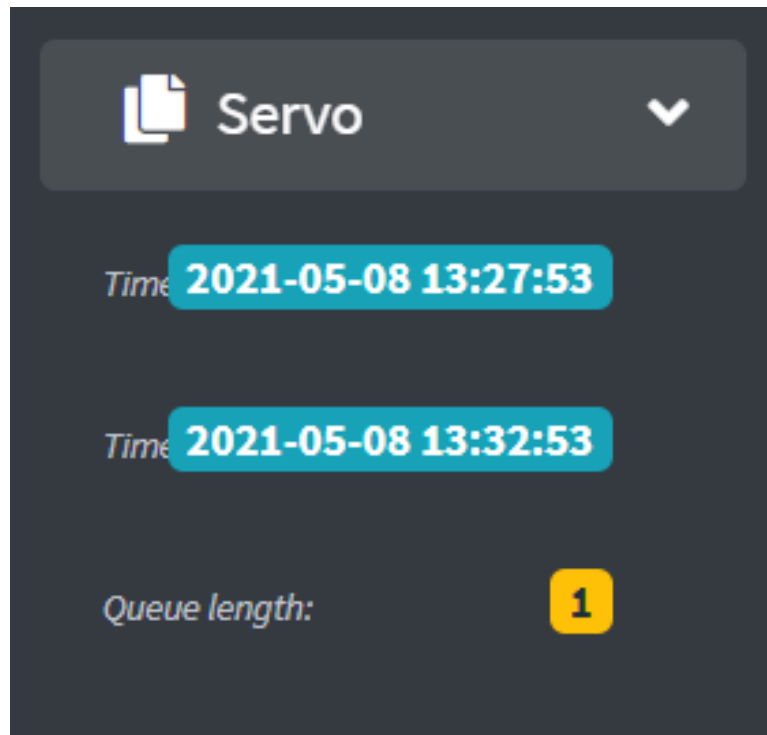


Рисунок 3.8 – Інформація про експеримент у боковій панелі

Таблиця 3.1 - Розроблені методи класу Menu

Назва методу	Властивості
getExperiments	Отримує дані для виводу
getTimeQueue	Отримує час початку та час закінчення кожного експерименту.
getCountQueue	Розраховує довжину черги на кожний експеримент.
checkRecord	Якщо користувач зареєстрован на експеримент, то повертає відформатований час початку та час завершення, інакше повертає значення null.

Діаграму розроблених класів представлено на рисунку 3.9.

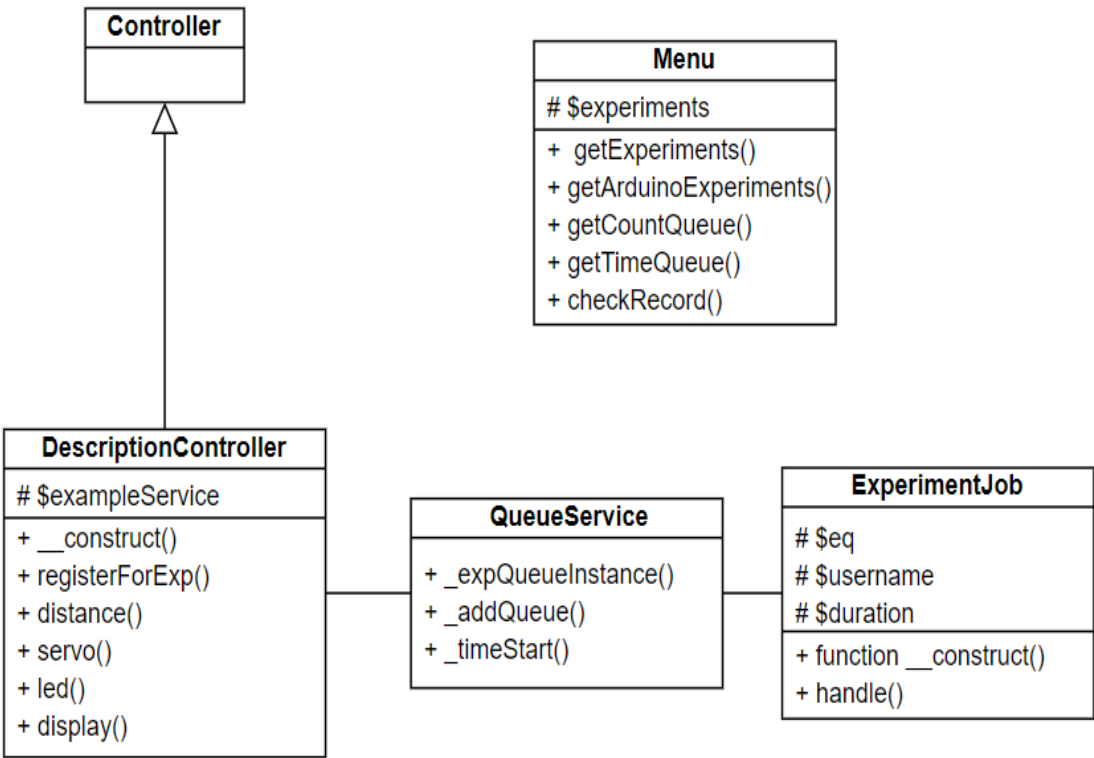


Рисунок 3.9 – Діаграма розроблених класів

Для написання власних методів використовувались вбудовані методи та функції, які наведені у табл. 3.2 та табл. 3.3.

Основні змінні, що використовувались для написання програми наведені у таблиці 3.4.

Таблиця 3.2 - Вбудовані методи, що використовувались при написанні програми

Назва методу	Властивості
1	2
delay	Метод затримки при відправленні задачі.
delete	Метод видалення конструктора запитів, який може використовуватися для видалення записів з таблиці.
first	Метод повертає перший елемент у колекції, який проходить заданий тест на істинність.

Продовження таблиці 3.2

1	2
format	Метод перетворює дату в будь-який формат, який буде вказано
onConnection	Метод дозволяє обрати з'єднання, яке слід використовувати для відправки задачі.
onQueue	Метод для вибору черги.
sleep	Метод змушує задачу «спати» указану кількість секунд
update	Метод оновлення. Він приймає масив пар стовпців та значень, що вказують на стовпці, які слід оновити.
where	Метод фільтрує колекцію за заданою парою ключ/значення.

Таблиця 3.3 – Вбудовані функції, що використовувались при написанні програми

Назва функції	Властивості
dispatch	Функція штовхає дане завдання на виконання.
getMessage	Функція повертає повідомлення у вигляді рядка.
now	Функція створює новий екземпляр для поточного часу.

Таблиця 3.4 – Основні змінні, які використовувались у програмі

Назва змінної	Значення
\$counts	Довжина черги
\$experimen	Ідентифікатор експерименту
\$job	Екземпляр задачі
\$model	Створена модель
\$timeEnd	Час завершення експерименту
\$timeStart	Час початку експерименту
\$username	Ідентифікатор користувача

4 КЕРІВНИЦТВО ПРОГРАМІСТА

4.1 Призначення програми та умови виконання програми

Програмний застосунок призначений для використання у віддаленій лабораторії RELDES. Він дозволяє додавати користувача до черги на виконання експерименту, стежити за її станом та видаляти користувача з черги після завершення експерименту. Це організовує доступ до експериментів та підвищує ефективність роботи користувачів.

Для проведення експериментів та запису до черги на них необхідно мати комп'ютер під'єднаний до мережі Інтернет на якому встановлений веб-браузер. Комп'ютер повинен задовольняти наступні апаратно-технічні вимоги:

- операційна система Windows або Linux;
- процесори Intel або AMD;
- 2048 Мб оперативної пам'яті або більше;
- відеокарта з підтримкою роздільної здатності екрану 1280 x 720 пікселів та більше.

4.2 Характеристика програми

Програмний застосунок розроблено на скриптовій мові програмування PHP з використанням фреймворку Laravel та інтегрованого середовища розробки для PHP-проектів. Розроблений програмний застосунок дозволяє користувачу займати чергу на експерименти ВЛ RELDES та стежити за її станом.

4.3 Звертання до програми

Для того, щоб стати в чергу на проведення експериментів у лабораторії RELDES необхідно бути зареєстрованим в системі та увійти в

свій акаунт. Для цього на головній сторінці сайту є вкладки для реєстрації та авторизації. Після натискання на вкладку «Реєстрація» або «Вхід» користувачу надається відповідна форма. Після входу в свій акаунт користувач може перейти до обраного експерименту та стати у чергу натиснувши кнопку «Register». Після натискання кнопки вона зникне, а на боковій панелі користувач зможе побачити довжину черги, час початку та час завершення експерименту. Після завершення користувачем експерименту кнопка «Register» з'явиться знову.

4.4 Вхідні та вихідні дані

Вхідними даними є ідентифікатор користувача та ідентифікатор експерименту, які програма отримує після того як користувач натисне кнопку «Register». Вихідними даними є час початку експерименту, час закінчення експерименту та довжина черги.

4.5 Додавання експерименту

У разі додавання до віддаленої лабораторії нового експерименту, необхідно додати його до створеного модуля черги. Для цього у файлі QueueService.php необхідно додати константу NEW_EXPERIMENT = new_experiment_id, де NEW_EXPERIMENT назва нового експерименту, а new_experiment_id ідентифікатор цього експерименту, у цьому ж файлі слід прописати наступний код

```
if ($eq->experiment_id == self::NEW_EXPERIMENT)
{ $job->onQueue('new_experiment'); }.
```

Далі у файлі Menu.php теж необхідно додати константу NEW_EXPERIMENT = experiment_id та до масиву \$experiments додати ключ 'new_experiment'. У цьому ж файлі до методу getCountQueue() додати наступний код

```

$queue['new_experiment'] =
ExperimentQueue::where(['experiment_id' =>
self::NEW_EXPERIMENT, 'status' => self::DISABLE])->
count();.

```

До методу getTimeQueue() додати

```

$result = ExperimentQueue::where(['experiment_id'
=> self::NEW_EXPERIMENT, 'status' => self::ACTIVE])->
first();

```

```

$queue['new_experiment'] = $this->
checkRecord($result);.

```

У файлі DescriptionController.php необхідно додати наступний метод

```

public function new_experiment()
{ if (Auth::check())
{return
view('arduino.experiments.new_experiment.page', [
'queue' =>
QueueService::_expQueueInstance(new_experiment_id)
]);}
return
view('arduino.experiments.new_experiment.page');}.

```

5 КЕРІВНИЦТВО КОРИСТУВАЧА

5.1 Призначення програми

Розроблене ПЗ для модернізації лабораторії RELDES спрямоване на коректну організацію доступу до обладнання ВЛ, що дозволить студентам та викладачам раціонально використовувати час, відведений на виконання експериментів та підвищити ефективність своєї роботи. В результаті роботи програми користувач може займати чергу на експерименти та стежити за її станом, що робить користування лабораторією RELDES більш зручним.

5.2 Умови виконання програми

Для коректної роботи програми необхідно виконання наступних умов:

- лабораторія RELDES має доступ до мережі Інтернет;
- увімкнено стенд з експериментами;
- користувач авторизований у системі;
- на сайті лабораторії не проводяться технічні роботи.

Також необхідно мати комп'ютер під'єднаний до мережі Інтернет на якому встановлений веб-браузер. Комп'ютер повинен задовольняти наступні апаратно-технічні вимоги:

- операційна система Windows або Linux;
- процесори Intel або AMD;
- 2048 Мб оперативної пам'яті або більше;
- відеокарта з підтримкою роздільної здатності екрану 1280 x 720 пікселів та більше.

5.3 Виконання програми

Для того, щоб стати в чергу на проведення експериментів у лабораторії RELDES необхідно бути зареєстрованим в системі та увійти в

свій акаунт. Для цього на головній сторінці сайту є вкладки для реєстрації та авторизації. Після натискання на вкладку «Реєстрація» або «Вхід» надається відповідна форма. Після входу в свій акаунт можна перейти до обраного експерименту та стати у чергу натиснувши кнопку «Register» Після натискання кнопки вона зникне, а на боковій панелі з'явиться довжина черги, час початку та час завершення експерименту (рис. 5.1). Після завершення експерименту кнопка «Register» з'явиться знову.

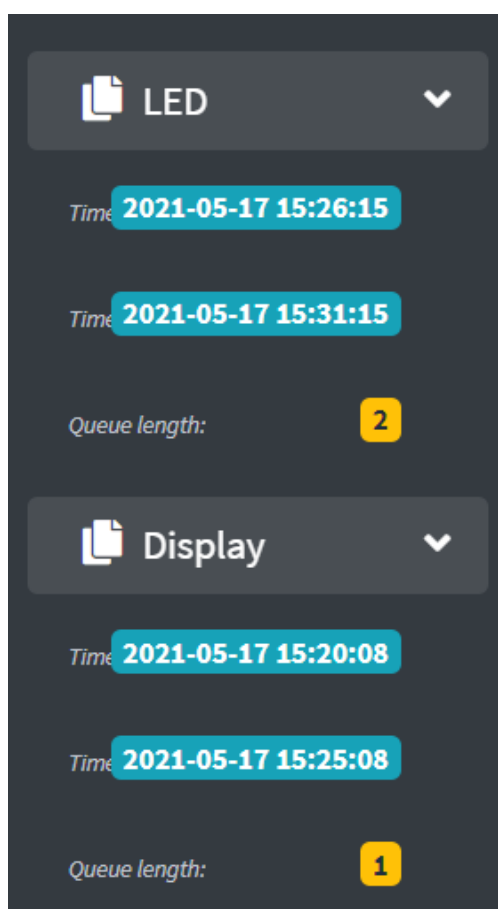


Рисунок 5.1 – Інформація про експерименти «LED» та «Display» у боковій панелі

ВИСНОВКИ

В результаті виконання дипломної кваліфікаційної роботи бакалавра було проведено огляд існуючих ВЛ. Характерною особливістю таких лабораторій є те, що вони представляють собою програмно-апаратний комплекс, який дозволяє проводити віддалені експерименти без безпосереднього контакту з реальним обладнанням. Користувач комп'ютера з доступом до мережі Інтернет може з'єднатися з лабораторією, перебуваючи в будь-якій точці земної кулі та провести необхідні йому експерименти.

Оскільки проведення експериментів у ВЛ передбачає роботу з реальним обладнанням, то декілька користувачів не можуть використовувати одне й те саме обладнання одночасно. Для цього в лабораторіях передбачена система черги. Як показали дослідження існує два типи організації черг на основі попереднього резервування та в режимі онлайн.

В процесі модернізації лабораторії RELDES виникла задача візуалізації розробленого механізму роботи черги. Для реалізації ПЗ черги було оглянуто та проаналізовано мови розробки вебзастосунків PHP, Ruby та Python. Через ряд переваг було обрано мову PHP. Оскільки використання фреймворків значно прискорює розробку вебпроектів та полегшує подальшу їх підтримку, було оглянуто та проаналізовано найпопулярніші PHP-фреймворки: Codeigniter, Yii та Laravel. Для створення ПЗ було обрано фреймворк Laravel. Розроблене ПЗ надає можливість користувачу займати чергу на експерименти ВЛ RELDES та стежити за її станом.

В подальшій роботі планується продовжити модернізацію лабораторії RELDES, а саме створити таймери які будуть відраховувати час до початку та до кінця експерименту. Також планується розробити модуль який буде виводити інформацію з COM-порту плат Arduino на вебсторінку для експериментів: LED; Display; Distance; Servo. Основні положення й результати роботи були представлені в тезах доповіді НПК Тиждень науки – 2021.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. WebLab-Deusto [Electronic resource]. – Access mode: <https://weblab.deusto.es/website/>.
2. Welcome to the Electromet Remote Labs [Electronic resource]. – Access mode: <https://lab.electromet.com/Home>.
3. Створення та використання віддаленої лабораторії як відкритого онлайн-ресурсу / [А. Туленков, О. Соколянський, А. Пархоменко та ін.] // Вісник Національного університету “Львівська політехніка”. Серія: Інформатизація вищого навчального закладу. – Львів: Львівська політехніка, 2018. – № 903. – 90 с.
4. WebLab-Deusto-CPLD: A Practical Experience [Text] / [J. García-Zubía, I. Angulo, P. Orduna et al.] // International Journal of Online Engineering. - 2012. – vol. 8(1). – P.17-18.
5. Grid of online lab devices Ilmenau: Demonstration of online experimentation [Text] / [K. Henke, H. D. Wuttke, T. Vietzke et al.] // Remote Engineering and Virtual Instrumentation: 13th International Conference, Madrid, Spaine, 2015: proceedings. – Los Alamitos: IEEE, 2015. – P. 109-110.
6. Віддалена лабораторія RELDES [Електрон. ресурс]. – Режим доступу: <https://swed.zntu.edu.ua/>.
7. Implementation of reusable solutions for remote laboratory development [Text] / [A. Parkhomenko, O. Gladkova, A. Sokolyanskii et al.] // International Journal of Online Engineering. - 2016. - vol.12(7). - P. 24–29.
8. Development and application of remote laboratory for embedded systems design [Text] / [A. Parkhomenko, O. Gladkova, E. Ivanov et al.] // International Journal of Online Engineering. - 2015. – vol.11(3). - P. 27–31.
9. Laboratory as a Service (LaaS): A Novel Paradigm for Developing and Implementing Modular Remote Laboratories [Text] / [M. Tawfik, C. Salzmann, D. Gillet et al.] // International Journal of Online Engineering. – 2014. – vol. 10(4). – P. 13-21.

10. The smart device specification for remote labs [Text] / [C. Salzmann, S. Govaerts, W. Halimi et al.] // Remote Engineering and Virtual Instrumentation: 12th International Conference, Bangkok, Thailand, 2015: proceedings. – Los Alamitos: – IEEE, 2015. – P. 199-208.
11. Redis на практических примерах [Электрон. ресурс]. – Режим доступа: <https://habr.com/ru/company/manychat/blog/507136/>
12. PHP [Electronic resource]. – Access mode: <https://en.wikipedia.org/wiki/PHP>.
13. Ruby on Rails [Electronic resource]. – Access mode: https://ru.wikipedia.org/wiki/Ruby_on_Rails.
14. Лучшие фреймворки Python для веб-разработки, которые подойдут начинающим [Электрон. ресурс]. – Режим доступа: <https://tproger.ru/articles/best-python-frameworks-for-beginners/>.
15. PHP [Электрон. ресурс]. – Режим доступа: <https://siteactiv.ru/terminy/php/>.
16. Язык программирования PHP [Электрон. ресурс]. – Режим доступа: https://depix.ru/articles/yazyk_programmirovaniya_php.
17. PHP | Unique Features [Electronic resource]. – Access mode: <https://www.geeksforgeeks.org/php-unique-features/>.
18. Какой PHP-фреймворк выбрать: сравниваем Symfony, Laravel и Yii2 [Электрон. ресурс]. – Режим доступа: <https://ru.hexlet.io/blog/posts/kakoy-php-freymvork-vybrat-sravniваем-symphony-laravel-i-yii2>.
19. Фреймворк CODEIGNITER V 2.1.3 [Электрон. ресурс]. – Режим доступа: <https://php5.kiev.ua/codeigniter/>.
20. Полное руководство по Yii 1.1 [Электрон. ресурс]. – Режим доступа: <https://www.yiiframework.com/doc/guide/1.1/ru/quickstart.what-is-yii>.
21. Что такое Ларавел? [Электрон. ресурс]. – Режим доступа: <https://wezom.com.ua/blog/7-prichin-po-kotorym-stoit-ispolzovat-laravel-pri-sozdanii-prilozhenij-dlya-kompanij#1>.
22. 18 ключевых особенностей Laravel фреймворка [Электрон.

ресурс]. – Режим доступа: <http://cccp-blog.com/laravel/laravel-frejmwork#18-klyuchevyh-osobennostej-laravel-frejmworka>.

23. PhpStorm [Electronic resource]. – Access mode: <https://www.jetbrains.com/phpstorm/>.

24. Features [Electronic resource]. – Access mode: <https://www.jetbrains.com/phpstorm/features/>.

25. Описание Jira [Электрон. ресурс]. – Режим доступа: <https://startpack.ru/application/jira>.

26. Asana – переосмыслите управление рабочим процессом [Электрон. ресурс]. – Режим доступа: <https://cloudfresh.com/ru/cloud-blog/asana-pereosmyslyte-upravlenye-rabochym-protsessom/>.

27. Trello [Электрон. ресурс]. – Режим доступа: <https://startpack.ru/application/fog-trello>.

28. Kanban: принципы и возможности в управлении проектами [Электрон. ресурс]. – Режим доступа: <https://www.unisender.com/ru/support/about/glossary/kanban/>.

29. Введение в Vagrant [Электрон. ресурс]. – Режим доступа: <https://khashtamov.com/ru/vagrant-how-to-setup/>.

30. Что такое Vagrant и для чего его использовать [Электрон. ресурс]. – Режим доступа: <https://blog.vistro.ru/vagrant/>.

31. Nanobox [Electronic resource]. – Access mode: <https://github.com/nanobox-io/nanobox>.

32. Nanobox [Electronic resource]. – Access mode: <https://www.capterra.com/p/159391/Nanobox/>.

33. Docker [Electronic resource]. – Access mode: <https://www.docker.com/>.

34. Docker [Электрон. ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/Docker>.

35. Полное практическое руководство по Docker: с нуля до кластера на AWS [Электрон. ресурс]. – Режим доступа: <https://habr.com/ru/post/310460/>.

36. Руководство по Docker Compose для начинающих [Электрон. ресурс]. – Режим доступа: <https://habr.com/ru/company/ruvds/blog/450312/>.

37. Preventing CSRF Requests [Electronic resource]. – Access mode: <https://laravel.com/docs/8.x/csrf#preventing-csrf-requests>.

38. Что такое асинхронный запрос AJAX [Электрон. ресурс]. – Режим доступа: <https://itchief.ru/lessons/javascript/ajax-asynchronous-requests-in-the-examples#chto-takoe-asinhronnyy-zapros-ajax>.

39. AJAX для новичков [Электрон. ресурс]. – Режим доступа: <https://habr.com/ru/post/14246/>.

ДОДАТОК А
Текст програми

A1. Код файлу `page.blade.php`

```

...
@auth //контент для аутентифікованих користувачів
@if(!$queue)
    <div class="container">
        <div class="text-center">
            <form id="form" method="POST" data-
exp="4" data-user={{ Auth::user()->id }}>
                @csrf //захист від підробки
міжсайтових запитів
                <button type="submit" id="test"
class="btn btn-secondary">Register</button> //кнопка
реєстрації на експеримент
            </form>
        </div>
    </div>
@endif
@endauth
...

```

A2. Код файлу `experiments.js`

```

let token = $('meta[name=csrf-token]').attr('content')

//функціонал який виконує кнопка після її натискання
$('#form').submit(function(event) {
    event.preventDefault()
    var $this = $(this)

    var exp_id = $this.attr('data-exp')
    var user_id = $this.attr('data-user')

    data = {

```

```

        '_token': token,
        'exp_id': exp_id,
        'user_id': user_id
    }

    //створення Ajax запиту
    $.ajax({
        url: '/experiments/register',
        method: 'POST',
        data: data
    }).done(function() {
        $this.remove()
    })
})

```

A3. Код файлу webpack.mix.js

```

...

//компіляція JavaScript

.js(
    'resources/js/pages/experiments.js',
    'public/js/pages/experiments.js'
);

...

```

A4. Код файлу web.php

```

...

//реєстрація запиту та виклик методу
registerForExp
Route::post('/register',
'DescriptionController@registerForExp')->
name('experiments.register');

...

```

A5. Код файлу DescriptionController.php

```
<?php

namespace App\Http\Controllers\Arduino;

use App\Http\Controllers\Controller;
use App\Jobs\ExperimentJob;
use App\Models\Experiment\Experiment;
use App\Models\Experiment\ExperimentQueue;
use App\Services\Experiment\Example;
use App\Services\QueueService;
use Exception;
use Illuminate\Support\Facades\Auth;
use Illuminate\Http\Request;

//створення класу DescriptionController, який наслідує
клас Controller
class DescriptionController extends Controller
{
    protected $exampleService;

    public function __construct()
    {
        $this->exampleService = new Example();
    }

    public function registerForExp(Request $request)
    {
        //створення моделі
        try {
            $model = new ExperimentQueue;
            $model->user_id = $request->user_id;
            $model->experiment_id = $request->exp_id;

            $model->save();

            //розрахунок часу початку експерименту
            $timeStart =
QueueService::_timeStart($model);
            //додавання до черги
            QueueService::_addQueue($model, auth()-
>user()->name, $timeStart);
```

```

        return response()->json(['model' =>
$model]);
    } catch (Exception $ex) {
        return response()->json(['errMsg' => $ex-
>getMessage()])->setStatusCode(404);
    }
}

//отримання екземплярів черг для експериментів
public function distance()
{
    if (Auth::check()) {
        return
view('arduino.experiments.distance.page', [
            'queue' =>
QueueService::_expQueueInstance(1)
        ]);
    }

    return
view('arduino.experiments.distance.page');
}

public function servo()
{
    if (Auth::check()) {
        return
view('arduino.experiments.servo.page', [
            'queue' =>
QueueService::_expQueueInstance(2)
        ]);
    }

    return view('arduino.experiments.servo.page');
}

public function led()
{
    if (Auth::check()) {
        return view('arduino.experiments.led.page',
[
            'queue' =>
QueueService::_expQueueInstance(3)
        ]);
    }
}

```

```

        return view('arduino.experiments.led.page');
    }

    public function display()
    {
        $examples = $this->exampleService-
>getExamples('display');

        if (Auth::check()) {
            return
view('arduino.experiments.display.page', [
                'queue' =>
QueueService::_expQueueInstance(4),
                'examples' => $examples
            ]);
        }

        return view('arduino.experiments.display.page',
[
            'examples' => $examples
        ]);
    }
}

```

A6. Код файлу QueueService.php

```

<?php

namespace App\Services;

use App\Jobs\ExperimentJob;
use App\Models\Experiment\Experiment;
use App\Models\Experiment\ExperimentQueue;

//створення класу QueueService
class QueueService
{
    //ідентифікатори експериментів
    const DISTANCE = 1;
    const SERVO = 2;
    const LED = 3;
}

```

```

const DISPLAY = 4;

//метод отримує користувача, який знаходиться першим в
черзі
public static function _expQueueInstance($id)
{
    return ExperimentQueue::where([
        'experiment_id' => $id,
        'user_id' => auth()->user()->id
    ])->first();
}

//метод додавання до черги
public static function _addQueue(ExperimentQueue
$eq, $username, $timeStart)
{
    $experiment = Experiment::where('id', $eq-
>experiment_id)->first();
    $job = new ExperimentJob($eq, $username,
    $experiment->time);

    $job->onConnection('redis'); //під'єднання до
Redis

    //додавання до відповідної черги
    if ($eq->experiment_id == self::DISTANCE) {
        $job->onQueue('distance');
    }

    if ($eq->experiment_id == self::SERVO) {
        $job->onQueue('servo');
    }

    if ($eq->experiment_id == self::LED) {
        $job->onQueue('led');
    }

    if ($eq->experiment_id == self::DISPLAY) {
        $job->onQueue('display');
    }

    $eq->start = $timeStart;

    $job->delay($timeStart);
}

```

```

//розрахунок часу закінчення експерименту
    $timeEnd = $eq->start->modify('+ ' .
    $experiment->time . ' seconds');

    $eq->end = $timeEnd;

    $eq->save();

    dispatch($job); //запуск задачі
}

//метод для розрахунку часу початку експерименту
public static function _timeStart(ExperimentQueue
$eq)
{
    $firstModel = ExperimentQueue::where(
        'experiment_id',
        $eq->experiment_id
    )->first();

    if ($firstModel->end < now()) {
        return now();
    }

    return $firstModel->start ? $firstModel->end :
now();
}
}

```

A7. Код файлу ExperimentQueue.php

```

<?php

namespace App\Models\Experiment;

use App\Models\User;
use App\Models\Experiment\Experiment;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use
Database\Factories\Experiments\ExperimentQueueFactory;

//створення моделі
class ExperimentQueue extends Model

```

```

{
    use HasFactory;
    /**
     * The attributes that are mass assignable.
     *
     * @var array
     */
    protected $fillable = [
        'experiment_id',
        'user_id',
        'start',
        'end',
        'status'
    ];
    /**
     * The attributes that should be hidden for arrays.
     *
     * @var array
     */
    protected $hidden = [];

    /**
     * The attributes that should be cast to native
types.
     *
     * @var array
     */
    protected $casts = [];

    /**
     * Date attributes
     *
     * @var array
     */
    protected $dates = [
        'start', 'end'
    ];

    /**
     * Relation to experiment table
     *
     * @return
\Illuminate\Database\Eloquent\Relations\BelongsTo
     */
    public function experiment()

```

```

    {
        return $this->belongsTo(Experiment::class);
    }

    /**
     * Relation to user table
     */
    public function user() {
        return $this->belongsTo(User::class);
    }

    /**
     * Create a new factory instance for the model.
     *
     * @return
     * \Illuminate\Database\Eloquent\Factories\Factory
     */
    protected static function newFactory()
    {
        return ExperimentQueueFactory::new();
    }
}

```

A8. Код файлу ExperimentJob.php

```

<?php

namespace App\Jobs;

use App\Models\Experiment\Experiment;
use App\Models\Experiment\ExperimentQueue;
use Illuminate\Bus\Queueable;
use Illuminate\Contracts\Queue\ShouldBeUnique;
use Illuminate\Contracts\Queue\ShouldQueue;
use Illuminate\Foundation\Bus\Dispatchable;
use Illuminate\Queue\InteractsWithQueue;
use Illuminate\Queue\SerializesModels;

//клас задачі ExperimentJob
class ExperimentJob implements ShouldQueue
{
    use Dispatchable, InteractsWithQueue, Queueable,
    SerializesModels;
}

```

```

protected $eq;
protected $username;
protected $duration;

/**
 * Create a new job instance.
 *
 * @return void
 */
public function __construct(ExperimentQueue $eq,
$username, $duration)
{
    $this->eq = $eq;
    $this->username = $username;
    $this->duration = $duration;
}

/**
 * Execute the job.
 *
 * @return void
 */
//метод шукає та видаляє користувача, який завершив
експеримент, з черги
public function handle()
{
    ExperimentQueue::where([
        'experiment_id' => $this->eq-
>experiment_id,
        'status' => 1
    ])->delete();

    $this->eq->update([
        'status' => 1,
    ]);

    sleep(10);

    // $this->eq->delete();

    echo $this->username . PHP_EOL;

    return 0;
}
}

```

A9. Код файлу Menu.php

```

<?php
namespace App\Services\Experiment;

use App\Models\Experiment\ExperimentQueue;

//клас Menu
class Menu
{
    //ідентифікатори експериментів
    const DISTANCE = 1;
    const SERVO = 2;
    const LED = 3;
    const DISPLAY = 4;

    //статуси експериментів
    const ACTIVE = 1;
    const DISABLE = 0;

    protected $experiments = [
        'distance',
        'servo',
        'led',
        'display'
    ];

    //отримання даних експериментів
    public function getExperiments()
    {
        $times = $this->getTimeQueue();
        $counts = $this->getCountQueue();
        $result = [];
        foreach ($this->experiments as $experiment) {
            $result[$experiment] = [
                'tts' => $times[$experiment]['start'],
                'tte' => $times[$experiment]['end'],
                'queue_qty' => $counts[$experiment]
            ];
        }

        return $result;
    }
}

```

```

    }

    public function getArduinoExperiments()
    {
        $exps = ExperimentQueue::with(['experiments' =>
function ($query) {
            $query->whereIn('experiments.name', $this->experiments);
        }])->get();
    }

//розрахунок довжини черги
    public function getCountQueue()
    {
        $queue['distance'] =
ExperimentQueue::where(['experiment_id' =>
self::DISTANCE, 'status' => self::DISABLE])->count();
        $queue['servo'] =
ExperimentQueue::where(['experiment_id' => self::SERVO,
'status' => self::DISABLE])->count();
        $queue['led'] =
ExperimentQueue::where(['experiment_id' => self::LED,
'status' => self::DISABLE])->count();
        $queue['display'] =
ExperimentQueue::where(['experiment_id' =>
self::DISPLAY, 'status' => self::DISABLE])->count();

        return $queue;
    }

//отримання часу початку та часу завершення
експерименту
    public function getTimeQueue()
    {
        $result =
ExperimentQueue::where(['experiment_id' =>
self::DISTANCE, 'status' => self::ACTIVE])->first();

        $queue['distance'] = $this->checkRecord($result);

        $result =
ExperimentQueue::where(['experiment_id' => self::SERVO,
'status' => self::ACTIVE])->first();
        $queue['servo'] = $this->checkRecord($result);
    }

```

```

        $result =
ExperimentQueue::where(['experiment_id' => self::LED,
'status' => self::ACTIVE])->first();
        $queue['led'] = $this->checkRecord($result);

        $result =
ExperimentQueue::where(['experiment_id' =>
self::DISPLAY, 'status' => self::ACTIVE])->first();
        $queue['display'] = $this-
>checkRecord($result);

        // var_dump($queue);

        return $queue;
    }

//форматування часу
    public function checkRecord($model)
    {
        if ($model) {
            $result['start'] = $model->start-
>format('Y-m-d H:i:s');
            $result['end'] = $model->end->format('Y-m-d
H:i:s');

            return $result;
        }

        $result['start'] = null;
        $result['end'] = null;

        return $result;
    }
}

```

ДОДАТОК Б
Слайди презентації



Рисунок Б.1 – Слайд 1

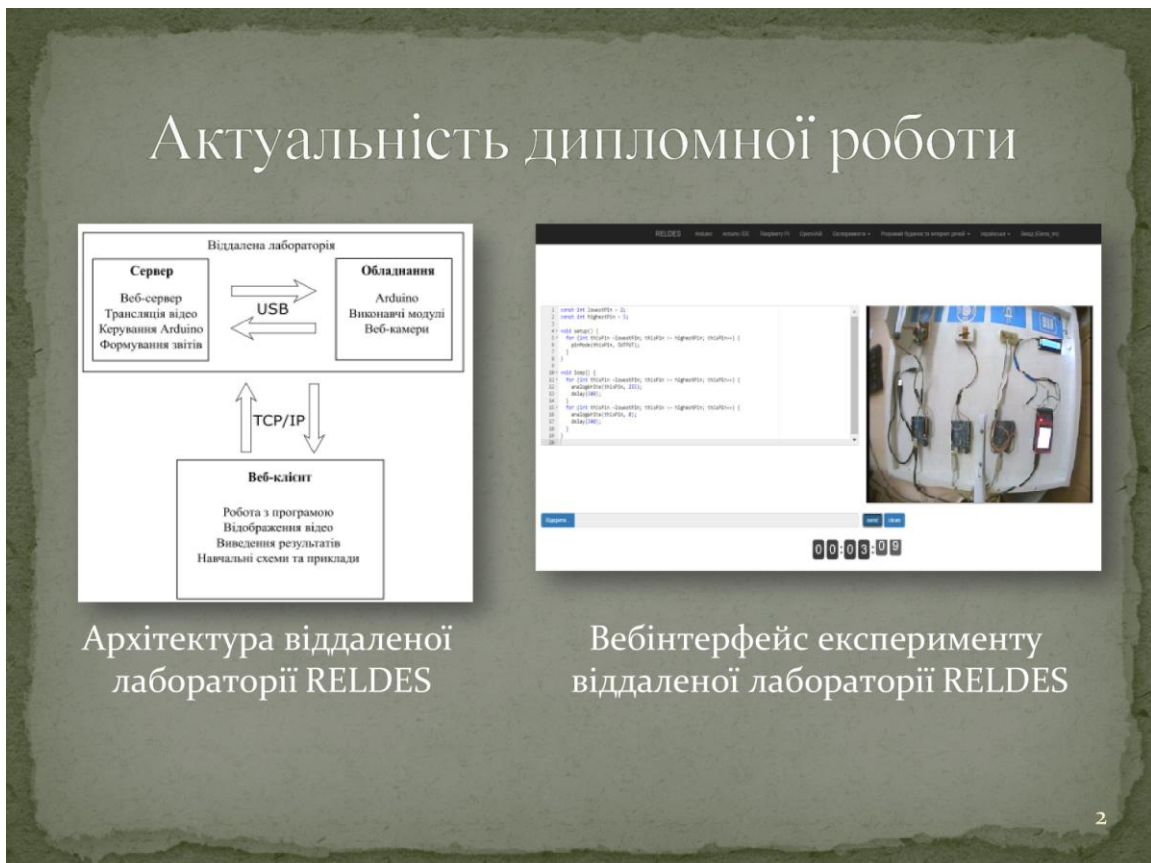


Рисунок Б.2 – Слайд 2

Мета та задачі дипломної роботи

Метою роботи є розробка програмного забезпечення управління чергою користувачів для зручного виконання експериментів віддаленої лабораторії REIDES.

Задачі дипломної роботи:

- виконати аналіз вимог та розробку архітектури програмного забезпечення;
- обрати інструментарій розробки програми;
- організувати спільне середовище розробки та середовище віртуалізації;
- виконати програмну реалізацію програмного застосунку;
- розробити програмну документацію.

3

Рисунок Б.3 – Слайд 3

Аналіз вимог до ПЗ



4

Рисунок Б.4 – Слайд 4

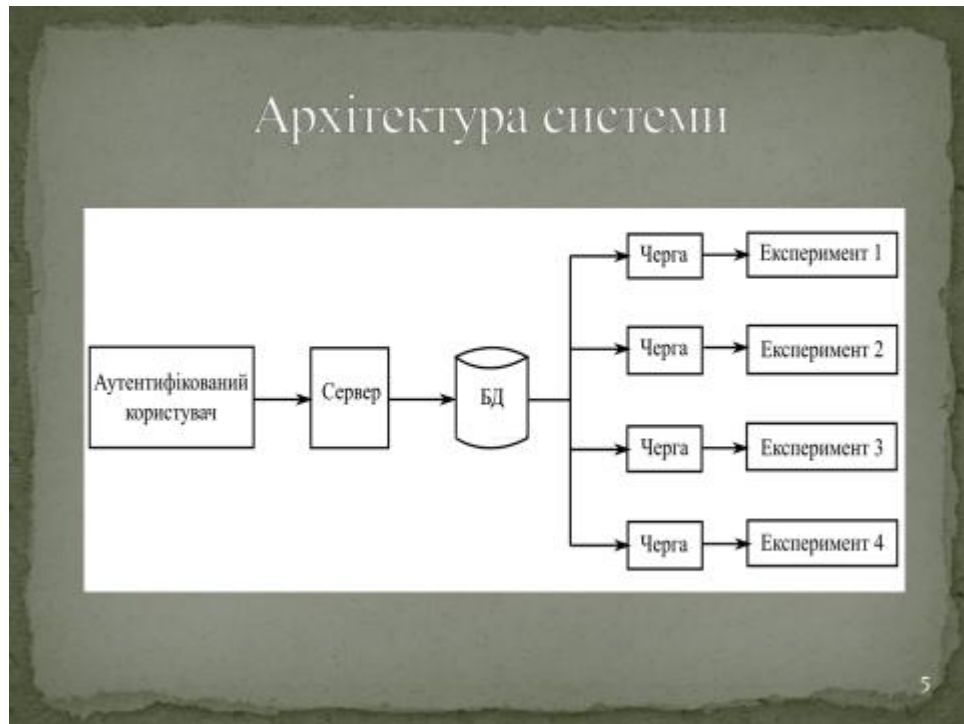


Рисунок Б.5 – Слайд 5

Аналіз мов програмування

Критерій порівняння	Ruby	Python	PHP
Велика кількість фреймворків для веброзробок	-	+	+
Відсутність необхідності підключення бібліотек	-	-	+
Розширена підтримка баз даних	-	+	+
Об'єктно-орієнтований підхід	+	+	+
Висока продуктивність	+	-	+

6

Рисунок Б.6 – Слайд 6

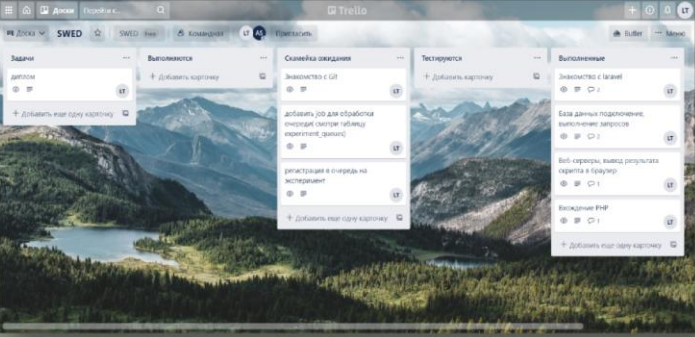
Аналіз фреймворків			
Критерій порівняння	Codeigniter	Yii	Laravel
Легкий у вивченні	+	-	+
Постійне оновлення	-	+	+
Швидка інтеграція з великою кількістю додаткових бібліотек	-	+	+
MVC-архітектура вебзастосунків	+	+	+
Вбудовані механізми для аутентифікації, авторизації, реєстрації користувачів	-	+	+

Рисунок Б.7 – Слайд 7

Організація спільного середовища розробки

Що треба зробити	В процесі виконання	Виконано
Завдання 4	Завдання 3	Завдання 1
Завдання 6		Завдання 2
Завдання 5		

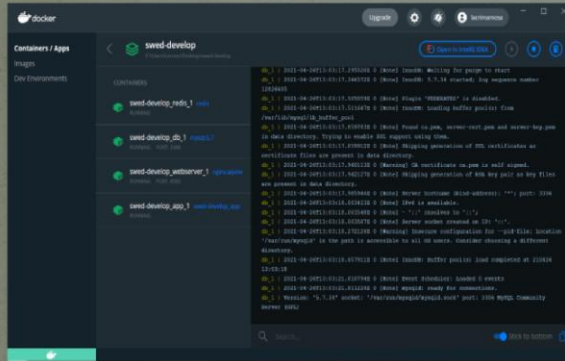
Схема класичної дошки Kanban



Інтерфейс системи управління проектами Trello

Рисунок Б.8 – Слайд 8

Розгортання проєкту в середовищі віртуалізації



Інтерфейс програми
Docker Desktop for Windows

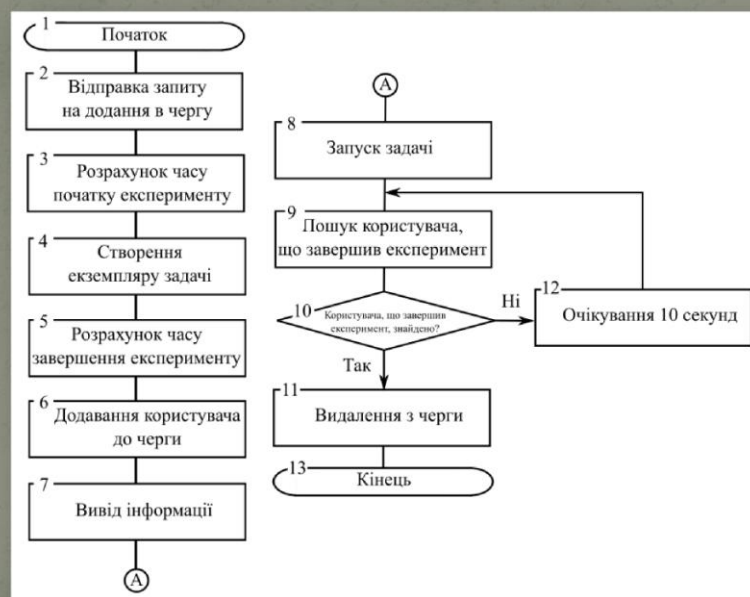
Проблема: середовище розробки значно відрізняється від середовища, де буде працювати кінцевий застосунок.

Вирішення: створення необхідного оточення для розробки застосунку за допомогою програмного забезпечення для створення ізолюваних середовищ розробки.

9

Рисунок Б.9 – Слайд 9

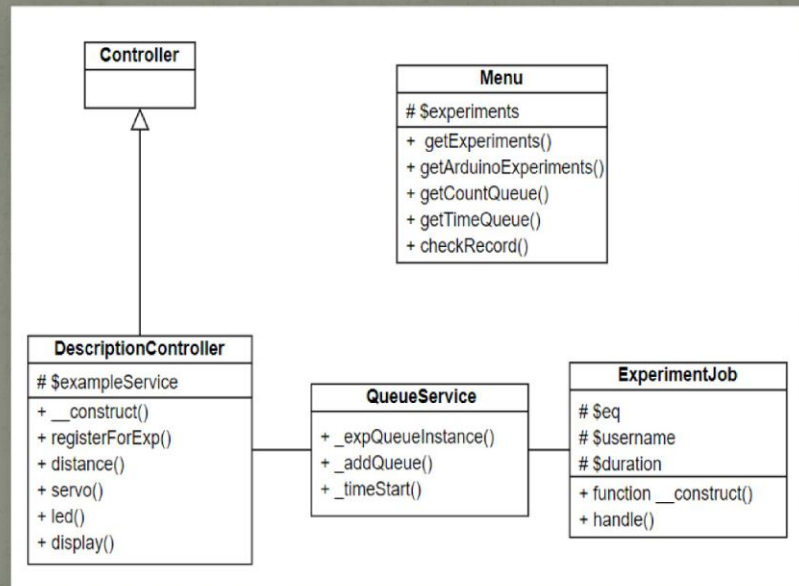
Алгоритм роботи ПЗ



10

Рисунок Б.10 – Слайд 10

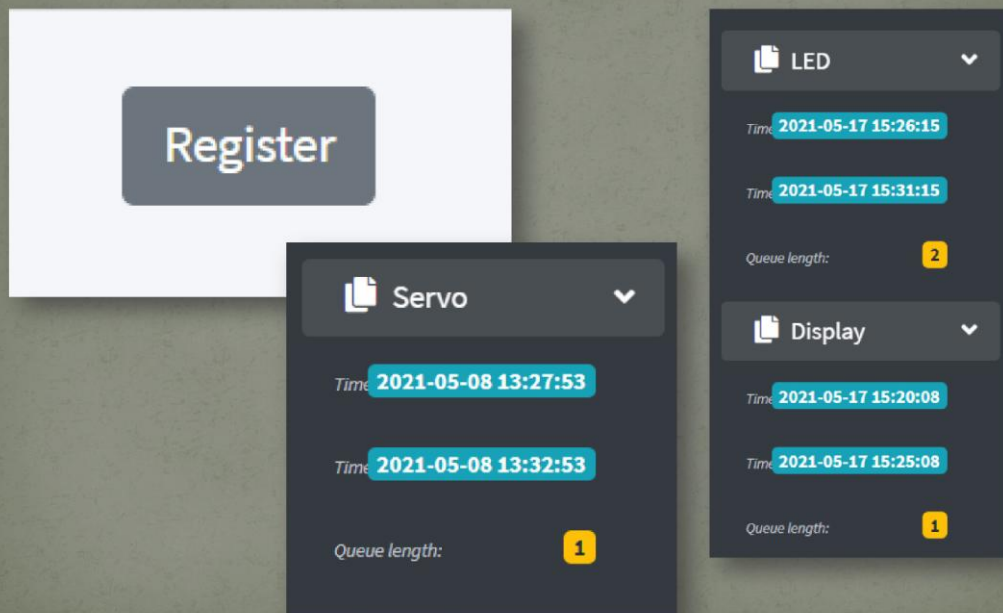
Діаграма класів ПЗ



11

Рисунок Б.11 – Слайд 11

Результати роботи



12

Рисунок Б.12 – Слайд 12

Висновки

В результаті виконання дипломної роботи було розроблено модуль черги для віддаленої лабораторії RELDES. Програма надає можливість користувачу займати чергу на експерименти та стежити за її станом.

В подальшій роботі планується продовжити модернізацію лабораторії RELDES, а саме створити таймери які будуть відраховувати час до початку та до кінця експерименту. Також планується розробити модуль який буде виводити інформацію з COM-порту плат Arduino на вебсторінку для експериментів: LED; Display; Distance; Servo.

Основні положення й результати роботи були представлені в тезах доповіді НПК Тиждень науки – 2021.

13

Рисунок Б.13 – Слайд 13

Дякую за увагу!

14

Рисунок Б.14 – Слайд 14