

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ
ПЛАТФОРМИ ДЛЯ РОЗРОБКИ ВІРТУАЛЬНОЇ ЛАБОРАТОРІЇ.
RESEARCH AND SOFTWARE IMPLEMENTATION OF A PLATFORM
FOR DEVELOPING A VIRTUAL LABORATORY

Виконав(ла): студентка 2 курсу, групи КНТ-113м

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

КАРНАУХ В.Д.

(ПРИЗВИЩЕ та ініціали)

Керівник ПАРХОМЕНКО А.В.

(ПРИЗВИЩЕ та ініціали)

Рецензент ЗЕЛЕНЬОВА І.Я.

(ПРИЗВИЩЕ та ініціали)

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

Факультет ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2024 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

КАРНАУХ Вероніки Дмитрівни

(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація платформи для розробки віртуальної лабораторії. Research and Software Implementation of a Platform for developing a Virtual Laboratory
керівник проєкту (роботи) к.т.н., доцент, ПАРХОМЕНКО Анжеліка Володимирівна

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

затверджені наказом закладу вищої освіти від “07” жовтня 2024 року № 398

2. Строк подання студентом проєкту (роботи) 20 грудня 2024 року

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

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження. 2. Проєктування програмного забезпечення. 3. Основні рішення щодо реалізації компонентів системи. 4. Керівництво програміста. 5. Керівництво оператора. Додатки.

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

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, 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 « 04 » вересня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	2 тиждень	Розділ 1
3	Матеріали і методи.	4 тиждень	Розділ 2
4	Розробка архітектури програми.	5-6 тижні	Розділ 2, 3
5	Розробка програми.	6-8 тижні	Розділ 3
6	Тестування та експериментальне дослідження програмного забезпечення.	9 тиждень	Розділ 4, 5
7	Оформлення пояснювальної записки та документів до неї	10 тиждень	Розділ 5 Додатки
8	Нормоконтроль та рецензування.	11 тиждень	
9	Захист роботи.	12 тиждень	

Студент(ка)

(підпис) Вероніка КАРНАУХ
(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Анжеліка ПАРХОМЕНКО
(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до випускної кваліфікаційної роботи магістра:
101 с., 9 табл., 65 рис., 3 дод., 45 джерел.

ARDUINO, DOCKER, ВІРТУАЛЬНА ЛАБОРАТОРІЯ, ЕКСПЕРИМЕНТ,
ФРЕЙМВОРК.

Об'єкт дослідження – процес розробки віртуальної лабораторії.

Предмет дослідження – методи та програмні засоби розробки віртуальної лабораторії.

Мета роботи – дослідження та програмна реалізація платформи для розробки віртуальної лабораторії для підвищення рівня автоматизації проектних процедур за допомогою сучасних технологій розробки програмного забезпечення.

Матеріали, методи та технічні засоби: мови програмування PHP, JavaScript; середовище розробки Visual Studio Code; система керування базами даних MySQL; фреймворки Laravel, Vue.js; платформа Wokwi; програмне забезпечення Docker; розподілена система контролю версій git; процедурне та об'єктно-орієнтоване програмування; персональний комп'ютер із процесором AMD Ryzen 7 7700 під управлінням операційної системи Ubuntu Mate 22.04.

Результати. Виконано розробку програмного забезпечення платформи, яка дозволяє створювати віртуальну лабораторію для дослідження вбудованих систем на платформі Arduino.

Висновки. Використання розробленої платформи спрощує та прискорює процес створення віртуальної лабораторії за рахунок інтеграції сучасних технологій розробки програмного забезпечення.

Галузь використання – розробка віртуальних та віддалених лабораторій для системи освіти.

ABSTRACT

Explanatory note to the final qualifying work of the master: 101 pages, 9 tables, 65 figures, 3 appendixes, 45 sources.

ARDUINO, DOCKER, VIRTUAL LABORATORY, EXPERIMENT, FRAMEWORK.

Object of study is the process of the development of a virtual laboratory.

Subject of study are methods and software tools for the development of a virtual laboratory.

The purpose of the work is research and software implementation of a platform for developing a virtual laboratory to increase the level of automation of design procedures using modern software development technologies.

Materials, methods, and tools: programming languages PHP and JavaScript; integrated development environment Visual Studio Code; relational database management system MySQL; frameworks Laravel and Vue.js; platform Wokwi; software Docker; distributed version control system git; procedural and object-oriented programming; personal computer with an AMD Athlon II X4 635 processor under the control of the Ubuntu Mate 22.04 operating system.

Results. The software platform has been developed, which allows to create a virtual laboratory for researching embedded systems on the Arduino platform.

Conclusions. The use of the developed platform simplifies and accelerates the process of creating a virtual laboratory by integrating modern software development technologies.

The field of use is the development of the virtual and remote laboratories for education.

ЗМІСТ

	С.
Перелік скорочень та умовних познач 8	8
Вступ 9	9
1 Аналіз проблеми та постановка завдань дослідження 10	10
1.1 Аналіз структурних та функціональних особливостей віддалених та віртуальних лабораторій 10	10
1.1.1 Огляд існуючих віртуальних та віддалених лабораторій.....11	11
1.1.2 Дослідження існуючих підходів до розробки віддалених та віртуальних лабораторій 15	15
1.2 Постановка завдань дипломної роботи 18	18
2 Проектування програмного забезпечення 20	20
2.1 Аналіз особливостей фреймворку 20	20
2.2 Вибір інструментарію розробки програмного забезпечення 22	22
2.2.1 Огляд особливостей мови програмування.....22	22
2.2.2 Огляд особливостей середовища розробки 23	23
2.2.3 Огляд особливостей системи управління базами даних 24	24
2.2.4 Огляд особливостей фреймворків 26	26
2.2.5 Огляд особливостей системи ведення документації 27	27
2.2.6 Огляд особливостей тестування 28	28
2.3 Моделювання програмного забезпечення 29	29
2.3.1 Діаграма активностей.....30	30
2.3.2 Діаграма прецедентів 33	33
2.4 Висновки за розділом 2.....36	36
3 Основні рішення щодо реалізації компонентів системи 37	37
3.1 Розширення функціоналу 37	37
3.1.1 Ідея експерименту 37	37
3.1.2 Програмна реалізація 38	38
3.1.3 Тестування експерименту 44	44
3.2 Висновки за розділом 3 46	46
4 Керівництво програміста 47	47

4.1 Призначення системи.....	47
4.2 Підготовка до розробки	47
4.2.1 Вибір операційної системи.....	47
4.2.2 Встановлення Docker Engine	48
4.2.3 Налаштування Git проєкту	51
4.3 Запуск системи.....	65
4.4 Розширення функціоналу фреймворку	65
4.4.1 Робота із експериментами	65
4.4.2 Створення шаблонів	66
4.4.3 Створення компонентів	67
4.4.4 Тестування.....	67
4.5 Характеристики системи	68
4.6 Початкові та вихідні дані.....	69
4.7 Висновки за розділом 4.....	69
5 Керівництво оператора	70
5.1 Призначення системи.....	70
5.2 Функції системи.....	70
5.3 Умови виконання системи.....	70
5.3.1 Апаратні вимоги системи	70
5.3.2 Вимоги до користувача.....	70
5.4 Взаємодія із системою.....	71
5.4.1 Взаємодія із LED	72
5.4.2 Взаємодія із LED Button	74
5.5 Висновки за розділом 5.....	75
Висновки.....	76
Перелік джерел посилання	77
Додаток А Технічне завдання	82
Додаток Б Текст програми.....	87
Додаток В Слайди презентації	94

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API	– Application Programming Interface;
HTML	– Hypertext Markup Language;
HTTP	– Hypertext Transfer Protocol;
IDE	– Integrated Development Environment;
OAS	– OpenAPI Specification;
RL	– Remote Laboratory;
SSH	– Secure Shell;
VL	– Virtual Laboratory;
БД	– база даних;
ВВЛ	– віддалені та віртуальні лабораторії;
ОС	– операційна система;
ПЗ	– програмне забезпечення;
СВД	– система ведення документації;
СКБД	– система керування базами даних.

ВСТУП

Використання різноманітних віддалених та віртуальних лабораторій (ВВЛ) для навчальних та наукових цілей стає повсякденним у сучасному світі, що зумовлено безупинним розвитком технологій онлайн-інженерії. Через те, що сфери досліджень постійно розширюються, розробку нових лабораторій можна назвати нескінченним процесом.

На сьогоднішній день немає єдиних стандартів і підходів у сфері розробки ВВЛ, тож різні розробники по-різному реалізують власні рішення щодо організації взаємодії користувача з набором експериментів. При цьому, створення ВВЛ з нуля є дуже складним завданням, що вимагає чимало фінансових та часових витрат. Тому подальші дослідження щодо впровадження нових методів та засобів для підвищення ефективності процесу розробки ВВЛ тривають.

Одним із прикладів реалізації платформи для розробки віртуальної лабораторії є фреймворк [1], за допомогою якого розробник може програмно створювати нові експерименти, а користувач – взаємодіяти з ними за допомогою веббраузера. Проте створення будь-якого нового експерименту, навіть якщо він використовує ті ж самі елементи, які вже має один із існуючих експериментів, не було достатньо автоматизовано, що призводило до повільного розвитку системи. Тому тема роботи є актуальною та спрямована на подальше дослідження та модифікацію фреймворку для створення віртуальної лабораторії.

Метою даної роботи є дослідження та програмна реалізація платформи для розробки віртуальної лабораторії для підвищення рівня автоматизації проектних процедур за допомогою сучасних технологій розробки програмного забезпечення.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

У даному розділі розглянуто існуючі приклади вирішення завдання створення віртуальних та віддалених лабораторій, що реалізують потребу роботи із експериментами в сфері освіти.

1.1 Аналіз структурних та функціональних особливостей віддалених та віртуальних лабораторій

У сучасному світі все більше такі процеси, як робота й освіта, переходять на віддалений режим, що дозволяє отримувати різноманітні знання та досвід дистанційно. У тому числі, наразі розробляється велика кількість лабораторій, що надають можливість цілодобово проводити експерименти з різних галузей наук [1].

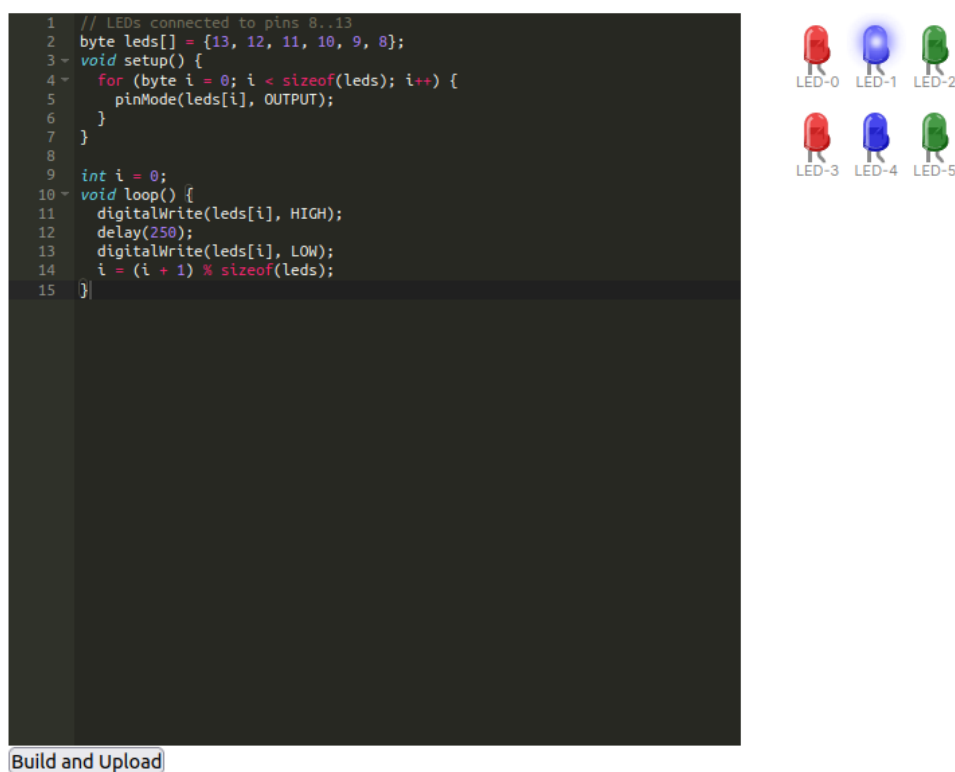
Якщо взяти до уваги, що не всі мають можливість організовувати кожен варіацію відповідного експерименту в університеті, школі або навіть на кухні, поширення віртуальних та віддалених лабораторій буде надавати доступ усім зацікавленим особам у будь-якій точці світу.

Віртуальні та віддалені лабораторії надають відповідно можливість використання певних симуляторів, що повторюють роботу реальних експериментів, і можливість керування реальним обладнанням та слідкуванням за ходом експерименту в реальному часі. Обидва типи лабораторій мають свої переваги та широко використовуються.

Було спроектовано та створено безліч варіантів віртуальних, віддалених та гібридних лабораторій, кожен з яких має свої певні особливості та включають різні технології. Багато дослідників використовує такі лабораторії повсякденно, що стимулює розвиток цієї сфери [1].

1.1.1 Огляд існуючих віртуальних та віддалених лабораторій

Фреймворк [2] є базовою системою із низьким рівнем автоматизації, що надає користувачу можливість роботи із існуючими експериментами на платі Arduino або програмно створювати власні та додавати їх до проєкту за допомогою Git. Також використовуються такі технології, як контейнери Docker, фреймворки Laravel і Next.js, СКБД MySQL, бібліотека Wokwi та інші. Приклад сторінки експерименту наведено на рис. 1.1.



OUTPUT

Status: Success

Sketch uses 1028 bytes (3%) of program storage space. Maximum is 32256 bytes. Global variables use 17 bytes (0%) of dynamic memory, leaving 2031 bytes for local variables. Maximum is 2048 bytes. [92mUsed platform[0m [92mVersion[0m

Рисунок 1.1 – Сторінка експерименту із 6 світлодіодами [2]

Запропонована віддалена лабораторія (RL) у дисциплінах STEM [3] представляє собою пов'язані між собою елементи схеми: навчальну платформу для контролю учасників; систему резервування ресурсів експериментів відповідно до ролі користувача (викладач або студент) на певний проміжок

часу; реальні лабораторії із необхідним обладнанням, якими можна скористатися за допомогою віддалених керувальних пристроїв (рис. 1.2).

Окрім ролі викладача та ролі студента, передбачено ролі системного адміністратора, що відповідає за роботу системи, створеної на фреймворку, та адміністратору віддаленої лабораторії, що відповідає за обладнання та експерименти у лабораторіях [3].

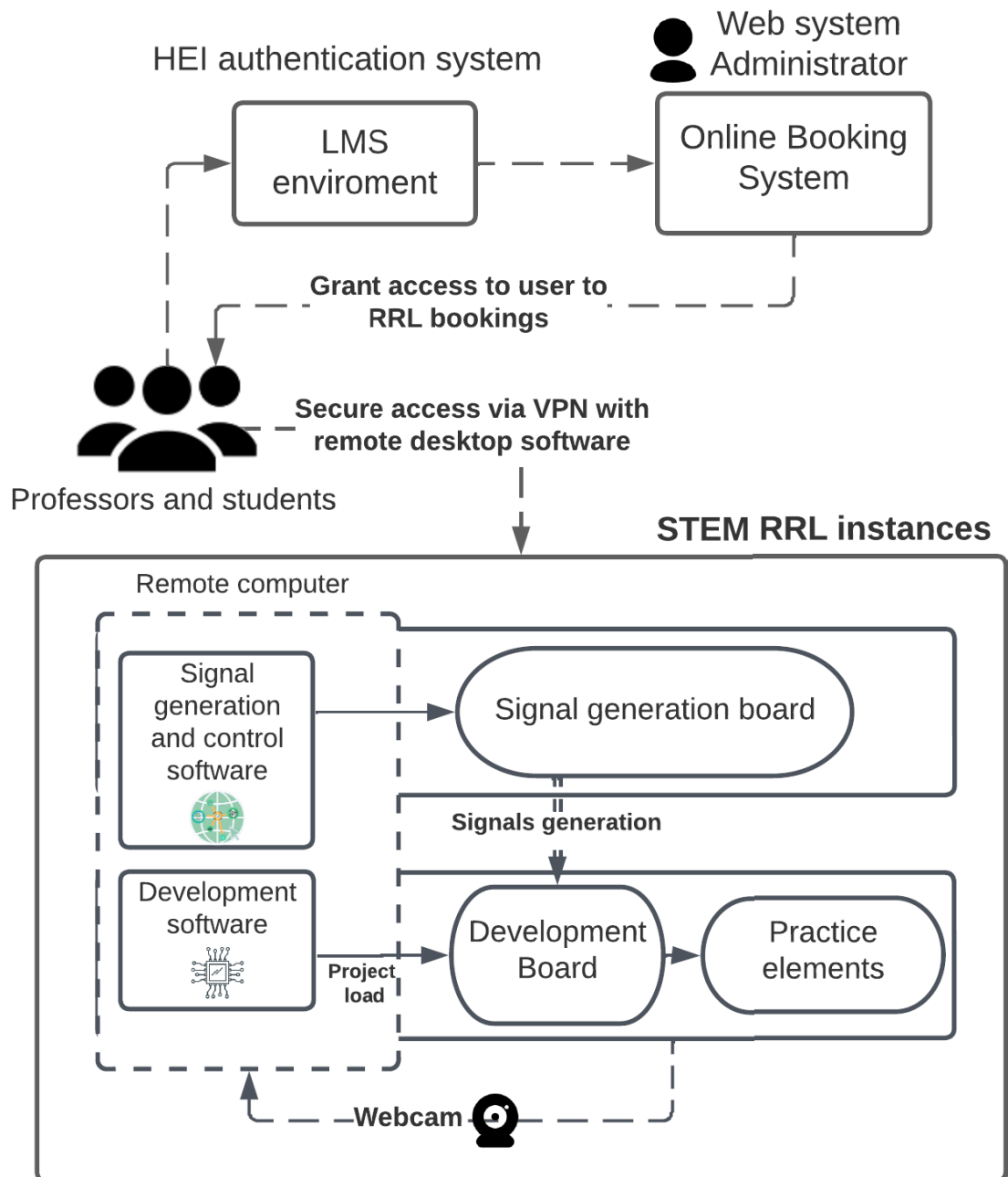


Рисунок 1.2 – Структура фреймворку віддаленої лабораторії [3]

Запропонована віддалена лабораторія (RL) [4] складається із трьох шарів: графічного інтерфейсу (на базі HTML5 та Web 2.0 технологій), ядра платформи (що складається із модулів, які об'єднуються за допомогою Apache та Node.js, ядра Cloud9 IDE, рушію MediaWiki, модулю передачі даних у режимі реального часу і модулю передачі відео у режимі реального часу) та системного шару (у якому Docker гарантує ідентичне встановлення та виконання ряду драйверів на різних пристроях). Більш детально структуру фреймворку наведено на рис. 1.3.

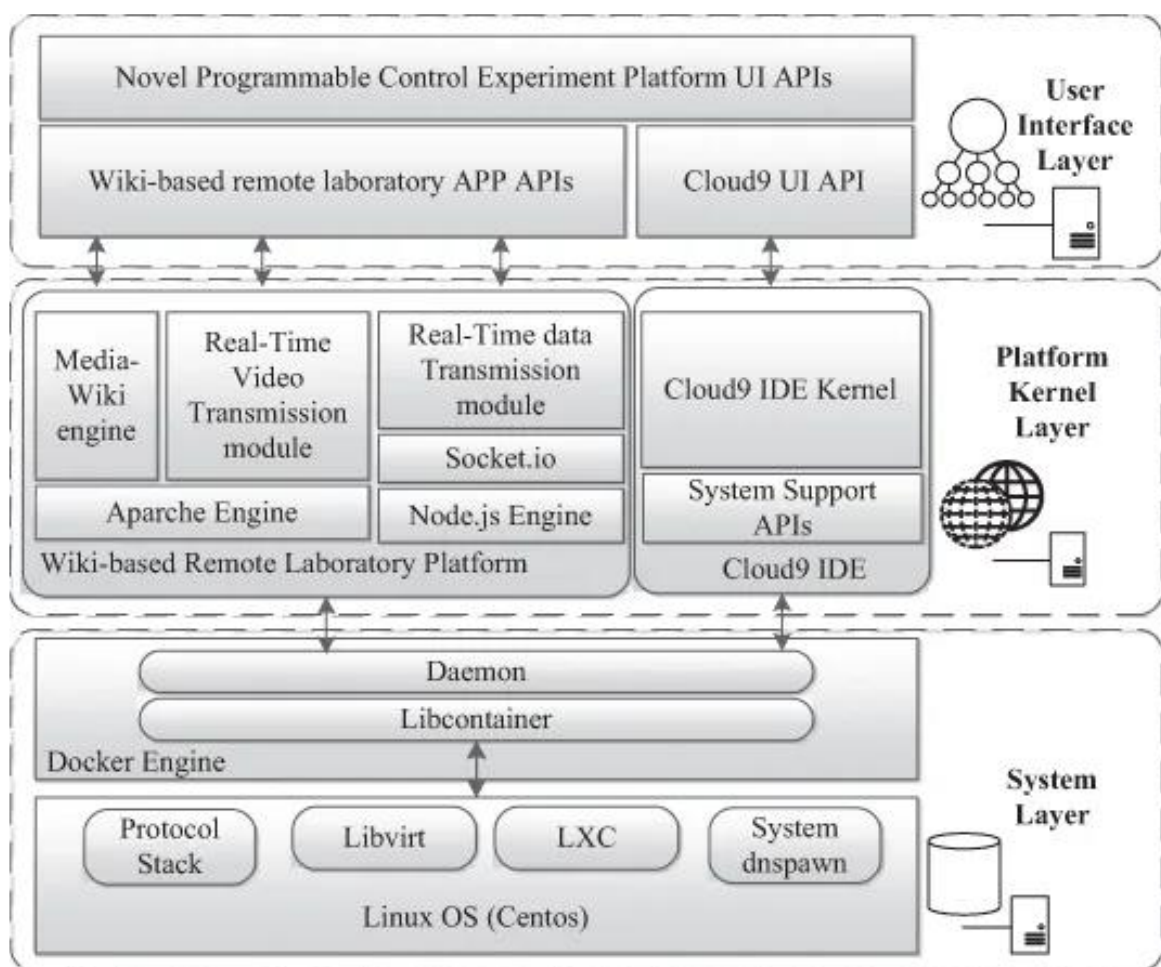


Рисунок 1.3 – Структура RL для програмування експериментів [4]

Онлайн-платформа Tinkercad [5] є середовищем розробки (VL), що дозволяє створювати 3D-моделі й електричні схеми із представлених елементів – світлодіодів, кнопок, датчиків та інших – та проводити симуляції із ними за допомогою написання коду або використання блоків (рис 1.4).

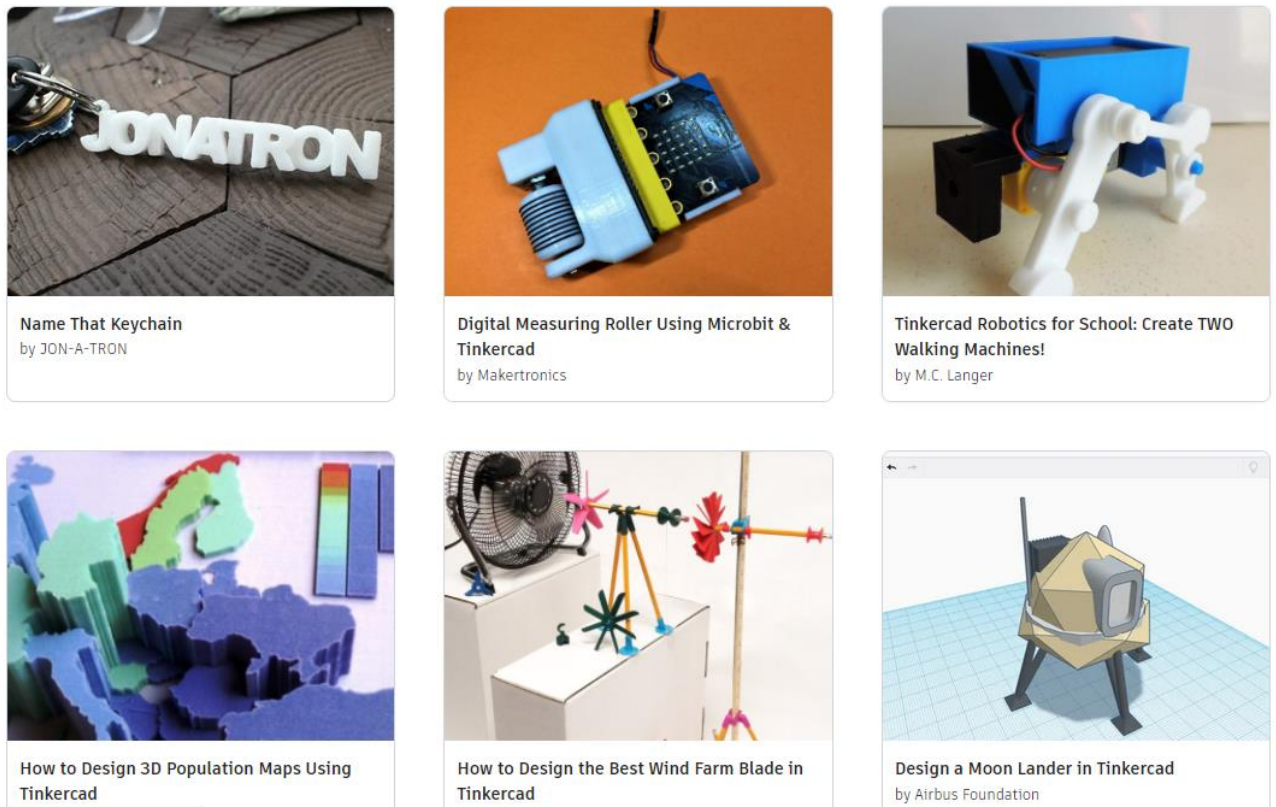


Рисунок 1.4 – Проєкти Tinkercad [5]

Лабораторія GOLDi [6] представляє собою гібридну лабораторію (тобто комбінацію RL та VL) із набором експериментів (у тому числі: керування ліфтом, керування рівнем води, проходження лабіринту, програмування плати), що у випадку віддаленого експерименту поєднує користувача із лабораторним обладнанням за допомогою інтернету та керує ним за допомогою спеціальних сервісів, а у випадку віртуального експерименту симулює дії користувача у спеціальному середовищі. Загальну структуру роботи лабораторії GOLDi наведено на рис. 1.5.

У таблиці 1.1 наведено порівняння вже існуючих програмних засобів та системи [2]-[6] за типом лабораторії (VL чи RL) і відповідністю можливостей системи із очікуваними (а саме можливість створення користувачем експериментів і можливість зробити це із високим рівнем автоматизації).

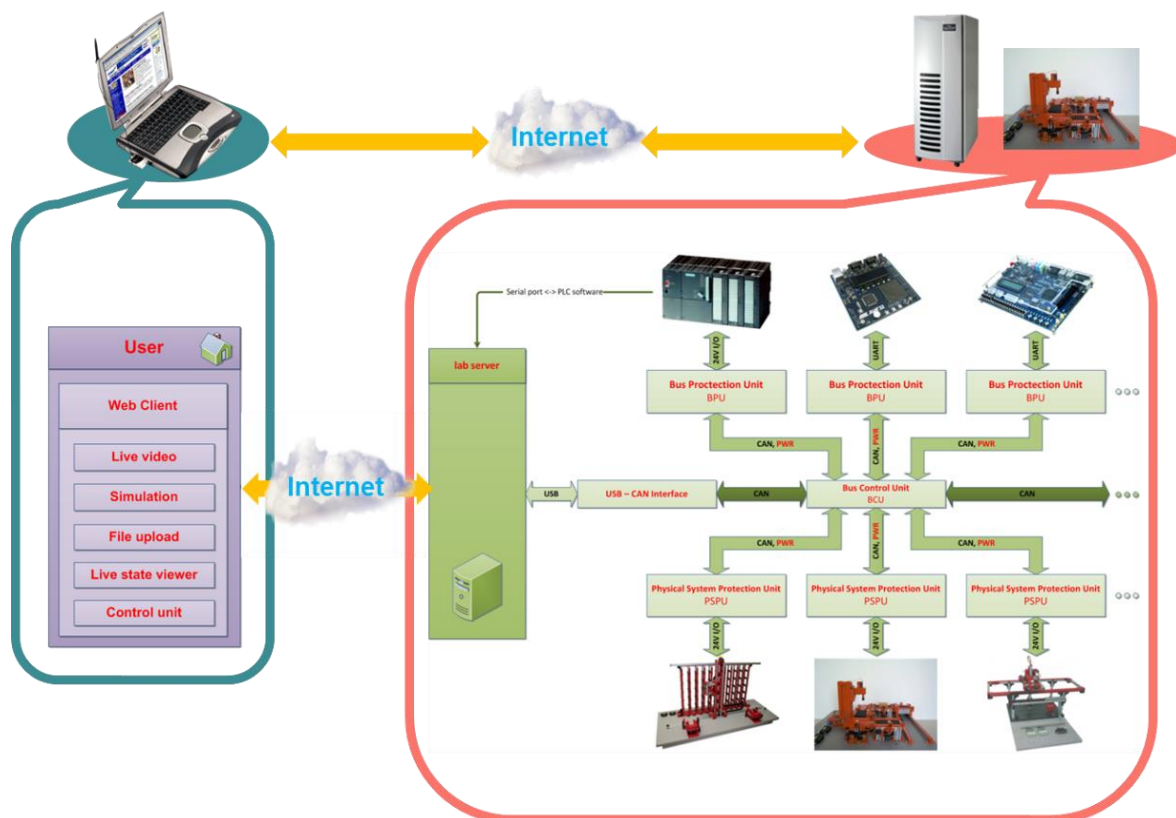


Рисунок 1.5 – Структура GOLDi [6]

Таблиця 1.1 – Порівняння програмних продуктів [1]-[6]

Лабораторії	VL	RL	Доступність	Створення експериментів	Автоматичне створення експериментів
[2]	+	планується	+	+	-
[3]	-	+	+-	-	-
[4]	-	+	+-	-	-
[5]	+	-	+	+	+
[6]	+	+	+	-	+-

1.1.2 Дослідження існуючих підходів до розробки віддалених та віртуальних лабораторій

Із самого початку створення ВВЛ розробники намагалися визначити парадигми, які б допомогли спростити розгортання онлайн лабораторій [7]-[8].

Розробники iLab Shared Architecture, WebLab-Deusto [9], Lab-share Saharan та інші зробили великий внесок до цієї галузі досліджень.

Багато підходів та інструментів було запропоновано для створення ВВЛ, зокрема методологія повторного використання [10], системи управління віддаленими лабораторіями [11]-[12], системи управління змістом [13] та фреймворки для інтеграції експериментів до лабораторій або розгортання таких систем [14]-[15].

У статті [16] автори представили інфраструктуру на основі хмарних технологій, що запускає симуляцію забезпечення у циклі відповідно до інструкції, встановленої на симуляторі. При використанні цієї платформи студенти можуть розроблювати та реалізовувати коди програм цільових архітектур і прототипів, а також налагоджувати проєкти на віртуальній платформі.

Автори статті [4] зазначили, що розробка хмарних та мобільних рішень для гнучкого розгортання та підтримки ВВЛ все ще є важливим завданням, і запропонували декілька варіантів: гнучкий фреймворк для інтеграції офлайн експериментів до систем віддалених лабораторій, фреймворк віддаленої лабораторії у якості сервісу на основі хмарних технологій, мобільно-оптимізований застосунок для віддаленої лабораторії, програмована онлайн платформа для розробки віддаленого програмованого керування експериментами.

У таблиці 1.2 наведено порівняння особливостей підходів до розробки ВВЛ, що включає до себе використані технології, отримані переваги та недоліки.

Таблиця 1.2 – Порівняння підходів до розробки ВВЛ [4], [7]-[16]

Джерело	Технології	Архітектура	Переваги	Недоліки
1	2	3	4	5
[4]	Вебтехнології, технології IoT	Модульна архітектура	Гнучкість, масштабованість	Високі витрати на розробку та підтримку
[7]	Хмарні обчислення	Модульна архітектура	Модульність, масштабованість, доступність	Залежність від стабільного інтернет-з'єднання
[8]	Хмарні обчислення, віртуалізація	Модульна архітектура	Спрощення розробки та впровадження	Потенційні проблеми з безпекою
[9]	Вебтехнології	Мікросервіси	Документація та приклади, доступність	Вимоги до технічних знань

Продовження таблиці 1.2

1	2	3	4	5
[11]	Вебтехнології, шаблони проектування	Централізована система	Спрощення розробки, легкість управління	Обмеженість у функціоналі
[12]	Хмарні обчислення	Розподілена система	Спільна розробка, гнучкість	Залежність від стабільного інтернет-з'єднання
[13]	Вебтехнології, CMS платформи	Централізована система	Легкість управління та використання	Обмеженість у налаштуваннях
[14]	Технології IoT	Розподілені та вбудовані системи	Інтеграція з IoT пристроями	Високі витрати на впровадження
[15]	Вебтехнології, Node-RED	Інтегрована система	Простота інтеграції та налаштування	Можливі проблеми з масштабуванням
[16]	Хмарні технології	Хмарна платформа	Масштабованість, доступність	Залежність від стабільного інтернет-з'єднання

При аналізі таблиці 1.2 можна зробити висновки:

- ВВЛ мають бути доступними для використання та розуміння як новачками, так і експертами;
- ВВЛ мають надавати широкий функціонал для роботи та можливість інтеграції з іншими системами;
- ВВЛ мають надавати користувачам документацію щодо роботи системи та достатню кількість прикладів, що демонструють базові можливості системи;
- розробники мають намагатися використовувати методи та підходи для зменшення часових і фінансових витрат необхідних для розгортання ВВЛ;
- розробники мають намагатися робити функціонал ВВЛ можливим для використання у випадку нестабільного інтернет-з'єднання.

У таблиці 1.3 наведено базові технології для розробки ВВЛ.

За висновками щодо функціоналу та технологій було розроблено технічне завдання (Додаток А).

Таблиця 1.3 – Базові технології розробки ВВЛ [3]-[4], [11]-[16]

	Джерела	Технології
1	Implementation of reusable solutions for remote laboratory development (A. Parkhomenko et al.) [10]	Методологія повторного використання
2	The remote laboratory management system (RLMS) pattern (L. F. Rivera) [11]	RLMS
3	Towards a microRLMS approach for shared development of remote laboratories (P. Orduna et al.) [12]	
4	Content management system for remote laboratory development (O. Tereshchenko et al.) [13]	Система керування вмістом
5	Design and development of remote laboratories with Internet of Things setting (M. K. A. Azad) [14]	Фреймворки
6	Development of a remote industrial laboratory for automatic control based on node-RED (M. Dominguez et al.) [15]	
7	A flexible framework for the deployment of STEM real remote laboratories in digital electronics and control systems (R. G. Vera et al.) [3]	
8	Cloud based remote virtual prototyping platform for embedded control applications (S. Werner et al.) [16]	Хмарна інфраструктура
9	Development of a Remote Laboratory for Engineering Education (N. Wang et al.) [4]	Хмарні та мобільні технології

1.2 Постановка завдань дипломної роботи

Як показали проведені дослідження, усі існуючі реалізації ВВЛ досі не підлягають єдиним стандартам і підходам. Через це розробка загальних архітектур та моделей для забезпечення ефективної розробки онлайн лабораторій, їх розгортання, використання, конфігурації та підтримки продовжується.

Аналіз існуючих віддалених та віртуальних лабораторій та їх можливостей дозволяє сформульовано основні вимоги до розробки ВВЛ із точки зору користувачів та розробників.

Огляд технологій, що використовуються для розробки різних ВВЛ, свідчить про проблематику в цій галузі та необхідність подальших досліджень. Тому тема роботи є актуальною.

Мета роботи – дослідження та програмна реалізація платформи для розробки віртуальної лабораторії для підвищення рівня автоматизації проектних процедур за допомогою сучасних технологій розробки програмного забезпечення.

Для досягнення поставленої мети необхідно розв'язати наступні завдання:

- дослідити структурні та функціональні особливості прототипу платформи;
- виконати аналіз та вибір інструментарію розробки;
- виконати моделювання системи;
- розробити методику створення віртуальної лабораторії;
- виконати практичну реалізацію віртуальної лабораторії з використанням модифікованої платформи.

2 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У даному розділі було розглянуто особливості програмних засобів для розробки удосконаленої платформи для створення віртуальної лабораторії, а також основні рішення щодо логічної структури розроблюваної програми.

2.1 Аналіз особливостей фреймворку

Фреймворк [2] використовує контейнери Docker, які підключено до мережі `public_net`, що дозволяє їм спілкуватися один із одним за допомогою імені або IP.

Загальну архітектуру платформи [1] наведено на рис. 2.1. У таблиці 2.1 наведено опис модулів платформи.

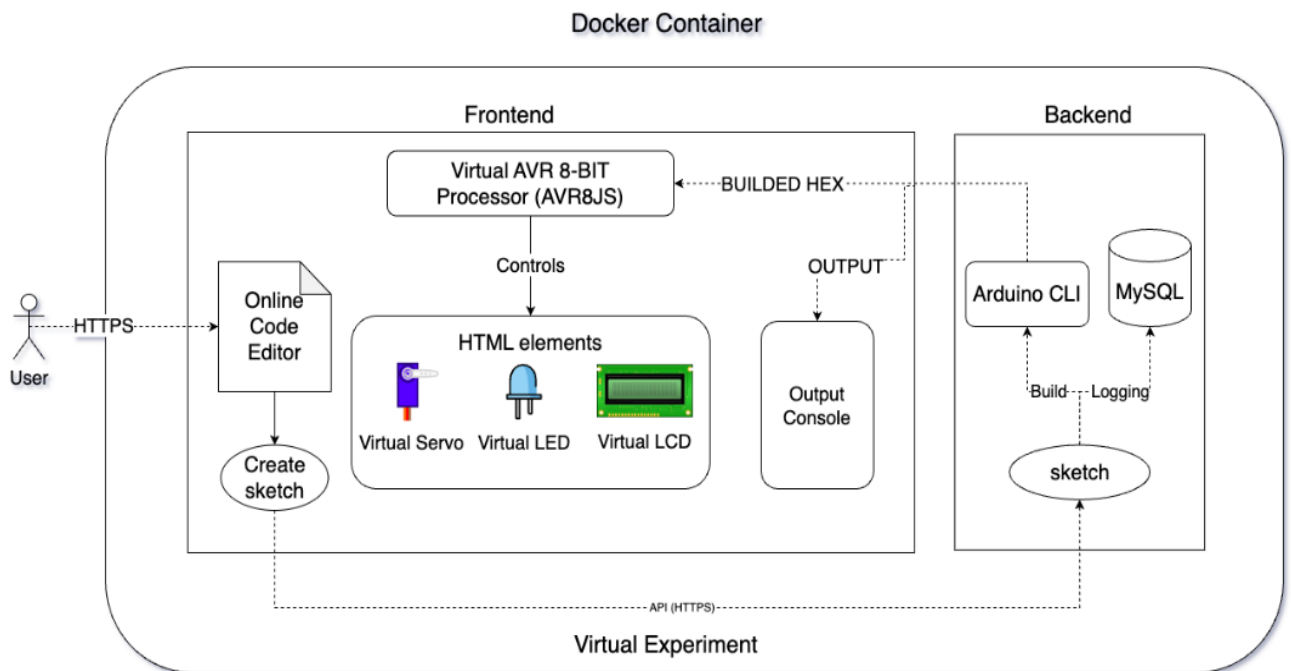


Рисунок 2.1 – Архітектура платформи [1]

Таблиця 2.1 – Модулі платформи

	Модулі	Функції
1	Frontend (Next.js)	HTTP запити до Backend для отримання JSON даних, потрібних для візуалізації користувацького інтерфейсу
		обробка даних і динамічне відображення інтерфейсу із HTML коду
2	Backend (Laravel API)	отримання запитів від Frontend та відправлення JSON відповідей до Frontend
		оброблення запитів через виконання необхідного PHP коду і звернення до MySQL бази даних
3	База даних (MySQL)	зберігання всіх даних, що використовує вебзастосунок
		витягання, додавання, оновлення або видалення даних за допомогою SQL запитів від Backend

При запуску серверу фреймворку [2] використовуються наступні сервіси:

а) nginx – «client-api-nginx-1»:

1) відкриває порти 8080:80 та 4430:443 для доступу з хост-машини;

2) монтує томи, де мапується локальний каталог у контейнері із кодом бекенду до контейнеру Nginx, змінюється конфігураційний файл за замовчуванням на власний файл із налаштуванням маршрутизації;

3) підключений до мережі public_net;

б) php-fpm – «openlab-api»:

1) монтує томи, де мапується локальний каталог у контейнері із кодом бекенду, додається конфігурація для Xdebug (інструменту відлагодження PHP);

2) підключений до мережі public_net;

в) db – «openlab-db»:

1) додає можливість регулювання пріоритету процесів;

2) відкриває порт 3306 для доступу з хост-машини;

3) монтує том для зберігання бази даних;

4) підключений до мережі public_net.

2.2 Вибір інструментарію розробки програмного забезпечення

2.2.1 Огляд особливостей мови програмування

PHP (Hypertext Preprocessor) – це крос-платформна скриптова мова програмування високого рівня із підтримкою багатьох парадигм програмування (мультипарадигмальна мова): об'єктно-орієнтованої, імперативної та функціональної. Окрім цього, PHP має відкритий початковий код [17].

PHP є однією із найпоширеніших мов програмування для веброзробки – її скрипти зазвичай генерують увесь код на сервері, який потім видається клієнту. При цьому PHP містить команди із вбудованим кодом HTML і надає додаткові розширення для роботи з XML [17].

PHP підтримується у більшості операційних систем (ОС), що дозволяє не обмежувати вибір простору розробки. PHP підтримує велику кількість вбудованих функцій для роботи з базами даних (MySQL, PostgreSQL, Oracle), обробки файлів, маніпулювання рядками, роботою з масивами тощо [17].

JavaScript (JS) – це крос-платформна прототипна мова програмування, що використовує JIT (just-in-time) компіляцію та підтримує багато парадигм програмування: об'єктно-орієнтовану, імперативну та функціональну [18].

JavaScript є однією із найпоширеніших мов програмування для розробки сторінок динамічних та інтерактивних вебзастосунків. Синтаксис JavaScript відповідає спрощеному синтаксису мови Java. Динамічні можливості JavaScript включають: побудову об'єктів під час виконання, списки змінних параметрів, функціональні змінні, створення динамічного сценарію, інтроспекція об'єктів, а також відновлення вихідного коду [18].

У таблиці 2.2 наведено порівняння розповсюджених мов програмування (у даному випадку Python) із мовами, що було використано у фреймворку.

Таблиця 2.2 – Порівняння мов програмування [17]-[19]

	PHP	JavaScript	Python
Компільована/інтерпретована	I	I	I
Мультипарадигмальність	++	+++	+++
Крос-платформність	+	+	+
Типізація	Динамічна	Динамічна	Динамічна
Стандартизованість	+	+	-
Фреймворки	Laravel, Symfony	React, Angular, Vue	Django, Flask

2.2.2 Огляд особливостей середовища розробки

Visual Studio Code (VS Code) – це крос-платформний редактор коду з відкритим вихідним кодом. Він підтримує різні ОС (Windows, Linux, macOS) і багато мов програмування (C++, C#, Java, Python, PHP, Go, .NET), має вбудований Git [20].

При цьому, IDE (середовище розробки) надає велику кількість розширень для різних мов програмування та технологій (таких як Docker, MongoDB та інші). VS Code надає можливості автозаповнення, стилізації, підсвічування та рефакторингу коду [20]. Окрім цього, VS Code має невеликі вимоги до системи, що робить його доступним для більшості користувачів [20].

У таблиці 2.3 наведено порівняння популярних IDE (таких як PyCharm і PhpStorm) для програмування обраними мовами (PHP і JS) із VS Code, обраним для використання.

Таблиця 2.3 – Порівняння середовищ розробки [20]-[22]

	VS Code	PyCharm	PhpStorm
Багатоплатформність	+	+	+
Візуалізація даних	Плагіни	Плагіни	Плагіни
Інтерактивний режим	+	+	+
Безкоштовність	+	+ -	-
Підтримка PHP	+ -	-	+

2.2.3 Огляд особливостей системи управління базами даних

При виборі бази даних (БД) зазвичай використовують такі основні критерії, як висока надійність, легкість у розумінні та використанні, висока швидкість роботи та стандартизований доступ до даних, тому що це – один з основних компонентів програми, від якого буде залежити якість та комфорт не тільки у користуванні, а й у написанні програмного продукту.

MySQL – це вільна система керування реляційними базами даних (СКБД), що є однією із найпоширеніших СКБД через підтримку багато мов програмування. MySQL підтримує великі бази даних і необмежену кількість користувачів, що одночасно взаємодіють із БД, впроваджує високу швидкість виконання команд та ефективну систему безпеки [23].

У таблиці 2.4 наведено порівняння найбільш розповсюджених СКБД (MongoDB і PostgreSQL) і MySQL, що було обрано для розробки.

Таблиця 2.4 – Порівняння СКБД [23]-[25]

	MySQL	MongoDB	PostgreSQL
Тип	реляційна	документно-орієнтована	реляційна
Типи даних	дані SQL	неструктуровані дані	різні
Особливість	швидкість, низька затримка	гнучкі дані	складні запити та транзакції
Масштабованість	вертикальна і горизонтальна	горизонтальна	вертикальна
Безпека	+	+-	+
Застосування	вебзастосунки, електронна комерція	гнучкі структури даних, неструктуровані або напівструктуровані дані	реляційні дані, складні запити, звіти, аналітика

Кешування – це інформаційна технологія для тимчасового зберігання певних даних, що часто використовуються. Використання кешу прискорює роботу комп’ютерних та мобільних систем, оскільки спочатку відбувається звернення до обраних копій інформації, а не до всього об’єму БД, і за умови

знаходження цілі запиту процес пошуку завершується. Таким чином, кеш впливає на ефективність і продуктивність роботи, оптимізуючи такі сценарії, як завантаження вебсторінок і файлів, витягування даних із БД тощо. Проте важливим аспектом кешування є алгоритм видалення кешу, що зберігає старі дані або ті, що більш не використовуються, оскільки це також оптимізує процес роботи [26].

Деякі бази даних виконують власне кешування без додаткового налаштування [26]. Але тимчасові дані, що генеруються у процесі роботи та не зберігаються для подальшого використання у майбутньому, можуть бути потрібні знов та оброблені за допомогою створення кешу замість повторного їх створення. Одним із прикладом IMDB (in-memory database) є Redis, який працює на різних ОС (у тому числі інтегрується у Docker), надає ряд бібліотек для роботи різними мовами програмування, підтримує безліч типів структур даних та автовидалення даних із кешу, що довго не використовувалися [27].

У таблиці 2.5 наведено порівняння популярних IMDB (Memcached і Apache Ignite), чиї можливості відповідають завданню, та бази даних Redis, обраної для майбутнього інтегрування.

Таблиця 2.5 – Порівняння IMDB [27]-[29]

	Redis	Memcached	Apache Ignite
Типи даних	рядки, кеш, списки, множини	рядки	кеш, SQL таблиці
Запис даних на диск	+	-	+
Швидкість	+	++	+-
Кластеризація	+	-	+
Підтримка інтеграції			
Застосування	кешування, черги, аналітика, сесії	просте кешування	кешування, аналітика, обробка великих даних

2.2.4 Огляд особливостей фреймворків

Laravel — це PHP-фреймворк для розробки серверної частини вебзастосунків, заснований на архітектурі MVC (Model-View-Controller). Він забезпечує структурований підхід до створення вебзастосунків з можливостями роботи з базами даних, аутентифікацією, маршрутизацією та багатьма іншими важливими функціями [30].

У таблиці 2.6 наведено порівняння популярних фреймворків backend (Symfony, CodeIgnite і Yii), чийі можливості відповідають завданню, та Laravel, обраного для розробки.

Таблиця 2.6 – Порівняння засобів для backend [30]-[33]

	Laravel	Symfony	CodeIgniter	Yii
Документація	++	-	+	+
Модульність	++	+++	+	+++
Безечність	+	+	+-	+
Підтримка API	+	+	+-	+
Масштабованість	+	+	+-	+
Застосування	вебзастосунки, CMS, API, E-commerce	корпоративні системи, додатки	вебзастосунки	портали, додатки

React.js — це JavaScript-бібліотека для створення інтерфейсів користувача, заснований на компонентному та функціональному підходах. Компонентом React є JavaScript-функція, а синтакс називається JSX (JavaScript XML). При цьому React добре працює із сторінками HTML. React використовує багато сторонніх бібліотек, що підключаються за необхідності [34].

Vue.js — це JavaScript-фреймворк для створення динамічних вебзастосунків, заснований на декларативному стилі програмування та реактивному моделюванні. Основні компоненти проєктуються за допомогою шаблонів, схожих на HTML/CSS. Vue не вимагає встановлення додаткових модулів [35].

Було здійснено перехід від React.js до Vue.js, спричинений тим, що останній надає більш гнучку систему, яку легко та поступово можна інтегрувати в проєкт, а також має більш простий та зрозумілий арсенал компонентів [34]-[35].

Wokwi – це онлайн-симулятор, що дозволяє проводити онлайн симуляцію та дебаг схем на таких платформах, як Arduino, ESP32, STM32 тощо. Він також дозволяє під'єднувати отримане забезпечення до інтернет ресурсів за допомогою протоколів MQTT, HTTP, NTP та може інтегруватися до VS Code. Бібліотека Wokwi також надає елементи схем, які можна використовувати в HTML [36].

У таблиці 2.7 наведено порівняння засобів, що використовуються на frontend у завданні.

Таблиця 2.7 – Засоби для frontend [34]-[36]

	React.js	Vue.js	Wokwi
Тип	бібліотека	фреймворк	платформа
Документація	+	+	+-
Тестування	+	+	-
Гручкість	+	+	-
Масштабованість	+	+	-
Навички	JavaScript, JSX	JavaScript	електроніка
Застосування	вебзастосунки	додатки	симуляція електроніки

2.2.5 Огляд особливостей системи ведення документації

Для майбутньої підтримки системи буде використовуватися автоматичне ведення документації.

Наявність документації системи, її доступність і регулярне оновлення протягом всього періоду розробки та використання суттєво необхідно як для майбутніх зацікавлених осіб та нових учасників, так і для поточних розробників.

Система ведення документації (СВД) – це система чи бібліотека, що дозволяє автоматично генерувати технічну документацію проєкту. Одним із прикладів такої системи є OpenAPI Specification (OAS, раніше Swagger) – стандарт структури API мовою HTTP, що підходить для Laravel і використовується для спрощення розуміння можливостей системи, перетворення спеціальних коментарів коду у сторінки тощо [37].

2.2.6 Огляд особливостей тестування

Для майбутньої підтримки системи буде використовуватися тестування різними методологіями.

Існує 2 основні методології до розробки програмного забезпечення – TDD (Test-Driven Development) і BDD (Behavior-Driven Development), де на початку здійснюється підготовка до тестування, і лише потім – написання коду функціоналу, що допомагає розробнику сформулювати вимоги до системи, до яких вже буде створено код [38]-[39].

При використанні TDD і BDD розробник виконує наступний алгоритм дій у циклі [38]-[39]:

а) створення тесту:

1) TDD – автоматизований тест, що описує очікувану поведінку функціоналу;

2) BDD – тест поведінкового сценарію (Given-When-Then) використання системи за вимогами до неї;

б) створення мінімального коду функціоналу, що успішно проходить тест;

в) оптимізація та редагування коду без впливу на функціональність.

Підвиди TDD [40]:

а) ATDD (Acceptance Test-Driven Development) – тестування на рівні прийняття вимог:

1) учасники – всі стейкхолдери (аналітики, власники, менеджери, розробники, тестувальники, кінцеві користувачі тощо);

2) процес – всі тести пишуться на початку розробки;

3) автоматизація – опціональна;

б) UTDD (Unit Test-Driven Development) – тестування на рівні одиничних компонентів:

1) учасники – розробники та тестувальники;

2) процес – кожний наступний тест пишеться після успішного виконання попереднього;

3) автоматизація – обов’язкова.

У таблиці 2.8 наведено порівняння методологій тестування, що будуть використовуватися у системі.

Таблиця 2.8 – Порівняння методологій тестування [38]-[39]

	TDD	BDD
Переваги	стандартизація написання коду	спрощення комунікації між тезнічними і нетехнічними учасниками проєкту
	висока якість коду	тести описують частину документації
	швидке виявлення та виправлення помилок	зменшення непорозумінь між вимогами і реальним функціоналом
Недоліки	необхідність досвіду у написанні тестів	необхідність командної взаємодії
	потребує багато часу	потребує багато часу
	необхідність визначених вимог	складність опису сценаріїв

2.3 Моделювання програмного забезпечення

Розроблювана платформа буде мати однакову ціль і схожий принцип дії із фреймворком [2], проте буде його поліпшеною версією, що полегшує роботу розробника й автоматизує частину звичайних робіт, потрібних для створення та редагування експериментів за допомогою існуючих шаблонів і компонентів.

Було розроблено ряд моделей у вигляді UML-діаграм для відображення логічної структури програмного продукту.

2.3.1 Діаграма активностей

Потік подій для діаграми активностей фреймворку [2] (рис. 2.2).

Крок 1. Запускається сервер.

Крок 2. Користувач відкриває сайт (<http://localhost:3000/>).

Крок 3. Користувач обирає експеримент для роботи.

Крок 4. Користувач змінює код.

Крок 5. Користувач запускає експеримент.

Крок 6. Через інтерфейс дані (код) передаються до серверу.

Крок 7. Сервер обробляє запит, отриманий від інтерфейсу, та передає його до клієнту.

Крок 8. Клієнт виконує необхідні дії для отримання результату та передає відповідні команди до серверу,

Крок 9. Сервер передає результати до інтерфейсу.

Крок 10. Інтерфейс відображає результати (успішність білду та працюючий експеримент).

Крок 11. Користувач переглядає результати та взаємодіє із елементами експерименту.

Крок 12. Кінець потоку.

Потік подій для нової діаграми активностей платформи (рис. 2.3).

Крок 1. Запускається сервер.

Крок 2. Користувач відкриває сайт (якщо користувач – адміністратор, то перехід за посиланням <https://admin.openlab.org.ua/templates>; а якщо користувач – розробник, то перехід за <https://client.openlab.org.ua/dashboard>).

Крок 3. Якщо користувач – адміністратор, то перехід до кроку 4, інакше – до кроку 9.

Крок 4. Користувач обирає шаблон для роботи.

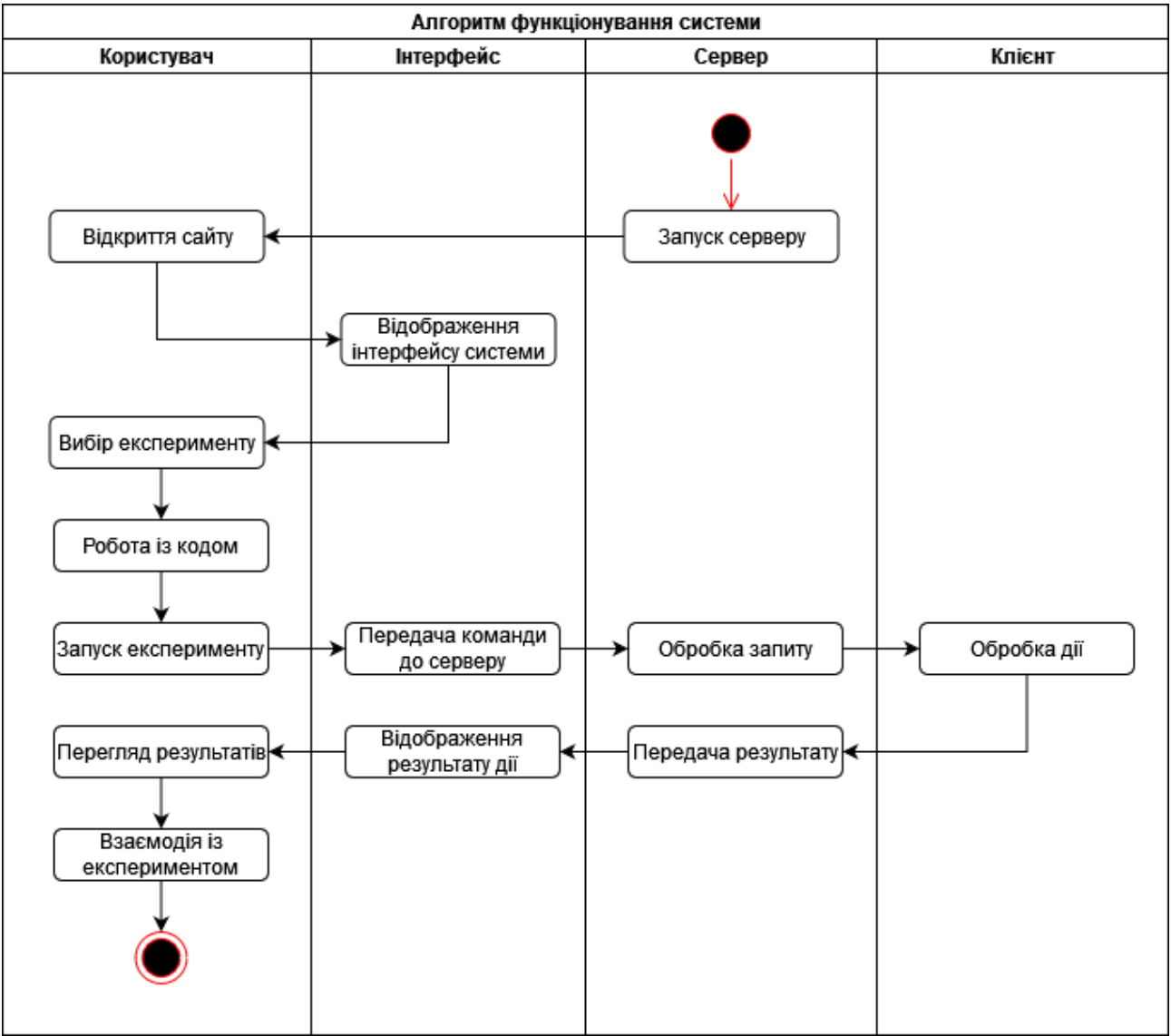


Рисунок 2.2 – Діаграма активностей [2]

- Крок 5. Користувач змінює шаблон (назву, компонент шаблону, картинку).
- Крок 6. Користувач обирає експеримент для роботи.
- Крок 7. Користувач змінює експеримент (назву, опис, картинку, шаблон).
- Крок 8. Кінець потоку.
- Крок 9. Користувач обирає експеримент для роботи.
- Крок 10. Користувач змінює код експерименту.
- Крок 11. Користувач запускає експеримент.

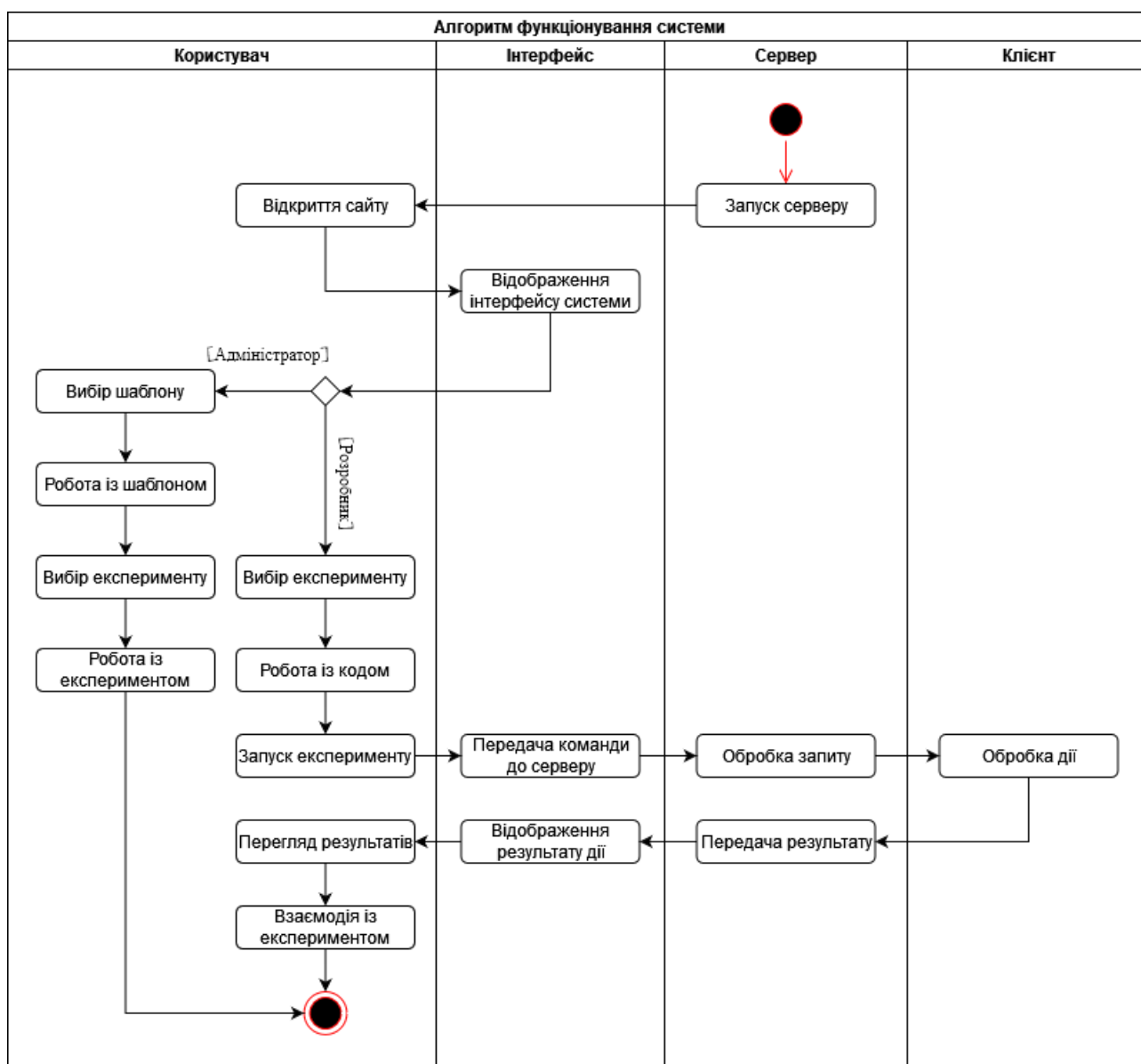


Рисунок 2.3 – Нова діаграма активностей

Крок 12. Через інтерфейс дані (код) передаються до серверу.

Крок 13. Сервер обробляє запит, отриманий від інтерфейсу, та передає його до клієнту.

Крок 14. Клієнт виконує необхідні дії для отримання результату та передає відповідні команди до серверу,

Крок 15. Сервер передає результати до інтерфейсу.

Крок 16. Інтерфейс відображає результати (успішність білду та працюючий експеримент).

Крок 17. Користувач переглядає результати та взаємодіє із елементами експерименту.

Крок 18. Кінець потоку.

При порівнянні двох потоків очевидно, що потік користувача у фреймворку [2] передбачає лише роботу із існуючими експериментами, тоді як потік користувача у платформі розширюється за допомогою розгалудження на потік адміністратора і потік розробника. Основною відмінністю цих потоків (у яких навіть відрізняються доменні імена) є різновид роботи з експериментами: адміністратор створює і редагує шаблони, із яких відповідно складаються експерименти, а розробник – використовує та тестує можливості наявних експериментів.

2.3.2 Діаграма прецедентів

Для фреймворку [2] було розроблено лише одну діаграму прецедентів.

Потік подій для розробленої діаграми прецедентів [2] (рис. 2.4).

Крок 1. Користувач запускає сервер, щоб отримати можливість взаємодіяти із функціоналом системи.

Крок 2. Користувач переглядає сторінки сайту: меню наявних експериментів і саме сторінки експериментів.

Крок 3. Користувач взаємодіє із експериментами: запускає білд наявного коду, переглядає результати білду та взаємодії із елементами керування експерименту (кнопками, перемикачами тощо). Це також може включати редагування усталеного коду для зміни умов експерименту.

Для розроблюваної платформи було розроблено дві діаграми прецедентів.

Потік подій для користувача-адміністратора для розробленої діаграми прецедентів платформи (рис. 2.5).

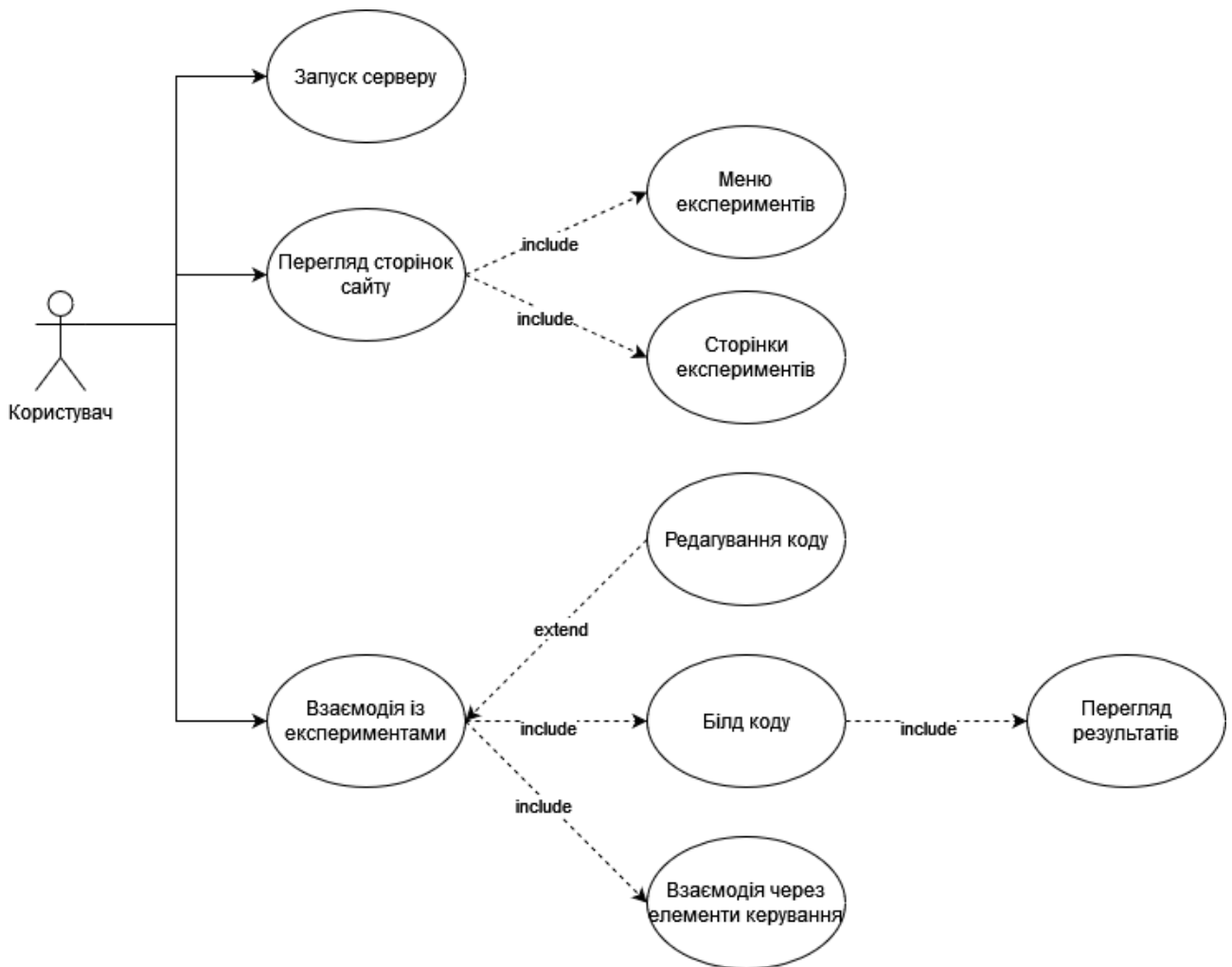


Рисунок 2.4 – Діаграма прецедентів [2]

Крок 1. Користувач запускає сервер, щоб отримати можливість взаємодіяти із функціоналом системи.

Крок 2. Користувач запускає сайт і переглядає його сторінки: меню наявних експериментів і сторінки експериментів, меню шаблонів і сторінки шаблонів.

Крок 3. Користувач взаємодіє із шаблонами: створює нові шаблони чи редагує у наявних назви, компоненти та картинки.

Крок 4. Користувач взаємодіє із експериментами: створює нові експерименти чи редагує у наявних назви, описи, картинки та шаблони.

Потік подій для користувача-розробника здебільшого повторює потіке подій для фреймворку [2] із деякими відмінностями.

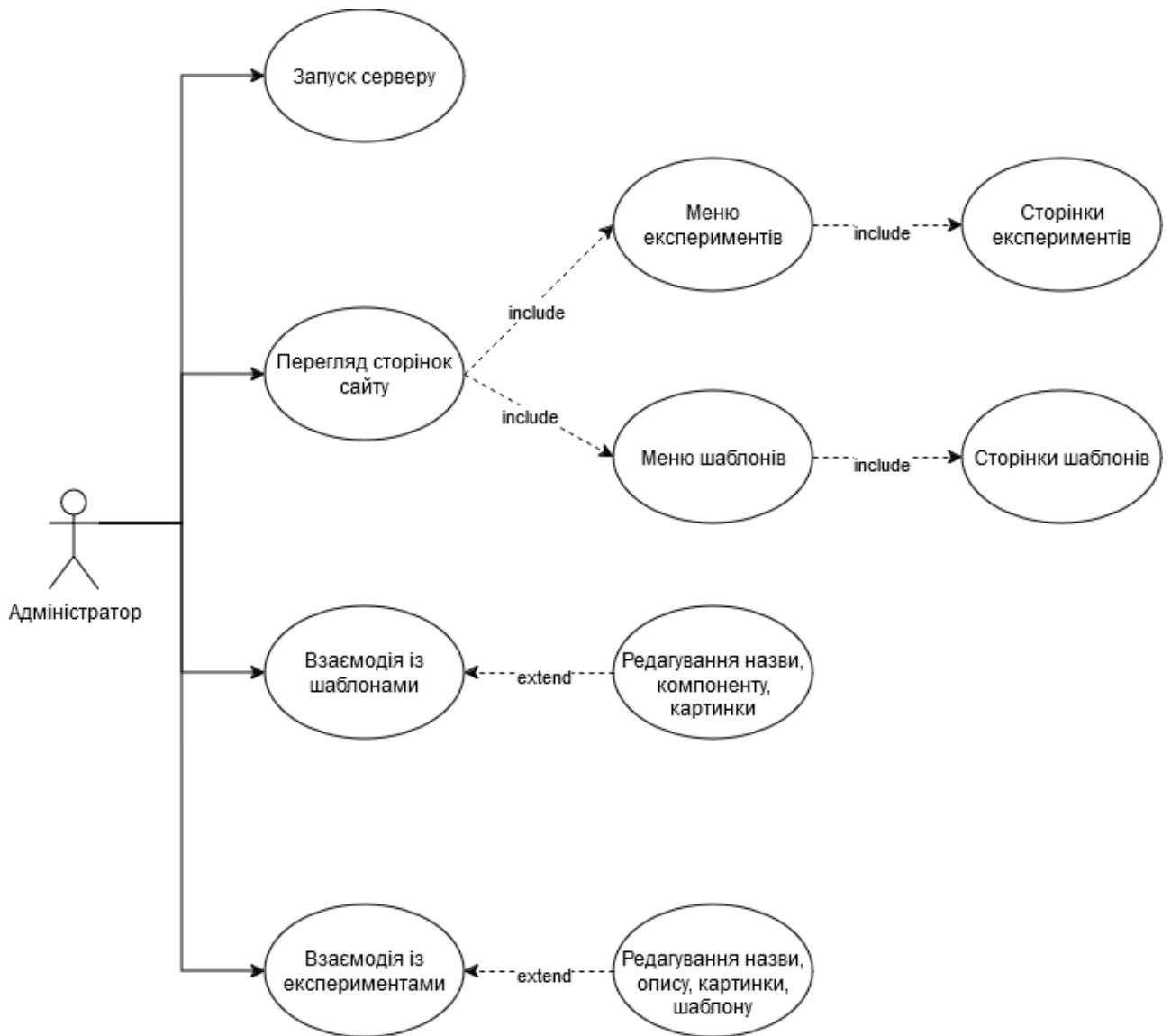


Рисунок 2.5 – Нова діаграма прецедентів адміністратора

Потік подій для користувача-розробника для розробленої діаграми прецедентів платформи (рис. 2.6).

Крок 1. Користувач підключається до серверу, щоб отримати можливість взаємодіяти із функціоналом системи.

Крок 2. Користувач переглядає сторінки сайту: меню наявних експериментів і саме сторінки експериментів.

Крок 3. Користувач взаємодіє із експериментами: запускає білд наявного коду, редагує код, переглядає результати білду і взаємодії із елементами керування експерименту (кнопками, перемикачами тощо).

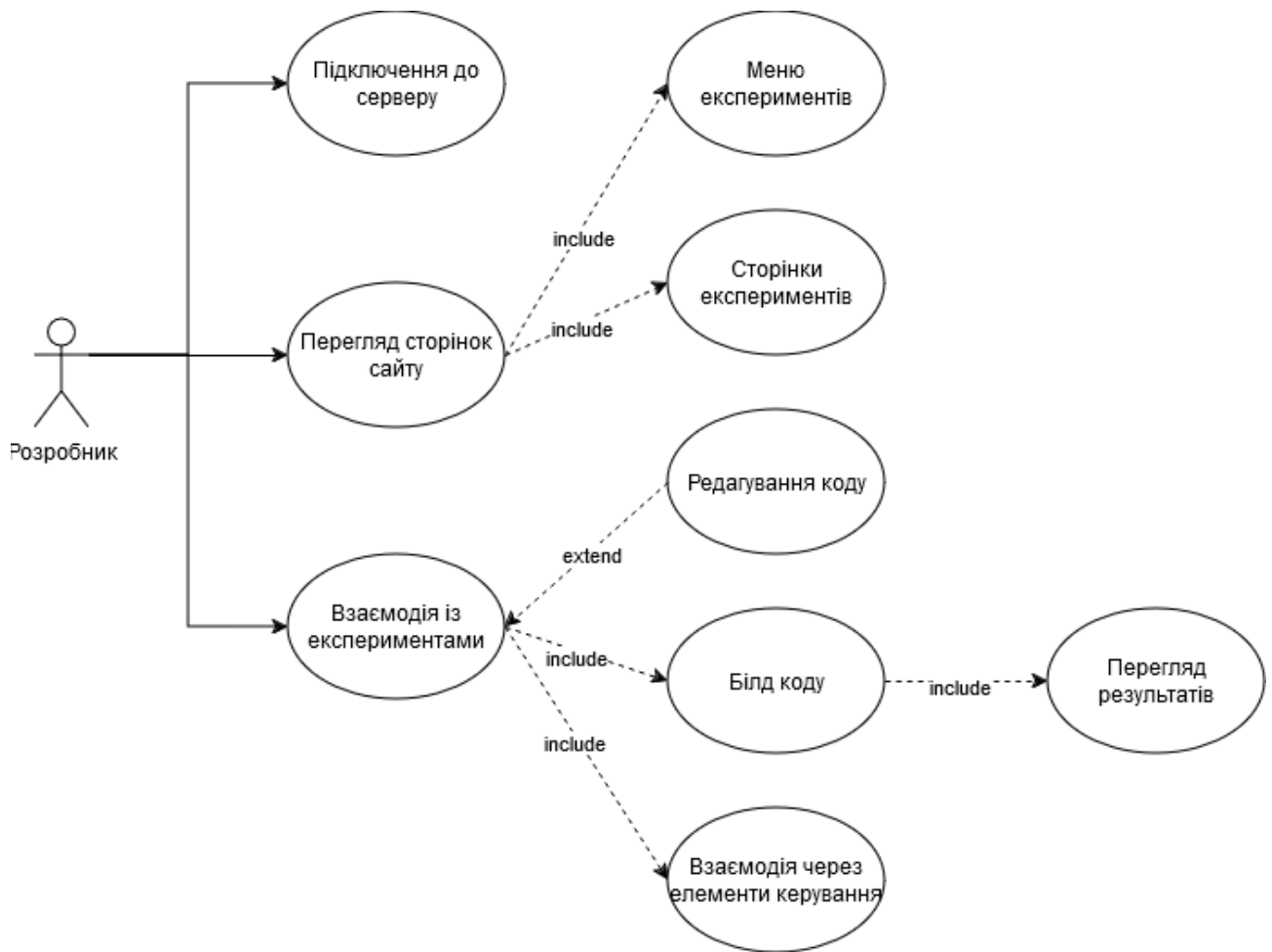


Рисунок 2.6 – Нова діаграма прецедентів розробника

2.4 Висновки за розділом 2

У даному розділі було розглянуто використані у платформах мови програмування, середовище розробки, систему керування базами даних, backend та frontend фреймворки, а також можливі для використання у майбутньому методології тестування та систему ведення документації.

Було проаналізовано архітектури програмного забезпечення і створено ряд UML-діаграм, що включають опис програмних продуктів.

Було детально розглянуто особливості фреймворків, їх модулів та сервісів, що використовуються.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

У даному розділі було розглянуто основні рішення із розробки власного експерименту та компоненту із шаблоном для нього.

3.1 Розширення функціоналу

Загальну методику процесу створення віртуальної лабораторії наведено на рис. 3.1.

Підготовка ОС описана у п. 4.2 цієї пояснювальної записки.

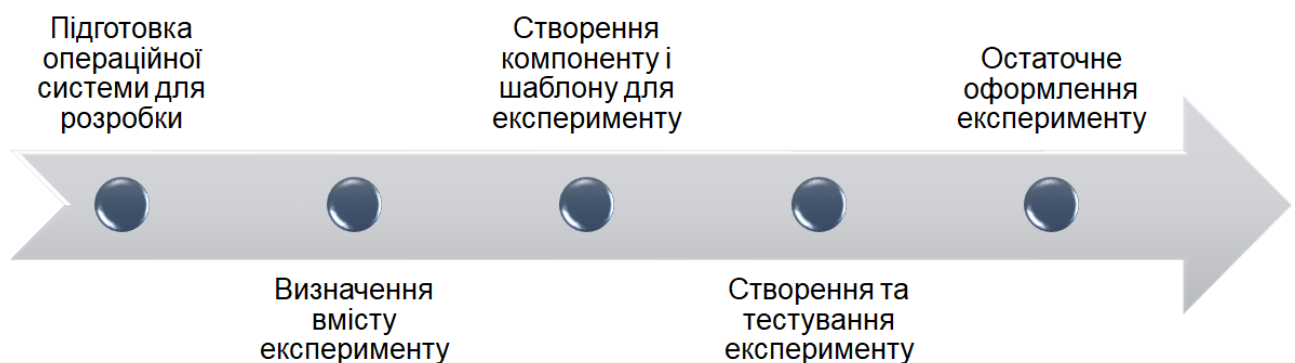


Рисунок 3.1 – Методика створення віртуальної лабораторії

3.1.1 Ідея експерименту

На початку було визначено умови нового експерименту.

Наявні 3 кнопки та 3 світлодіоди, які підключено до плати ардуїно. Під час запуску експерименту жоден світлодіод не горить. При натисканні та утримуванні кнопки, відповідний світлодіод загоряється, за який відповідає кнопка. При звільненні кнопки, світлодіод згасає.

Для сформульованого завдання було використано наступні елементи в скетчі експерименту:

- `byte leds[]` – масив пінів кнопок (у даному випадку це {13, 12, 11});
- `int i` – лічильник, що використовується для скорочення коду шляхом ітерації між пінами;
- `void setup()` – функція, що викликається один раз після запуску програми і в якій використані піни поєднуються або зі входом або з виходом;
- `void loop()` – функція, що нескінченно викликається протягом роботи програми після завершення роботи функції `setup()` і в якій постійно подається напруга на всі світлодіоди за допомогою лічильника.

Код скетчу експерименту буде зберігатися у відповідному компоненті у якості усталеного (наведено у Додатку Б.1).

3.1.2 Програмна реалізація

3.1.2.1 Створення компоненту

За прикладом було створено новий компонент «client/src/components/templates/Button.vue» (рис. 3.2).

Файл складається із наступних елементів і логічних частин:

- а) `<template>` – відповідає за структуру інтерфейсу:
 - 1) `<codemirror>` – відповідає за поле редагування коду і пов'язано зі змінною `code` за допомогою властивості `v-model`;
 - 2) `<v-btn>` – відповідає за виконання методу `sendCode` при натисканні кнопки білду «Build and simulate»;
 - 3) `<wokwi-led>` – відповідає за коректне відображення світлодіодів бібліотеки Wokwi [36];
 - 4) `<wokwi-pushbutton>` – відповідає за коректну обробку подій кнопок бібліотеки Wokwi [36], а саме натискання (`button-press`) за допомогою методу «`clickBtn(button.id)`» і звільнення (`button-release`) за допомогою методу «`buttonRelease(button.id)`»;

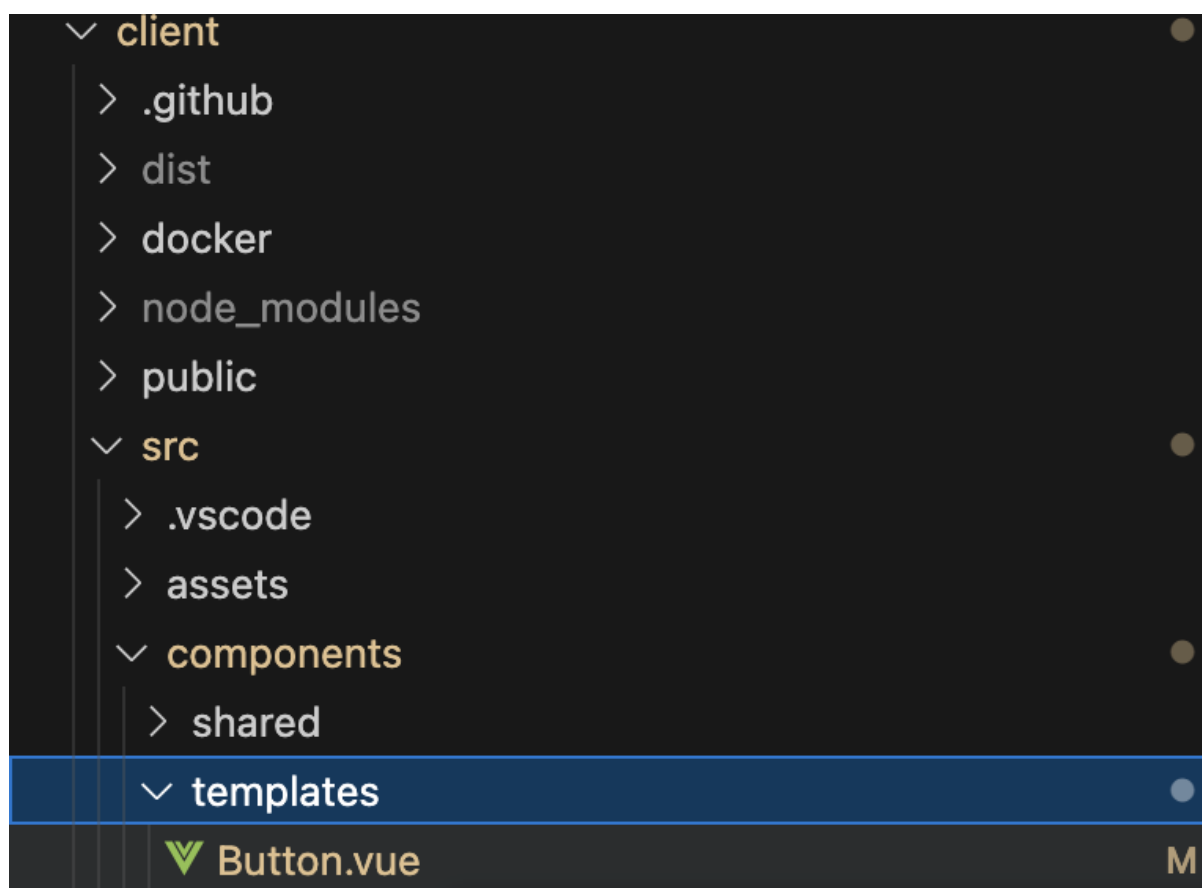


Рисунок 3.2 – Місцеположення файлу «Button.vue»

5) `<pre>` – відповідає за текстове поле успішності білду та його результатів, де відображається змінна `buildResult`;

б) `<script>` – відповідає за ініціалізацію та виконання методів:

1) на початку завантажуються необхідні бібліотеки;

2) `export default(...)` – відповідає за створення модулю, функції якого можна використати із інших файлів;

3) `data()` – метод, що відповідає за збереження значень таких змінних, як успішність білду коду (`status`, усталено пусто); результати білду (`buildResult`, усталено пусто), що включають розмір використаної пам'яті, помилки тощо; бінарний код програми (`hex`, усталено `null`); об'єкт `AVRRunner`, що відповідає за емуляцію мікроконтролеру Atmel AVR (`avrRunner`, усталено `null`); поточний код ардуїно (`code`, усталено пусто), налаштування поля редагування коду (`editorOptions`, усталено встановлено); початковий код експерименту (`code6LED`, усталено встановлено); масив світлодіодів у вигляді об'єктів (`leds`),

що складаються із унікального ідентифікатору (id), піну (pin), до якого під'єднано елемент, стан горить/не горить (value), кольору (color); масив кнопок у вигляді об'єктів (buttons), що складаються із унікального ідентифікатору (id), стану натиснуто/не натиснуто (value), і викликається під час створення екземпляру компоненту;

4) `created()` – метод, що ініціалізує `code` як `code6LED`, `avrRunner` – як новий об'єкт, додає слухачів на порти D і B (відповідно «`this.avrRunner.portD.addListener((value) => {...})`» і «`this.avrRunner.portB.addListener((value) => {...})`»), і який викликається один раз одразу після ініціалізації даних (тобто методу «`data()`»);

5) `methods` – методи, що включають до себе «`buttonRelease(id) {...}`» – зміна стану кнопки із відповідним id на ненатиснутий, «`clickBtn(id) {...}`» – зміна стану кнопки із відповідним id на натиснутий, «`async sendCode() {...}`» – встановлення hex значення `null` і спроба відправити `code` на компіляцію, із заповненням `buildResult` і `status` відповідно до результату, «`updateLEDs(value, startPin) {...}`» – оновлює `value` світлодіодів відповідно до даних, отриманих від слухачів, і викликаються лише при виникненні подій і використанні їх вручну;

в) `<style scoped>` – відповідає за стилі та оформлення конкретних елементів або класів поточного компоненту.

Код нового компоненту «`Button.vue`» наведено у Додатку Б.1.

Для того, щоб система використовувала новий компонент, у файлі «`client/src/views/Experiment.vue`» в «`export default`» поле «`components`» було додано назву компоненту (рис. 3.3).

Оновлений код файлу «`Experiment.vue`» наведено у Додатку Б.2.

Після завершення маніпуляцій із кодом необхідно запустити систему (описано у п. 4.3 цієї пояснювальної записки), увійти як адміністратор і перейти до створення шаблону.

```
import api from '@utils/helpers/api/api';
import LED from '@components/templates/LED.vue';
import Button from '@components/templates/Button.vue';

export default {
  components: {
    LED, Button
  },
}
```

Рисунок 3.3 – Додавання нового компонента у список компонентів

3.1.2.2 Створення шаблону

Щоб використати створений компонент, необхідно на сайті в якості адміністратора обрати список шаблонів і натиснути кнопку «Create» (рис. 3.4).

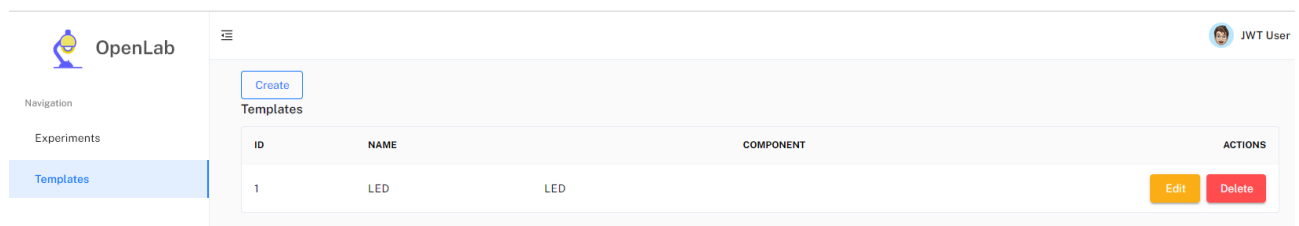


Рисунок 3.4 – Список шаблонів

Далі необхідно назвати шаблон у полі «Template name», у полі компоненту («Template component») необхідно ввести його назву («Button»). За бажанням можна додати картинку із файлів пристрою («Upload image»). Заповнену форму створення шаблону наведено на рис. 3.5.

Після натискання кнопки «Create» новий шаблон відображається у списку шаблонів (рис. 3.6).

Create template

Template name

LED Button

Template component

Button



Upload Image

Create

Close

Рисунок 3.5 – Створення шаблону

OpenLab

Navigation

Experiments

Templates

Create

Templates

ID	NAME	COMPONENT	ACTIONS
1	LED	LED	EditDelete
2	Button	Button	EditDelete

Рисунок 3.6 – Оновлений список шаблонів

3.1.2.3 Створення експерименту

Щоб використати створений шаблон, що базується на новому компоненті, необхідно на сайті в якості адміністратора обрати список експериментів і натиснути кнопку «Create» (рис. 3.7).

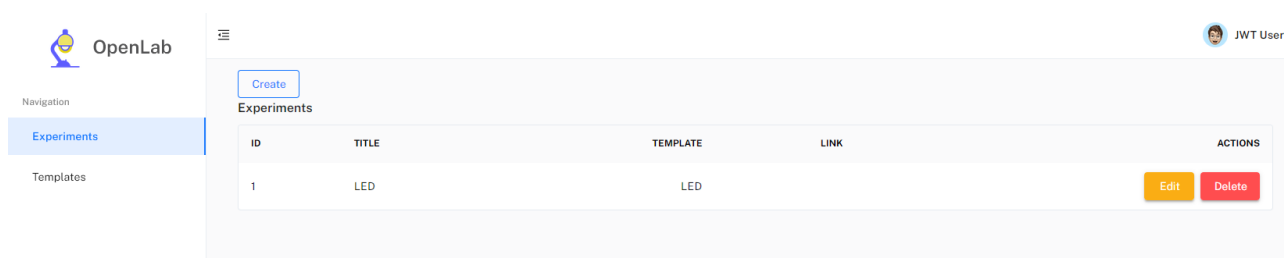


Рисунок 3.7 – Список експериментів

Далі необхідно назвати експеримент у полі «Experiment name». За бажанням у текстовому полі можна записати теоретичні вказівки, передбачувану взаємодію користувача із системою та варіанти розвитку експерименту на цій схемі (збільшення або зменшення часу на якісь дії, зміна їх порядку, додавання нових умов тощо). У полі шаблону («Select Template») необхідно обрати шаблон «Button». Заповнену форму створення експерименту наведено на рис. 3.8.

Після натискання кнопки «Create» новий експеримент відображається у списку (рис. 3.9).

Create experiment

Experiment name

Led button experiment

↩ ↪ Paragraph

B

I

U

🔗

🖼️

📅

⋮

Led experiment

LED

Button

Close

Рисунок 3.8 – Створення експерименту

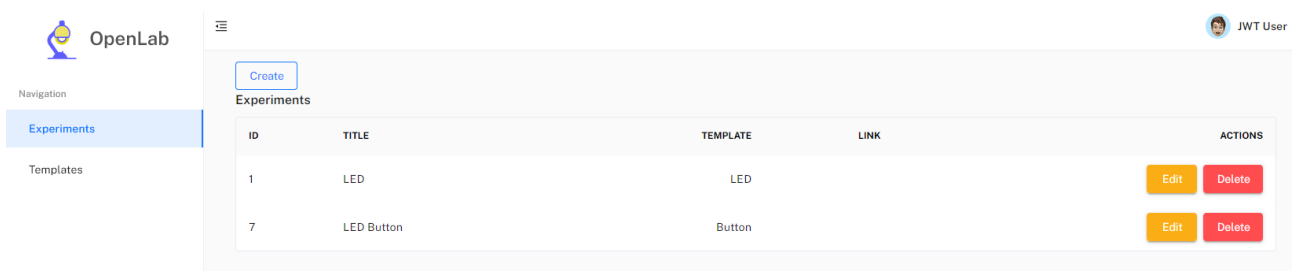


Рисунок 3.9 – Оновлений список експериментів

3.1.3 Тестування експерименту

Останнім і обов’язковим кроком у розширенні платформи є тестування розробленого функціоналу.

Після переходу на сторінку експерименту можна побачити, що всі елементи – назва, опис, поле редагування коду, код ардуїно та вид схеми, кнопка білду, поле результатів білду – відповідають тим, що було встановлено у процесі створення експерименту (рис. 3.10).

Після натискання на кнопку білду у текстовому полі нижче підтверджено, що помилок у процесі не виникло. При цьому всі світлодіоди на схемі спочатку не горять (рис. 3.11). При натисканні на одну чи декілька кнопок відповідні світлодіоди горять, при звільненні – потухають (рис. 3.12).

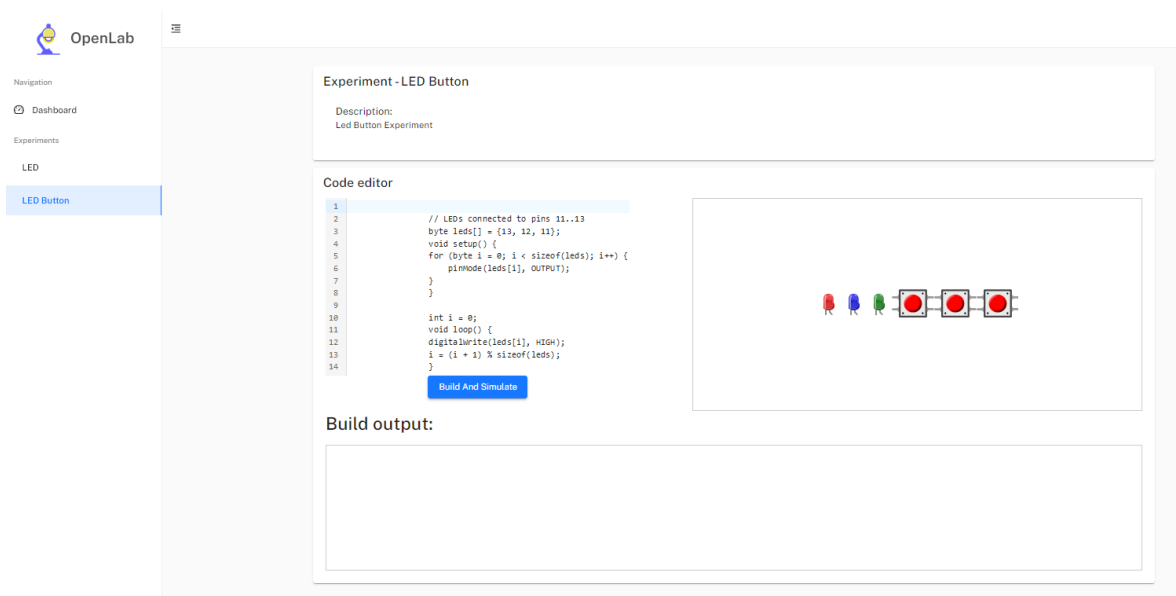


Рисунок 3.10 – Початковий вид експерименту

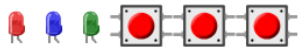
Code editor

```

1
2      // LEDs connected to pins 11..13
3      byte leds[] = {13, 12, 11};
4      void setup() {
5          for (byte i = 0; i < sizeof(leds); i++) {
6              pinMode(leds[i], OUTPUT);
7          }
8      }
9
10     int i = 0;
11     void loop() {
12         digitalWrite(leds[i], HIGH);
13         i = (i + 1) % sizeof(leds);
14     }

```

Build And Simulate



Build output:

```

[
  "Sketch uses 852 bytes (2%) of program storage space. Maximum is 32256 bytes.",
  "Global variables use 15 bytes (0%) of dynamic memory, leaving 2033 bytes for local variables. Maximum is 2048 bytes.",
  "",
  "\u001b[92mUsed platform\u001b[0m \u001b[92mVersion\u001b[0m \u001b[90mPath\u001b[0m",
  "\u001b[93marduino:avr\u001b[0m 1.8.6 \u001b[90m/var/www/html/.arduino15/packages/arduino/hardware/avr/1.8.6\u001b[0m"
]

```

Рисунок 3.11 – Експеримент після білду

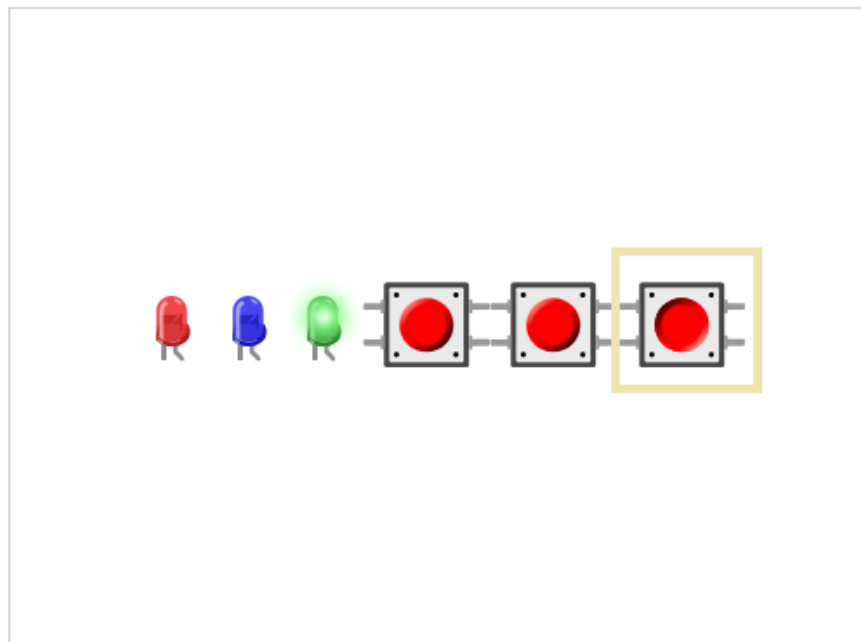


Рисунок 3.12 – Експеримент під час натискання кнопки

Робота експерименту відповідає початковим очікуванням від ідеї, описаної у п. 3.1.

3.2 Висновки за розділом 3

У третьому розділі було детально описано умови розроблюваного експерименту та всі внесені зміни до фреймворку.

Було детально описано роботу коду програмного забезпечення та його структуру.

4 КЕРІВНИЦТВО ПРОГРАМІСТА

У даному розділі було розглянуто вимоги до успішного практичного використання платформи та способів її розширення.

4.1 Призначення системи

Програмне забезпечення призначене для дослідження набору інструментів для розробки віртуальної лабораторії.

4.2 Підготовка до розробки

4.2.1 Вибір операційної системи

Для роботи із віртуальною лабораторією можна використовувати різні операційні системи (Windows, Linux, macOS, віртуальні машини), але розробник має переконатися у можливості віртуалізації для коректної роботи Docker.

Для коректного застосування фреймворку ОС та процесор, на яких буде розгортатися сервер, мають підтримувати можливість віртуалізації.

На Linux для переконання у можливості віртуалізації можна використати команди:

а) «kvm-ok» (необхідно встановити бібліотеку cpu-checker командою «sudo apt install cpu-checker»):

1) віртуалізація доступна, якщо у відповідь отримано «KVM acceleration can be used» (рис. 4.1);

2) віртуалізація не доступна, якщо у відповідь отримано «KVM acceleration can NOT be used» (рис. 4.2);

б) «egrep -c '(vmx|svm)' /proc/cpuinfo»:

1) віртуалізація доступна, якщо у відповідь отримано кількість процесорних ядер, що підтримують апаратну віртуалізацію, яка не 0 (рис. 4.3);

2) віртуалізація не доступна, якщо у відповідь отримано 0 як кількість процесорних ядер, що підтримують апаратну віртуалізацію (рис. 4.4).

```
user@ubuntu-mate:~$ kvm-ok
INFO: /dev/kvm exists
KVM acceleration can be used
```

Рисунок 4.1 – Перевірка віртуалізації (доступна) за допомогою «kvm-ok»

```
osboxes@osboxes:~$ sudo kvm-ok
[sudo] password for osboxes:
INFO: Your CPU does not support KVM extensions
KVM acceleration can NOT be used
```

Рисунок 4.2 – Перевірка віртуалізації (недоступна) за допомогою «kvm-ok»

```
user@ubuntu-mate:~$ egrep -c '(vmx|svm)' /proc/cpuinfo
3
```

Рисунок 4.3 – Перевірка віртуалізації (доступна) за допомогою «egrep -c '(vmx|svm)' /proc/cpuinfo»

```
osboxes@osboxes:~$ egrep -c '(vmx|svm)' /proc/cpuinfo
0
```

Рисунок 4.4 – Перевірка віртуалізації (недоступна) за допомогою «egrep -c '(vmx|svm)' /proc/cpuinfo»

4.2.2 Встановлення Docker Engine

Перед встановленням Docker Engine розробник має перевірити, чи відповідає обрана система вимогам. Для цього необхідно звернутися до

офіційного сайту, після чого слідувати інструкціям щодо встановлення Docker Engine [41] відповідно до ОС, на якій буде працювати розробник.

Для встановлення Docker Engine на Linux через командний рядок необхідно виконати наступні кроки (погоджуватися через «Y» на «Do you want to continue?»):

а) переконатися, що всі застарілі пакети видалено за допомогою команди «for pkg in docker.io docker-doc docker-compose docker-compose-v2 podman-docker containerd runc; do sudo apt-get remove \$pkg; done»:

1) у випадку, якщо Docker Engine перевстановлюється і виникають помилки типу «процес із id/назвою вже існує», їх можна скорегувати за допомогою команди «sudo docker rm -f <id/name>» (де «id» або «name» використовується як параметр для визначення відповідного елементу), що видаляє існуючий процес (рис. 4.5);

б) підготувати репозиторій:

- 1) «sudo apt-get update»;
- 2) «sudo apt-get install ca-certificates curl»;
- 3) «sudo install -m 0755 -d /etc/apt/keyrings»;
- 4) «sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc»;
- 5) «sudo chmod a+r /etc/apt/keyrings/docker.asc»;
- 6) використати багаторядкову команду:

```
«echo \
    "deb          [arch=$(dpkg --print-architecture) signed-
by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \
    $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
    sudo tee /etc/apt/sources.list.d/docker.list > /dev/null»
```

7) «sudo apt-get update»;

в) встановити пакети останньої версії:

1) «sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin»;

г) перевірити успішність установки:

1) «`sudo docker run hello-world`» (якщо ця команда ще не була використана (контейнер `hello-world` ще не існує), то спочатку контейнер завантажується, як на рис. 4.6, інакше – як на рис. 4.7).

```
Error response from daemon: Conflict. The container name "/laravel-api" is already in use by container "b57266557269da3f4f80acc52eb5374658adc16dfc00fa66d8b441d9ff3cd5". You have to remove (or rename) that container to be able to reuse the name.
```

```
user@ubuntu-mate:~/LAB/laboratory$ sudo docker rm -f b57266557269da3f4f80acc52eb5374658adc16dfc00fa66d8b441d9ff3cd5
b57266557269da3f4f80acc52eb5374658adc16dfc00fa66d8b441d9ff3cd5
```

Рисунок 4.5 – Видалення існуючого елементу

```
user@ubuntu-mate:~/LAB$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:a26bff933ddc26d5cdf7faa98b4ae1e3ec20c4985e6f87ac0973052224d24302
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
```

Рисунок 4.6 – Перевірка успішності встановлення Docker Engine (контейнер не існує)

```

user@ubuntu-mate:~$ sudo docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/

```

Рисунок 4.7 – Перевірка успішності встановлення Docker Engine (контейнер існує)

4.2.3 Налаштування Git проєкту

Для роботи із проєктом розробник має використовувати Git.

На Linux git можна встановити за допомогою команди «sudo apt install git» або подібних до неї в залежності від версії ОС.

4.2.3.1 Ключ SSH

Перед тим, як отримати проєкт з GitHub, необхідно у своєму акаунті мати ключ протоколу SSH (Secure Shell). Якщо він уже існує, новий створювати необов'язково. Для отримання та прив'язки ключа необхідно [42]:

а) перевірити наявність ключа (рис. 4.8):

- 1) використати команду «ssh-agent sh -c 'ssh-add; ssh-add -L'»;
- 2) ввести відповідний passphrase і отримати ключ у консолі

(також можна знайти файл із ключем і скопіювати його звідти);

б) згенерувати новий ключ (рис. 4.9):

```
root@ubuntu-mate:/home/user/LAB2# ssh-agent sh -c 'ssh-add; ssh-add -L'
Enter passphrase for /root/.ssh/id_ed25519:
Identity added: /root/.ssh/id_ed25519 (smartnica@gmail.com)
ssh-ed25519 AAAAC3NzaC1lbnRpdjEAAAADAQABAAQDAgkA8WwZ0UwzYmVtZW50IGFkaXN0eSBjaGVyIDB0MGE2IHRlc2UvZmVudG91dDQ7Ys
smartnica@gmail.com
root@ubuntu-mate:/home/user/LAB2#
```

- 1) перейти до можливостей адміністратора через «sudo -s -H»;
- 2) включити на фоні SSH-агента командою «eval "\$(ssh-agent -s)"», отримав ідентифікатор процесу;
- 3) додати створений ключ (файл у команді відображено усталений) командою «ssh-add ~/.ssh/id_ed25519» та ввести passphrase;
- г) додати ключ до GitHub (рис. 4.11-4.12):
 - 1) перейти до налаштувань акаунту GitHub, до пункту меню «SSH and GPG keys», і натиснути кнопку «New SSH key»;
 - 2) ввести назву та текст ключа у відповідні поля, натиснути кнопку «Add SSH key»;

```

root@ubuntu-mate: /home/user/LAB2 (as superuser)
File Edit View Search Terminal Help
root@ubuntu-mate:/home/user/LAB2# sudo -s -H
root@ubuntu-mate:/home/user/LAB2# eval "$(ssh-agent -s)"
Agent pid 50125
root@ubuntu-mate:/home/user/LAB2# ssh-add ~/.ssh/id_ed25519
Enter passphrase for /root/.ssh/id_ed25519:
Identity added: /root/.ssh/id_ed25519 (smartnica@gmail.com)
root@ubuntu-mate:/home/user/LAB2# ssh-agent sh -c 'ssh-add; ssh-add -L'
Enter passphrase for /root/.ssh/id_ed25519:
Identity added: /root/.ssh/id_ed25519 (smartnica@gmail.com)
smartnica@gmail.com
  
```

Рисунок 4.10 – Додавання ключа до агента

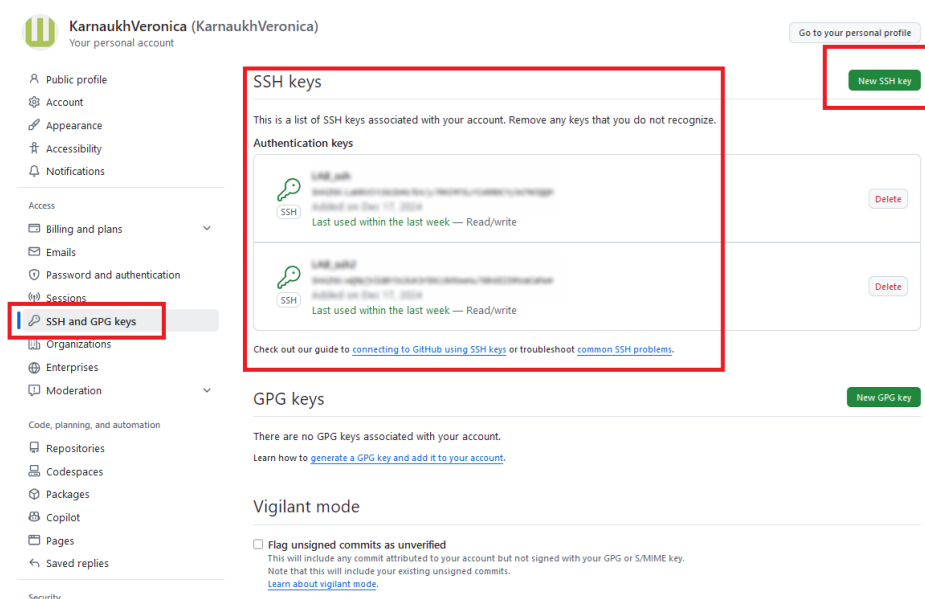
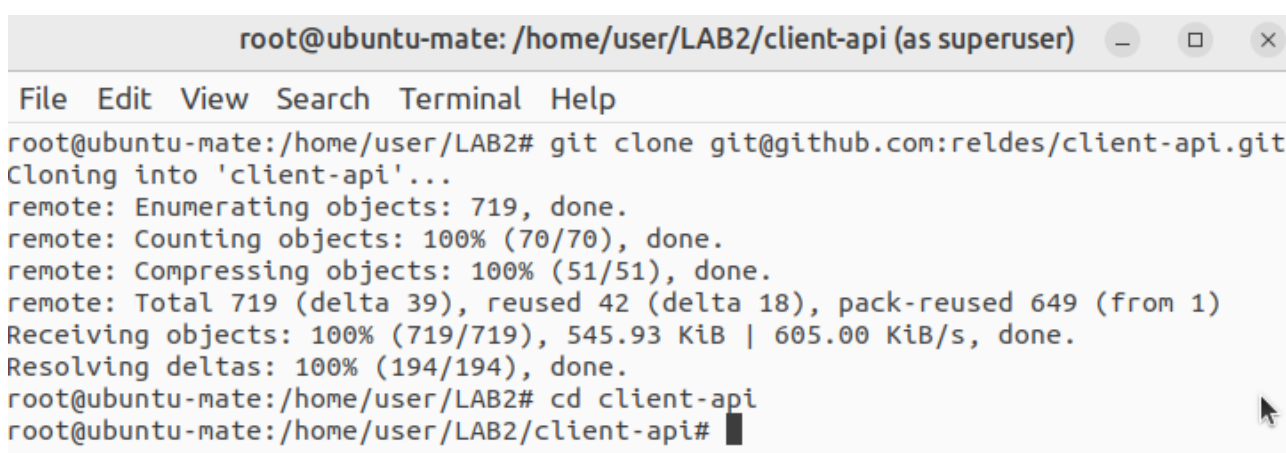


Рисунок 4.11 – Налаштування ключів SSH

4.2.3.2 Клонування проєкту

Для отримання локальної копії проєкту необхідно спочатку відкрити термінал у тій папці, де буде розташовано проєкт, і виконати команду клонування для client-api «`git clone git@github.com:reldes/client-api.git`» і перейти у терміналі до папки client-api за допомогою команди «`cd client-api`» (рис. 4.14).



```

root@ubuntu-mate: /home/user/LAB2/client-api (as superuser)
File Edit View Search Terminal Help
root@ubuntu-mate:/home/user/LAB2# git clone git@github.com:reldes/client-api.git
Cloning into 'client-api'...
remote: Enumerating objects: 719, done.
remote: Counting objects: 100% (70/70), done.
remote: Compressing objects: 100% (51/51), done.
remote: Total 719 (delta 39), reused 42 (delta 18), pack-reused 649 (from 1)
Receiving objects: 100% (719/719), 545.93 KiB | 605.00 KiB/s, done.
Resolving deltas: 100% (194/194), done.
root@ubuntu-mate:/home/user/LAB2# cd client-api
root@ubuntu-mate:/home/user/LAB2/client-api#
  
```

Рисунок 4.14 – Клонування client-api

Після клонування необхідно виконати наступні кроки:

а) відкрити термінал у папці із проєктом (рис. 4.15-4.16):

1) якщо це перший запуск чи було внесено зміни до проєкту, то виконати команду «`sudo docker compose build`»;

2) виконати команду «`sudo docker compose up`»;

3) при першому виконанні займає більше часу;

б) якщо це перший запуск, то відкрити другий термінал:

1) виконати команду «`sudo docker ps`» для перегляду працюючих контейнерів (рис. 4.17);

2) виконати команду для переходу до PHP-клієнту «`sudo docker exec -it <id_container> bash`», де <id_container> - ідентифікаційний номер контейнеру openlab-api, для переходу до цього контейнеру (рис. 4.18);

```

root@ubuntu-mate: /home/user/LAB2/client-api (as superuser)
File Edit View Search Terminal Help
root@ubuntu-mate:/home/user/LAB2/client-api# docker compose build
WARN[0000] /home/user/LAB2/client-api/docker-compose.yml: `version` is obsolete
[+] Building 116.5s (7/18)                                docker:default
=> => sha256:4ff6d89d3ef610dabc80f67546410c15d38e6c2e1c1b0c1 225B / 225B 1.7s
=> => sha256:f58c81aa650daba34415db72a2f7cf6806ca5c 104.15MB / 104.15MB 63.1s
=> => sha256:3939fcc93e45b0dfa23ab0cdd22a085bbcd10a850050c98 224B / 224B 3.4s
=> => sha256:41d32ab61a2356a4b3569e1e9ac371ea80174c6a 12.63MB / 12.63MB 12.0s
=> => sha256:fcbb62ad600013e86e34b48d6b176b14265f2e274df07f 486B / 486B 12.4s
=> => sha256:27581b1affb6abf5a03956188fa9a903e4f7e4ce 27.82MB / 27.82MB 34.2s
=> => extracting sha256:bc0965b23a04fe7f2d9fb20f597008fcf89891de1c705ffc 0.6s
=> => sha256:d6f4e845f2c4077e85902bf744a24f47c39beb0a4f 2.45kB / 2.45kB 31.3s
[+] Building 116.7s (7/18)                                docker:default
=> => sha256:4ff6d89d3ef610dabc80f67546410c15d38e6c2e1c1b0c1 225B / 225B 1.7s
=> => sha256:f58c81aa650daba34415db72a2f7cf6806ca5c 104.15MB / 104.15MB 63.1s
=> => sha256:3939fcc93e45b0dfa23ab0cdd22a085bbcd10a850050c98 224B / 224B 3.4s
=> => sha256:41d32ab61a2356a4b3569e1e9ac371ea80174c6a 12.63MB / 12.63MB 12.0s
=> => sha256:fcbb62ad600013e86e34b48d6b176b14265f2e274df07f 486B / 486B 12.4s
=> => sha256:27581b1affb6abf5a03956188fa9a903e4f7e4ce 27.82MB / 27.82MB 34.2s
=> => extracting sha256:bc0965b23a04fe7f2d9fb20f597008fcf89891de1c705ffc 0.6s
=> => sha256:d6f4e845f2c4077e85902bf744a24f47c39beb0a4f 2.45kB / 2.45kB 31.3s
[+] Building 116.8s (7/18)                                docker:default
=> => sha256:4ff6d89d3ef610dabc80f67546410c15d38e6c2e1c1b0c1 225B / 225B 1.7s
=> => sha256:f58c81aa650daba34415db72a2f7cf6806ca5c 104.15MB / 104.15MB 63.1s
=> => sha256:3939fcc93e45b0dfa23ab0cdd22a085bbcd10a850050c98 224B / 224B 3.4s

```

Рисунок 4.15 – Білд образу client-api

```

root@ubuntu-mate: /home/user/LAB2/client-api (as superuser)
File Edit View Search Terminal Help
=> [php-fpm 14/15] RUN chown dev:dev /home/dev 0.6s
=> [php-fpm 15/15] RUN chmod 755 /home/dev 0.5s
=> [php-fpm] exporting to image 2.5s
=> => exporting layers 2.4s
=> => writing image sha256:73c72c3a537410dd825b30f86e964940816f0ea981715 0.0s
=> => naming to docker.io/library/client-api-php-fpm 0.0s
root@ubuntu-mate:/home/user/LAB2/client-api# docker compose up
WARN[0000] /home/user/LAB2/client-api/docker-compose.yml: `version` is obsolete
[+] Running 2/5
✔ Network client-api_public_net Creat... 0.1s
✔ Volume "client-api_data-db" Created 0.0s
:: Container openlab-db Created 0.5s
:: Container openlab-api Created 0.5s
:: Container client-api-nginx-1 Create... 0.1s
Attaching to nginx-1, openlab-api, openlab-db
openlab-api | [17-Dec-2024 01:47:58] NOTICE: PHP message: Xdebug: [Config] The
setting 'xdebug.remote_port' has been renamed, see the upgrading guide at https:
//xdebug.org/docs/upgrade_guide#changed-xdebug.remote_port (See: https://xdebug.
org/docs/errors#CFG-C-CHANGED)
openlab-api | [17-Dec-2024 01:47:58] NOTICE: [pool www] 'user' directive is ign
ored when FPM is not running as root
openlab-api | [17-Dec-2024 01:47:58] NOTICE: [pool www] 'group' directive is ig
nored when FPM is not running as root
openlab-api | [17-Dec-2024 01:47:58] NOTICE: fpm is running, pid 1

```

Рисунок 4.16 – Запуск образу client-api

3) у новому клієнті виконати команду «ср .env.example .env» для копіювання файлу .env.example до файлу .env (рис. 4.18);

```

root@ubuntu-mate:/home/user/LAB2/client-api# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED
STATUS        PORTS          NAMES
a796206c1fc5   nginx:latest   "/docker-entrypoint..." 9 minutes a
go    Up 9 minutes   0.0.0.0:8080->80/tcp, :::8080->80/tcp, 0.0.0.0:4430-
>443/tcp, :::4430->443/tcp   client-api-nginx-1
38f48d51856b   client-api-php-fpm "docker-php-entrypoi..." 9 minutes a
go    Up 9 minutes   9000/tcp                 openlab-api
96c195d51b7f   mysql:8.0      "docker-entrypoint.s..." 9 minutes a
go    Up 9 minutes   33060/tcp, 0.0.0.0:3336->3306/tcp, :::3336->3306/tcp
openlab-db
root@ubuntu-mate:/home/user/LAB2/client-api#

```

Рисунок 4.17 – Перевірка працюючих контейнерів

```

user@ubuntu-mate:~/LAB2/client-api$ sudo docker exec -it fb2dfd971009 bash
dev@fb2dfd971009:/var/www/html$ USER root
bash: USER: command not found
dev@fb2dfd971009:/var/www/html$ cp .env.example .env

```

Рисунок 4.18 – Копіювання .env.example до .env

4) у файлі .env необхідно вручну змінити рядки «DB_HOST=127.0.0.1» на «DB_HOST=db» і «DB_PASSWORD=» на «DB_PASSWORD=password»;

5) виконати команду «composer install» для встановлення необхідних бібліотек (рис. 4.19);

6) оновити БД за допомогою команд: «php artisan config:clear» і «php artisan cache:clear» (видалення кешу), «php artisan migrate» і «php artisan key:generate» (якщо laravel БД не було знайдено, необхідно погодитися на її створення) (рис. 4.20-4.21);

7) після цього термінал можна закрити;

```
dev@fb2dfd971009:/var/www/html$ composer install
Xdebug: [Config] The setting 'xdebug.remote_port' has been renamed, see the upgr
ading guide at https://xdebug.org/docs/upgrade_guide#changed-xdebug.remote_port
(See: https://xdebug.org/docs/errors#CFG-C-CHANGED)
Installing dependencies from lock file (including require-dev)
Verifying lock file contents can be installed on current platform.
Package operations: 112 installs, 0 updates, 0 removals
- Downloading doctrine/inflector (2.0.10)
- Downloading doctrine/lexer (3.0.1)
- Downloading symfony/polyfill-ctype (v1.31.0)
- Downloading webmozart/assert (1.11.0)
- Downloading dragonmantank/cron-expression (v3.4.0)
- Downloading symfony/deprecation-contracts (v3.5.0)
- Downloading psr/container (2.0.2)
- Downloading fakerphp/faker (v1.24.0)
- Downloading symfony/polyfill-php83 (v1.31.0)
- Downloading symfony/polyfill-mbstring (v1.31.0)
- Downloading symfony/http-foundation (v7.1.7)
- Downloading fruitcake/php-cors (v1.3.0)
- Downloading psr/http-message (2.0)
```

Рисунок 4.19 – Встановлення необхідних бібліотек

```
dev@b2bfaa16be9f:/var/www/html$ php artisan migrate
Xdebug: [Config] The setting 'xdebug.remote_port' has been renamed, see the upgr
ading guide at https://xdebug.org/docs/upgrade_guide#changed-xdebug.remote_port
(See: https://xdebug.org/docs/errors#CFG-C-CHANGED)

WARN The database 'laravel' does not exist on the 'mysql' connection.

Would you like to create it? 
Yes

INFO Preparing database.

Creating migration table ..... 184.97ms DONE

INFO Running migrations.

2014_10_12_000000_create_users_table ..... 600.01ms DONE
2014_10_12_100000_create_password_reset_tokens_table ..... 188.41ms DONE
2019_08_19_000000_create_failed_jobs_table ..... 315.45ms DONE
2019_12_14_000001_create_personal_access_tokens_table ..... 439.49ms DONE
2024_10_06_085531_create_experiments_table ..... 181.41ms DONE
2024_10_14_151953_create_templates_table ..... 208.24ms DONE
2024_10_14_152532_add_template_id_to_experiments_table ..... 593.10ms DONE
2024_11_03_121328_expand_experiment_description_length ..... 488.43ms DONE
```

Рисунок 4.20 – Міграція БД

```
dev@b2bfaa16be9f:/var/www/html$ php artisan key:generate
Xdebug: [Config] The setting 'xdebug.remote_port' has been renamed, see the upgrading guide at https://xdebug.org/docs/upgrade_guide#changed-xdebug.remote_port (See: https://xdebug.org/docs/errors#CFG-C-CHANGED)
```

INFO Application key set successfully.

Рисунок 4.21 – Генерація ключів

в) відкрити браузер (рис. 4.22):

- 1) перейти за посиланням localhost:8080;
- 2) впевнитися у наявності повідомлення про успішне

розгортання.

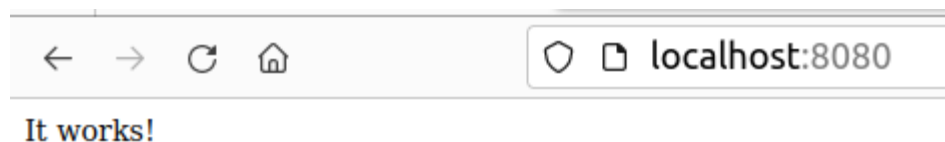


Рисунок 4.22 – Сторінка успішної роботи client-api

Далі необхідно відкрити другий термінал у тій папці, де розташовано client-api, виконати команду клонування для client «git clone git@github.com:reldes/client.git» і перейти у терміналі до папки client за допомогою команди «cd client» (рис. 4.23).

Також необхідно відкрити третій термінал у тій папці, де розташовано client-api, виконати команду клонування для admin «git clone git@github.com:reldes/admin.git» і перейти у терміналі до папки client за допомогою команди «cd admin» (рис. 4.24).

```

user@ubuntu-mate:~/LAB2$ git clone git@github.com:reldes/client.git
Cloning into 'client'...
remote: Enumerating objects: 416, done.
remote: Counting objects: 100% (416/416), done.
remote: Compressing objects: 100% (267/267), done.
remote: Total 416 (delta 119), reused 359 (delta 72), pack-reused 0 (from 0)
Receiving objects: 100% (416/416), 1.50 MiB | 1.02 MiB/s, done.
Resolving deltas: 100% (119/119), done.
user@ubuntu-mate:~/LAB2$ cd client
user@ubuntu-mate:~/LAB2/client$ 

```

Рисунок 4.23 – Клонування client

```

user@ubuntu-mate:~/LAB2$ git clone git@github.com:reldes/admin.git
Cloning into 'admin'...
remote: Enumerating objects: 301, done.
remote: Counting objects: 100% (301/301), done.
remote: Compressing objects: 100% (189/189), done.
remote: Total 301 (delta 72), reused 291 (delta 64), pack-reused 0 (from 0)
Receiving objects: 100% (301/301), 1.52 MiB | 869.00 KiB/s, done.
Resolving deltas: 100% (72/72), done.

```

Рисунок 4.24 – Клонування admin

Після клонування необхідно виконати наступні кроки у другому терміналі:

а) відкрити термінал у папці із проєктом (аналогічно до рис. 4.15-4.16):

1) якщо це перший запуск чи було внесено зміни до проєкту, то виконати команду «sudo docker compose build»;

2) виконати команду «sudo docker compose up»;

3) при першому виконанні займає більше часу;

б) у файлі docker-compose.yml необхідно вручну змінити ports з «5173:5173» до «5174:5173».

Після клонування необхідно виконати наступні кроки у третьому терміналі:

а) відкрити термінал у папці із проєктом (аналогічно до рис. 4.15-4.16):

- 1) якщо це перший запуск чи було внесено зміни до проєкту, то виконати команду «`sudo docker compose build`»;
- 2) виконати команду «`sudo docker compose up`»;
- 3) при першому виконанні займає більше часу.

Після виконання цих дій і при умові роботи всіх трьох контейнерів у терміналах, користувач може звернутися до `localhost:5174`, або `https://client.openlab.org.ua` (для розробника), і `localhost:5173`, або `https://admin.openlab.org.ua` (для адміністратора). Результати успішного запуску сторінок наведено на рис. 4.25-4.26.

Для завершення роботи із системою достатньо вийти із браузера та у терміналах із запусченими процесами натиснути «`Ctrl+C`».

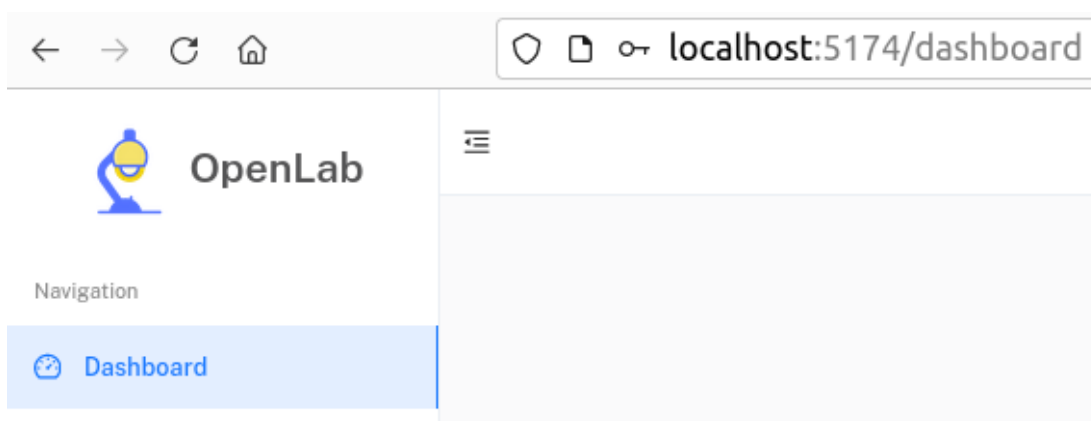


Рисунок 4.25 – Сторінка розробника

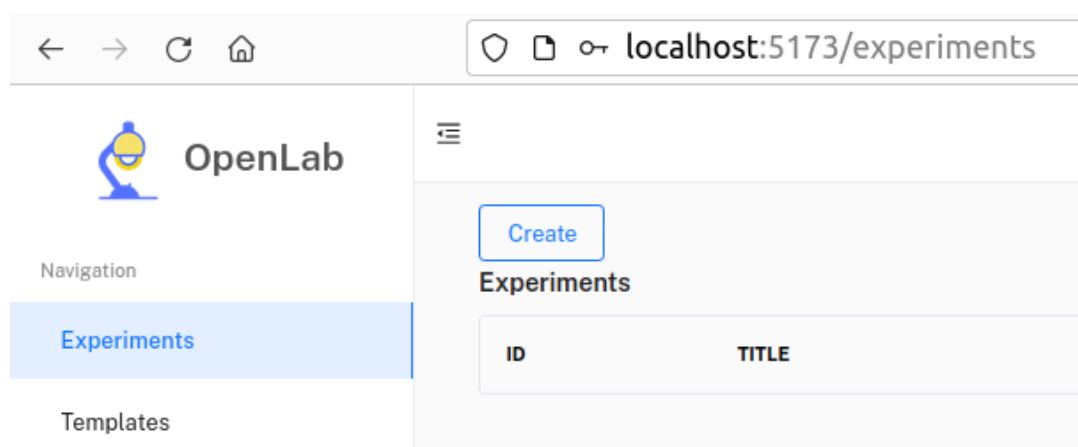


Рисунок 4.26 – Сторінка адміністратора

4.2.3.3 Використання акаунту

У випадку, якщо laravel БД була створена з нуля чи необхідно додати нового користувача, це можна зробити за допомогою терміналу, для чого необхідно:

а) відкрити термінал у папці із проєктом і виконати команду «sudo docker compose up»;

б) відкрити другий термінал, де перейти до PHP-клієнту («sudo docker exec -it openlab-api bash») і для обраного паролю («securepassword») згенерувати його хеш командою «php -r "echo password_hash('securepassword', PASSWORD_BCRYPT);"», отриманий результат скопіювати чи дець записати («\$2y\$10\$qUkHB8qiN4qZNHaxXdrK7eFniPGq9cH4gbfeI1lvJJl2PjNa5eMfi»), після чого вийти із клієнту командою «exit»;

в) у другому терміналі використати команду «sudo docker exec -it openlab-db mysql -u root -p» для переходу в MySQL-клієнт у контейнері «openlab-db». Далі необхідно ввести root пароль до СКБД («password») і почати працювати із її даними за допомогою SQL-запитів:

1) для отримання списку всіх БД необхідно виконати запит «SHOW DATABASES;» (рис. 4.27);

```
mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| db       |
| information_schema |
| laravel  |
| mysql    |
| performance_schema |
| sys      |
+-----+
6 rows in set (0.01 sec)
```

Рисунок 4.27 – Наявні БД

2) інформація про користувачів зберігається у laravel, тому виконуємо запит «USE laravel;» для переключення поточної БД, після чого виводимо всі таблиці БД за допомогою «SHOW TABLES;» (рис. 4.28-4.29);

3) для огляду структури (поля, типи даних, чи може значення дорівнювати null, усталене значення, чи є первинним ключем) таблиці users необхідно використати команду «DESCRIBE users;» (рис. 4.30);

4) переглянути, чи прив'язано адресу електронної пошти (яка має бути унікальною посеред усіх значень цього поля) до якогось акаунту можна за допомогою запиту «SELECT * FROM users WHERE email = '<email>';» (рис. 4.31);

5) додати користувача можна командою «INSERT INTO users (name, email, password) VALUES ('<name surname>', '<email>', '<hash>');», а верифікувати пошту – «UPDATE users SET email_verified_at = NOW() WHERE email = '<email>';»;

г) після цього можна зайти до системи за допомогою даних цього аканту (рис. 4.32).

```
mysql> USE laravel;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
```

Рисунок 4.28 – Переключення поточної БД

```
mysql> SHOW TABLES;
+-----+
| Tables_in_laravel |
+-----+
| experiments       |
| failed_jobs       |
| migrations        |
| password_reset_tokens |
| personal_access_tokens |
| templates         |
| users             |
+-----+
7 rows in set (0.00 sec)
```

Рисунок 4.29 – Таблиці у БД Laravel

```
mysql> DESCRIBE users;
```

Field	Type	Null	Key	Default	Extra
id	bigint unsigned	NO	PRI	NULL	auto_increment
name	varchar(255)	NO		NULL	
email	varchar(255)	NO	UNI	NULL	
email_verified_at	timestamp	YES		NULL	
password	varchar(255)	NO		NULL	
remember_token	varchar(100)	YES		NULL	
created_at	timestamp	YES		NULL	
updated_at	timestamp	YES		NULL	

Рисунок 4.30 – Структура users

```
mysql> INSERT INTO users (name, email, password)
-> VALUES ('test test', 'test@test.com', '$2y$10$qUkHB8qiN4qZNHaxXdrK7eFniPG
q9cH4gbfeI1lvJJl2PjNa5eMfi');
Query OK, 1 row affected (0.04 sec)

mysql> UPDATE users
-> SET email_verified_at = NOW() WHERE email = 'test@test.com';
Query OK, 1 row affected (0.04 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> SELECT * FROM users WHERE email = 'test@test.com'
-> ;
+-----+-----+-----+-----+-----+-----+
| id | name | email | email_verified_at | password |
| | | | | | |
+-----+-----+-----+-----+-----+-----+
| 3 | test test | test@test.com | 2024-12-18 10:53:03 | $2y$10$qUkHB8qiN4qZNHaxXdrK7eFniPGq9cH4gbfeI1lvJJl2PjNa5eMfi | NULL | NULL | NULL
```

Рисунок 4.31 – Робота із акаунтом у БД

Login

[Don't Have an account?](#)

Email Address

Password

Рисунок 4.32 – Вхід до системи

4.3 Запуск системи

Для запуску готової системи необхідно виконати наступні кроки:

- відкрити термінал у папці із проєктом, де знаходяться папки client-api, client і admin, після чого відкрити ще 2 термінали у тому же місці;
- у кожному терміналі перейти до однієї із папок за допомогою команди «cd <dir>»;
- у кожному терміналі, у порядку папок client-api, client, admin, виконати команду «sudo docker compose up»;
- щоб перейти до можливостей розробника, перейти за посиланням localhost:5174, до можливостей адміністратора – localhost:5173.

4.4 Розширення функціоналу фреймворку

4.4.1 Робота із експериментами

На сторінці експериментів (Experiments) адміністратору показано список усіх існуючих експериментів із їх ID, назвою і шаблоном (рис. 4.33). Якщо натиснути на кнопку «Edit», то відкриється вікно, де можна редагувати характеристики експерименту, включаючи картинки і опис (рис. 4.34). Для збереження змін достатньо натиснути кнопку «Update». Також можна видалити експеримент натисканням кнопки «Delete». Якщо натиснути на кнопку «Create», то відкриється вікно, подібне до рис. 4.34, де можна створити новий експеримент.

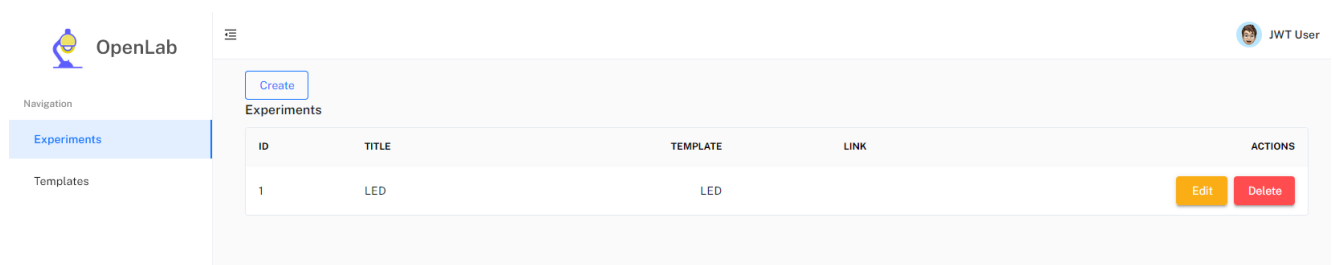


Рисунок 4.33 – Список експериментів у адміністратора

Update experiment

Experiment name

LED

↶

↷

Paragraph

▼

B

I

U





▼











▼

Description:

Objective:

To design and implement a circuit using an Arduino board to control the illumination patterns of four LEDs. The circuit will be programmed using Arduino IDE and uploaded to the Arduino board.

Procedure:

LED

Update

Рисунок 4.34 – Редагування експерименту в адміністратора

4.4.2 Створення шаблонів

На сторінці шаблонів (Templates) адміністратору показано список усіх існуючих шаблонів із їх ID, назвою і компонентом (рис. 4.35). Якщо натиснути на кнопку «Edit», то відкриється вікно, де можна редагувати характеристики шаблону, включаючи картинку (рис. 4.36). Для збереження змін достатньо натиснути кнопку «Update». Також можна видалити шаблон натисканням кнопки «Delete». Якщо натиснути на кнопку «Create», то відкриється вікно, подібне до рис. 4.36, де можна створити новий шаблон.


OpenLab




JWT User

Navigation

Experiments

Templates

Create

Templates

ID	NAME	COMPONENT	ACTIONS
1	LED	LED	Edit Delete

Рисунок 4.35 – Список шаблонів у адміністратора

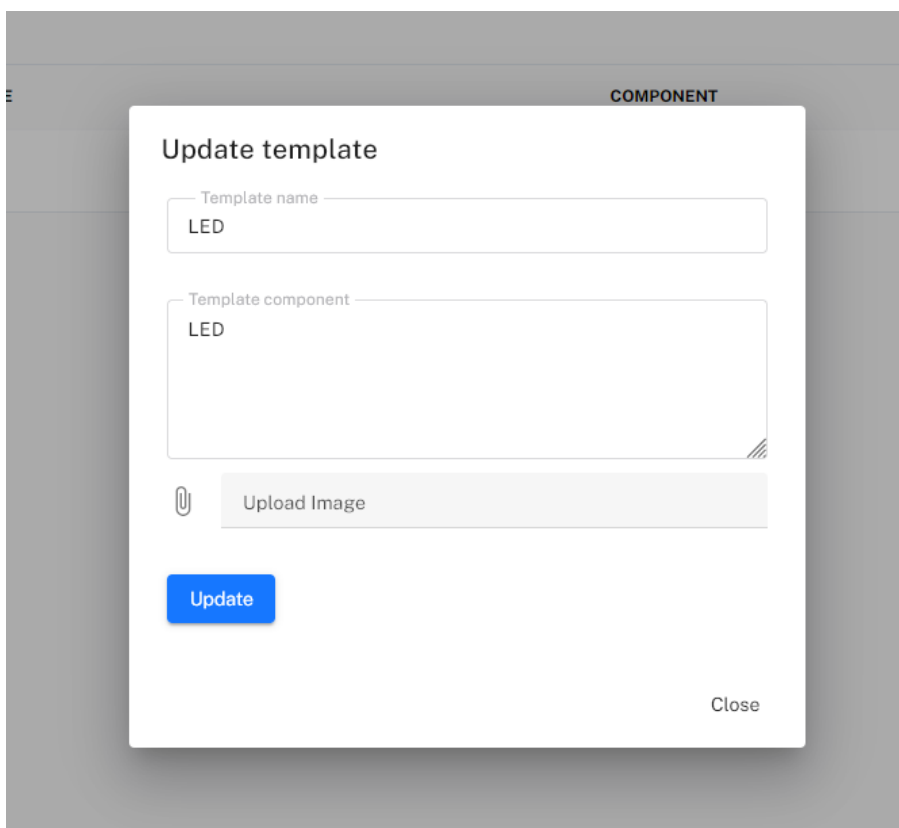


Рисунок 4.36 – Редагування шаблону в адміністратора

4.4.3 Створення компонентів

Робота із компонентами (створення, редагування та видалення) можлива лише через зміну файлів ситсеми, за допомогою інтерфейсу на сайті можливо лише їх використовувати.

Основний алгоритм створення компонентів наведено у п. 3.1.2.1 цієї пояснювальної записки.

4.4.4 Тестування

Для перевірки роботи експерименту необхідно перейти на сторінку розробника (localhost:5174), де обрати експеримент і запустити його (рис. 4.37). Якщо експеримент успішно почав симуляцію, необхідно переконатися, що виконуються основні можливі сценарії, передбачувані його ціллю та наявними елементами.

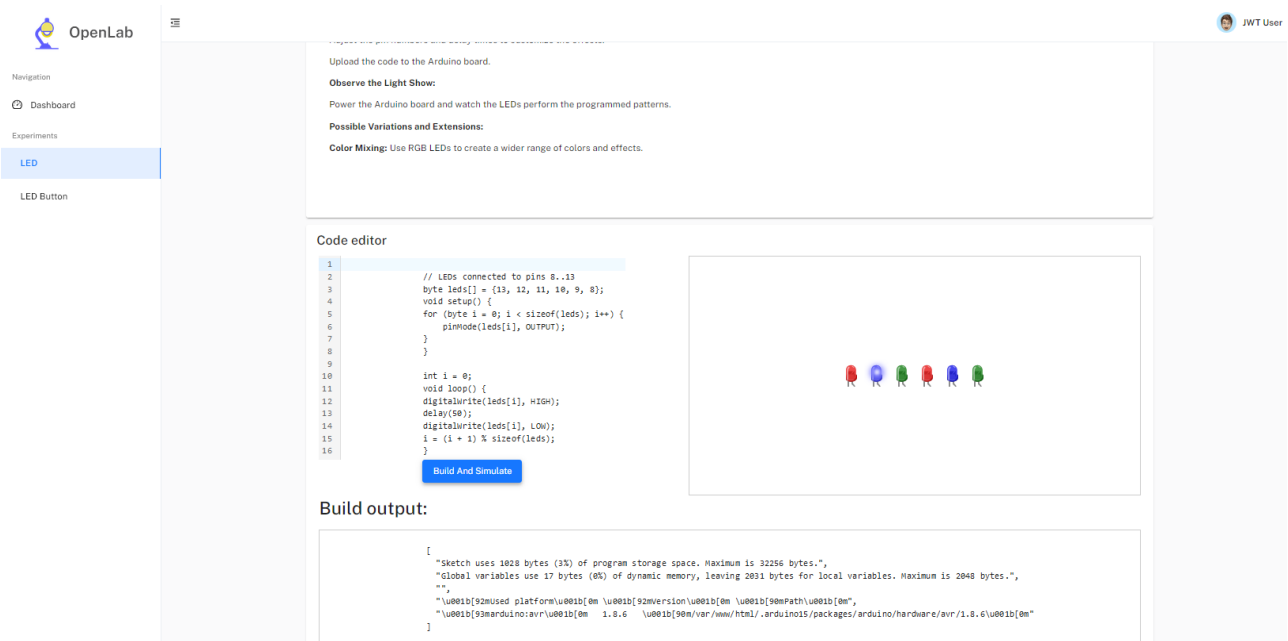


Рисунок 4.37 – Тестування експерименту

4.5 Характеристики системи

Програма виконана за допомогою мов програмування високого рівня PHP і JavaScript у середовищі розробки Visual Studio Code, СКБД MySQL. Для збереження версій використаного забезпечення було використано контейнери Docker. Для розробки дизайну вебсторінок використовувалися файли .css та .html.

Структура програми складається із набору контейнерів (backend, frontend, db) та сервісів, що супроводжуються відповідною файловою системою.

Після запуску серверу користувач може долучитися до сайту через «localhost:5174» (розробник) чи «localhost:5173» (адміністратор) у браузері. Користувачу відкривається головна сторінка сайту. Під час роботи програми відбуваються зміни даних (редагування коду), які паралельно записуються до системи.

4.6 Початкові та вихідні дані

Початкові дані представляють собою коди наявних експериментів ардуїно, шаблонів для них та доступних їх елементів.

Вихідні дані представляють собою змінені шаблони та експерименти, виведені результати білду на вебсторінках і симуляція роботи експериментів за допомогою html-елементів.

4.7 Висновки за розділом 4

У даному розділі було детально розглянуто процес розширення функціоналу програмного продукту та його результати.

Функціональність програмного забезпечення відповідає вимогам технічного завдання (Додаток А).

5 КЕРІВНИЦТВО ОПЕРАТОРА

У даному розділі було розглянуто практичне використання програмного забезпечення, умови та процес її виконання.

5.1 Призначення системи

Програмне забезпечення призначене для практичного використання інструментів віртуальної лабораторії за допомогою різних експериментів.

5.2 Функції системи

Програмне забезпечення здійснює:

- вибір експерименту для дослідження;
- завдання даних експерименту (редагування коду);
- білд коду (його зберігання у скетчах системи);
- вивід результатів роботи коду (успішність та робота Wokwi-елементів).

5.3 Умови виконання системи

5.3.1 Апаратні вимоги системи

Вимоги до комп'ютера, на якому виконується програмне забезпечення, наведено у Додатку А (А.3.4) даної пояснювальної записки.

5.3.2 Вимоги до користувача

До користувачів висуваються наступні вимоги:

- знання української мови (мова документації);
- знання англійської мови (мова інтерфейсу);

- вміння орієнтуватися у коді;
- розуміння мови програмування C, що використовується для ардуїно;
- базове розуміння роботи ардуїно.

5.4 Взаємодія із системою

Користувач має запустити браузер та перейти до сторінки сайту за адресою «<https://client.openlab.org.ua>». У вікні входу до системи необхідно погодитися із поштою та паролем, що встановлені заздалегідь (рис. 5.1).

Сторінку сайту для користувача зображено на рис. 5.2. На ній наведено список наявних експериментів, що представлено у вигляді списку посилань із їх коротким описом.

При натисканні посилання на експеримент, користувач перенаправляється на сторінку цього експерименту.

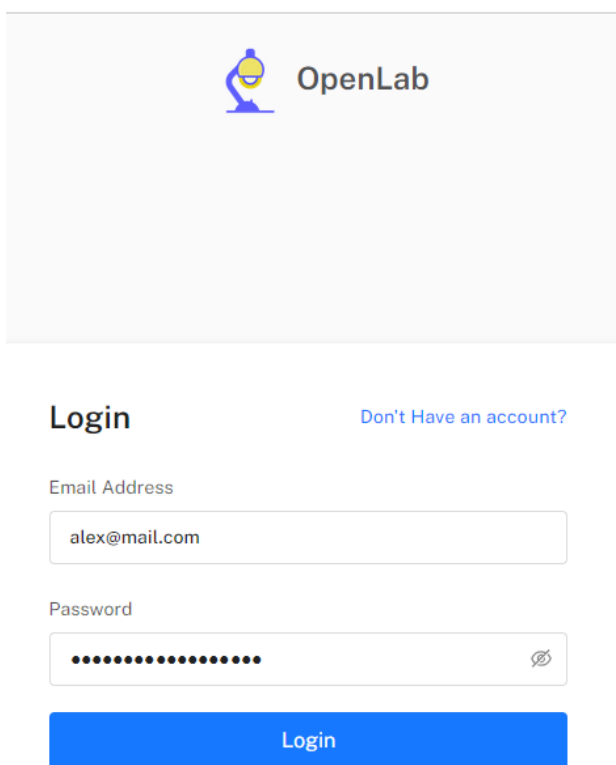


Рисунок 5.1 – Вхід до системи

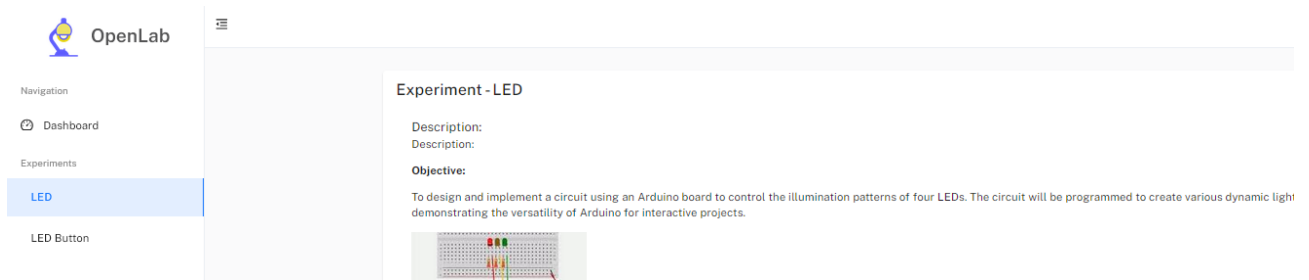


Рисунок 5.2 – Список експериментів

Користувач також може перейти до головної сторінки сайту «<https://openlab.org.ua>», де знаходиться інформація щодо роботи із системою, дані розробників, документація проєкту тощо (рис. 5.3).

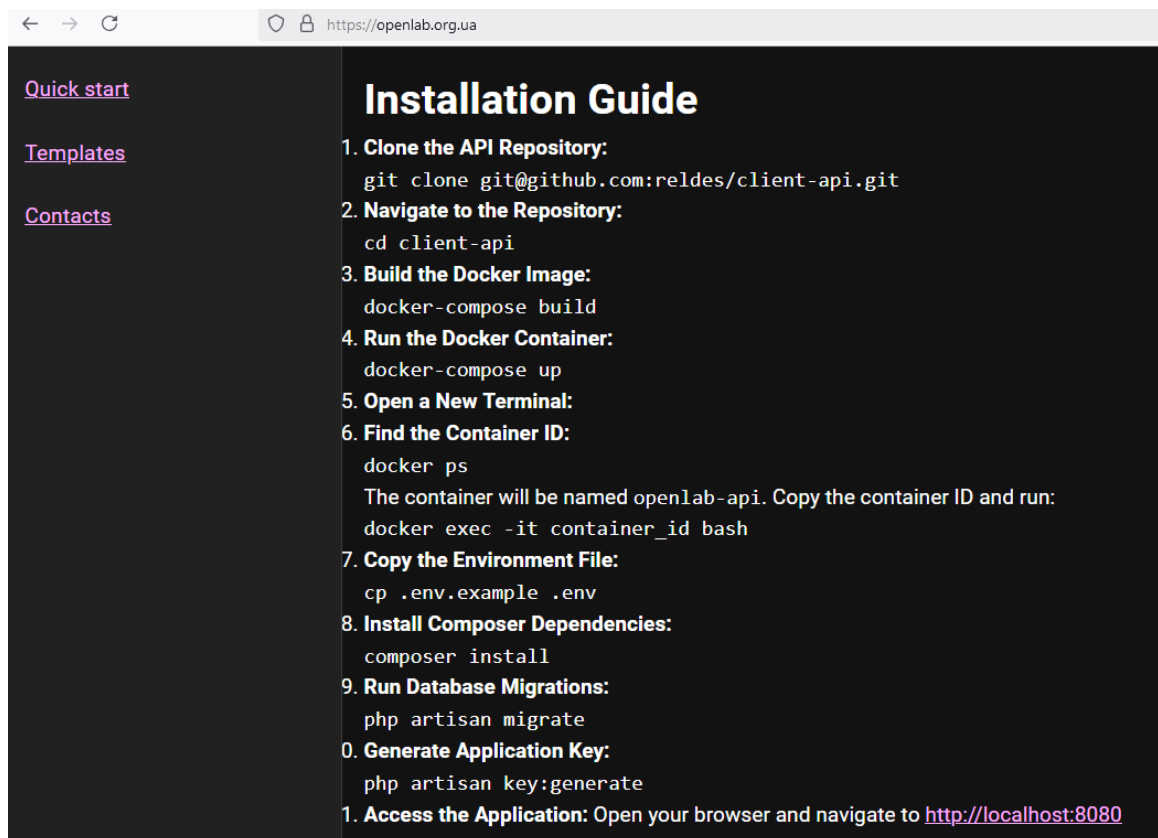


Рисунок 5.3 – Головна сторінка сайту

5.4.1 Взаємодія із LED

При відкритті експерименту користувач бачить опис експерименту, поле редагування коду «Code editor» (що підтримує синтаксис C та в якому можна

змінювати логіку експерименту) із певним усталеним кодом, елементи схеми (у даному випадку 6 світлодіодів різних кольорів – червоного, синього і зеленого), кнопку білду «Build and Simulate» (при натисканні на яку відбувається обробка коду, виводяться результати виконання коду та схема експерименту стає інтерактивною) і поле виведення результатів білду «Build output» (інформація щодо успішності білду та щодо помилок, які виникнули у процесі білду). Експеримент наведено на рис. 5.4.

При використанні усталеного коду після натискання кнопки білду у полі виведення результатів має з'явитися інформаційне повідомлення щодо успішності виконання коду та об'єму пам'яті, витраченого на його обробку, а всі світлодіоди (від 1-ого до 6-ого) починають нескінченно загоратися та потухати один за одним.

Користувач може змінити експеримент за допомогою редагування коду у відповідному полі (змінити порядок включення світлодіодів, час їх горіння, кількість світлодіодів, включених одночасно тощо).

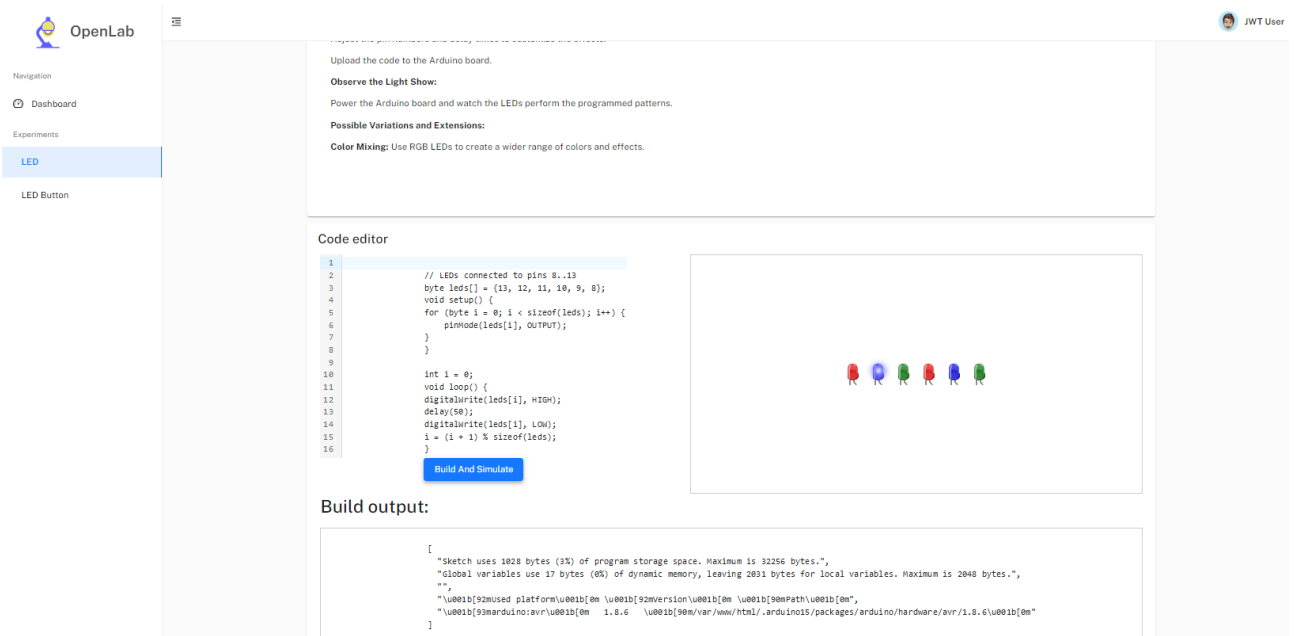


Рисунок 5.4 – Сторінка експерименту LED

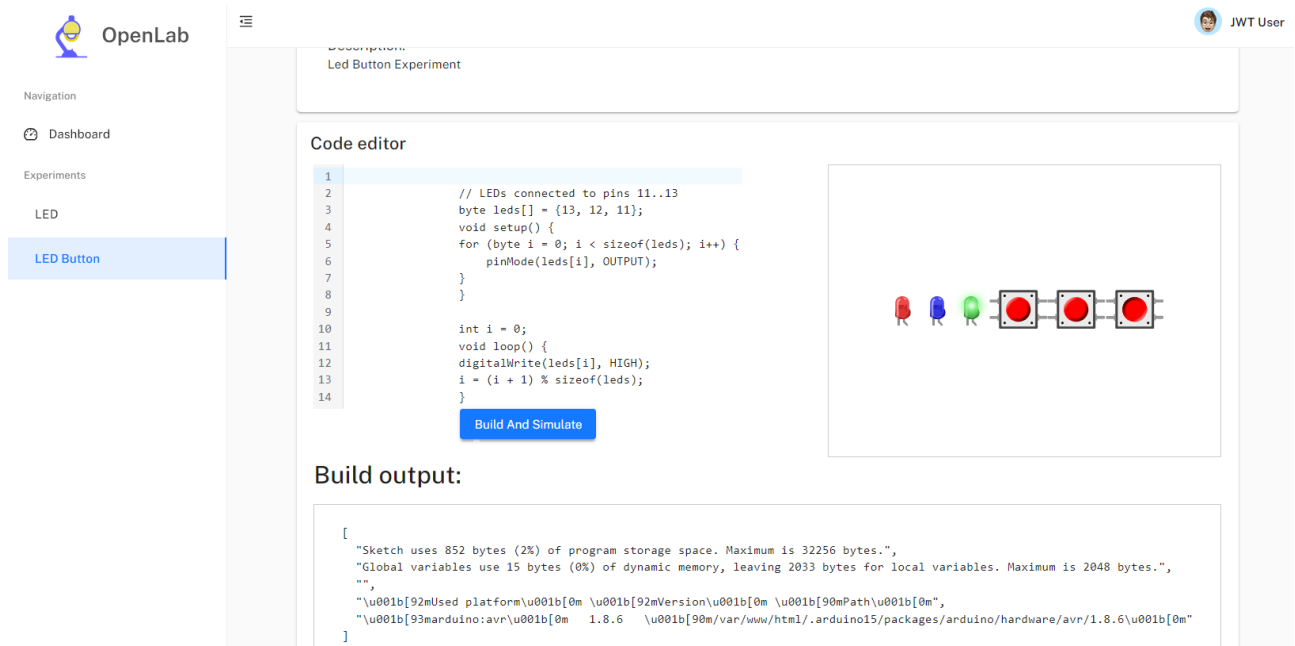
5.4.2 Взаємодія із LED Button

При відкритті експерименту користувач бачить опис експерименту, поле редагування коду «Code editor» (що підтримує синтаксис C та в якому можна змінювати логіку експерименту) із певним усталеним кодом, елементи схеми (у даному випадку 3 кнопки і 3 світлодіоди різних кольорів – червоного, синього і зеленого), кнопку білду «Build and Simulate» (при натисканні на яку відбувається обробка коду, виводяться результати виконання коду та схема експерименту стає інтерактивною) і поле виведення результатів білду «Build output» (інформація щодо успішності білду та щодо помилок, які виникнули у процесі білду). Експеримент наведено на рис. 5.5.

При використанні усталеного коду після натискання кнопки білду у полі виведення результатів має з'явитися інформаційне повідомлення щодо успішності виконання коду та об'єму пам'яті, витраченого на його обробку, при цьому всі світлодіоди залишаються невиключеними.

Якщо користувач натискає та тримає кнопку, то відповідний світлодіод (перший світлодіод – перша кнопка, другий – друга, третій – третя) починає горіти, якщо звільнює – перестає. Експеримент також передбачає, що може бути натиснуто декілька кнопок одночасно (наприклад, якщо сайт відкрит на телефоні).

Користувач може змінити експеримент за допомогою редагування коду у відповідному полі (зробити, що після звільнення кнопки світлодіод потухає; чи що після звільнення кнопки горить, а після повторного натискання тухне; чи що після звільнення кнопки світлодіод не потухає, а коли горять всі світлодіоди – вони одночасно потухають тощо).



OpenLab

Navigation

Dashboard

Experiments

LED

LED Button

JWT User

Led Button Experiment

Code editor

```

1 // LEDs connected to pins 11..13
2 byte leds[] = {13, 12, 11};
3 void setup() {
4   for (byte i = 0; i < sizeof(leds); i++) {
5     pinMode(leds[i], OUTPUT);
6   }
7 }
8
9 int i = 0;
10 void loop() {
11   digitalWrite(leds[i], HIGH);
12   i = (i + 1) % sizeof(leds);
13 }
14

```

Build And Simulate

Build output:

```

[
  "Sketch uses 852 bytes (2%) of program storage space. Maximum is 32256 bytes.",
  "Global variables use 15 bytes (0%) of dynamic memory, leaving 2033 bytes for local variables. Maximum is 2048 bytes.",
  "",
  "\u001b[92mUsed platform\u001b[0m \u001b[92mVersion\u001b[0m \u001b[90mPath\u001b[0m",
  "\u001b[93marduino:avr\u001b[0m 1.8.6 \u001b[90m/var/www/html/.arduino15/packages/arduino/hardware/avr/1.8.6\u001b[0m"
]

```

Visual representation of the LED circuit showing three LEDs (red, blue, green) and three red LEDs connected to pins.

Рисунок 5.5 – Сторінка експерименту LED Button

5.5 Висновки за розділом 5

У п'ятому розділі було розглянуто основний алгоритм роботи оператора із системою та вимоги до користувача, представлені програмним забезпеченням.

Було розглянуто всі наявні елементи системи та можливості, які вони надають.

ВИСНОВКИ

Під час виконання дипломної кваліфікаційної роботи магістра було вирішено актуальну задачу дослідження та практичної реалізації платформи для створення віртуальної лабораторії, що інтегрує набір сучасних технологій розробки програмного забезпечення

Для досягнення мети проєкту було проведено аналіз предметної області та виокремлено основні особливості віддалених, віртуальних та гібридних лабораторій, розроблено основні вимоги до проєкту та обрано інструментарій програмної реалізації.

Наукова новизна роботи полягає в тому, що розроблено платформу для створення віртуальної лабораторії, яка відрізняється від існуючих використовуваним стеком технологій розробки програмного забезпечення та рівнем їх інтеграції.

Практична цінність роботи полягає в тому, що використання розробленої платформи дозволяє підвищити рівень автоматизації процесу розробки віртуальних лабораторій та знизити часові витрати.

У майбутньому планується розширення системи: додавання згенерованої автоматично документації платформи; наповнення платформи експериментами різного типу; підключення реального обладнання для створення віддалених експериментів.

Результати цієї роботи було представлено на науково-технічних та науково-практичних конференціях в Україні та за кордоном [1], [44]-[45].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Research and Practical Implementation of a Framework for Teaching Students the Development of a Virtual Laboratory / [V. Karnaukh,, A. Sokolyanskii, A. Parkhomenko et al.] // Interactive Collaborative Learning : 27th International Conference, Tallinn, Estonia, 24-27 September, 2024 : proceedings. – New York: Springer, 2024. – P. 519–526. (SCOPUS).
2. Sokolianskiy, O. alexsokolianskiy/laboratory [Electronic resource] / O. Sokolianskiy. – Access mode: <https://github.com/alexsokolianskiy/laboratory>.
3. A Flexible Framework for the Deployment of STEM Real Remote Laboratories in Digital Electronics and Control Systems / [R. G. Vera, J. L. L. Galilea, A. G. Vicente et al.] // IEEE Access. – 2024. – Vol. 12. – P. 14563 – 14579.
4. Development of a Remote Laboratory for Engineering Education / [N. Wang, Q. Lan, X. Chen et al.]. – Boca Raton, Florida: CRC Press, 2020. – 98 p.
5. Tinkercad - Create 3D digital designs with online CAD [Electronic resource]. – Access mode: <https://www.tinkercad.com/>.
6. GOLDi - Lab Presentation [Electronic resource]. – Access mode: <https://goldi-labs.net/index.php?Site=2>.
7. Laboratory as a Service (LaaS): a novel paradigm for developing and implementing modular remote laboratories / [M. Tawfik, C. Salzmann, D. Gillet et al.] // International Journal of Online Engineering (iJOE). – 2014.- Vol. 10, №4.- P. 13–21.
8. Online lab infrastructure as a Service: a new paradigm to simplify the development and deployment of online labs / [D. G. Zutin, M. Auer, P. Orduna et al.] // Remote engineering and virtual instrumentation (REV2016) : 13th International conference, Madrid, Spain, 24-26 February 2016 : proceedings. – Los Alamitos: IEEE, 2016. – P. 208–214.
9. Remote laboratory development [Electronic resource]. – Access mode: https://weblabdeusto.readthedocs.io/en/latest/remote_lab_development.html.
10. Implementation of reusable solutions for remote laboratory development

/ [A. Parkhomenko, O. Gladkova, A. Sokolyanskii et al.] // International Journal of Online Engineering (iJOE). – 2016. – Vol. 12, №7. – P. 24-29.

11. Rivera, L. F. The remote laboratory management system (RLMS) pattern (2010) [Electronic resource]. – Access mode: <https://www.hillside.net/sugarloafplop/2018/program/papers/GroupA/12.3.pdf>.

12. Towards a microRLMS approach for shared development of remote laboratories / [P. Orduna, L. Rodriguez-Gil, I. Angulo et al.] // Remote engineering and virtual instrumentation (REV2014) : 11th International conference, Porto, Portugal, 26-28 February 2014 : proceedings. – Los Alamitos: IEEE, 2014. – P.375–381.

13. Tereshchenko, O. Content management system for remote laboratory development / O. Tereshchenko, A. Sokolyanskii, A. Parkhomenko // International research conference : Dortmund International Research Conference 2022, Dortmund, Germany, 1-2 July 2022 : proceedings. – Dortmund, Germany: Fachhochschule Dortmund, 2022. – P. 263–266.

14. Azad, M. K. A. Design and development of remote laboratories with Internet of Things setting / M. K. A. Azad // Advances in Internet of Things. – 2021. – Vol. 11, №3. – P. 95-112.

15. Development of a remote industrial laboratory for automatic control based on node-RED / [M. Domínguez, R. González-Herbón, J. Rodríguez-Ossorio et al.] // IFAC-PapersOnLine. – 2020. – Vol. 53, Issue 2. – P. 17210-17215.

16. Cloud based remote virtual prototyping platform for embedded control applications: Cloud-based infrastructure for large-scale embedded hardware related programming laboratories / [S. Werner, A. Lauber, J. Becker et al.] // Remote engineering and virtual instrumentation (REV2016) : 13th International conference, Madrid, Spain, 24-26 February 2016 : proceedings. – Los Alamitos: IEEE, 2016. – P. 168-175.

17. PHP: Посібник з PHP – Manual [Електронний ресурс]. – Режим доступу: <https://www.php.net/manual/en/>.

18. JavaScript | MDN by MDN contributors [Electronic resource]. – Access

mode: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>.

19. General Python FAQ — Python 3.13.0 documentation [Electronic resource]. – Access mode: <https://docs.python.org/3/faq/general.html>.

20. Documentation for Visual Studio Code [Electronic resource]. – Access mode: <https://code.visualstudio.com/docs>.

21. Getting started | PyCharm Documentation [Electronic resource]. – Access mode: <https://www.jetbrains.com/help/pycharm/getting-started.html>.

22. PhpStorm: The PHP IDE by JetBrains [Electronic resource]. – Access mode: <https://www.jetbrains.com/phpstorm/>.

23. MySQL :: MySQL Database [Electronic resource]. – Access mode: <https://www.mysql.com/products/enterprise/database/>.

24. Introduction to MongoDB - MongoDB Manual v8.0 [Electronic resource]. – Access mode: <https://www.mongodb.com/docs/manual/introduction/>.

25. PostgreSQL: About [Electronic resource]. – Access mode: <https://www.postgresql.org/about/>.

26. КЕШ — це що таке, суть, визначення, види, рівні і приклади [Електронний ресурс]. – Режим доступу: <https://termin.in.ua/kesh/>.

27. Community Edition | Docs [Electronic resource]. – Access mode: <https://redis.io/docs/latest/get-started/>.

28. Memcached Documentation [Electronic resource]. – Access mode: <https://docs.memcached.org/>.

29. Apache Ignite Documentation | Ignite Documentation [Electronic resource]. – Access mode: <https://ignite.apache.org/docs/latest/>.

30. Installation - Laravel 11.x - The PHP Framework For Web Artisans [Electronic resource]. – Access mode: <https://laravel.com/docs/11.x>.

31. What is Symfony [Electronic resource]. – Access mode: <https://symfony.com/what-is-symfony>.

32. Welcome to CodeIgniter4 — CodeIgniter 4.5.5 documentation [Electronic resource]. – Access mode: https://codeigniter.com/user_guide/intro/index.html.

33. Introduction: About Yii | The Definitive Guide to Yii 2.0 | Yii PHP Framework [Electronic resource]. – Access mode: <https://www.yiiframework.com/doc/guide/2.0/en/intro-yii>.

34. React [Electronic resource]. – Access mode: <https://react.dev/>.

35. Introduction | Vue.js [Electronic resource]. – Access mode: <https://vuejs.org/guide/introduction.html>.

36. Welcome to Wokwi! | Wokwi Docs [Electronic resource]. – Access mode: https://docs.wokwi.com/?utm_source=wokwi.

37. OpenAPI Specification - Version 3.1.0 | Swagger [Electronic resource]. – Access mode: <https://swagger.io/specification/>.

38. Difference between BDD vs TDD in Software Engineering – GeeksforGeeks by satyabrata_jena [Electronic resource]. – Access mode: <https://www.geeksforgeeks.org/difference-between-bdd-vs-tdd-in-software-engineering/>.

39. TDD vs. BDD: Detailed Overview and Comparison by Nikola Kostic [Electronic resource]. – Access mode: <https://phoenixnap.com/blog/tdd-vs-bdd>.

40. Test-Driven Development in the Larger Context [Electronic resource]. – Access mode: <https://www.pmi.org/disciplined-agile/how-to-start-with-acceptance-test-driven-development/test-driven-development-in-the-larger-context>.

41. Install Docker Engine | Docker Docs [Electronic resource]. – Access mode: <https://docs.docker.com/engine/install/>.

42. Connecting to GitHub with SSH - GitHub Docs [Electronic resource]. – Access mode: <https://docs.github.com/en/authentication/connecting-to-github-with-ssh>.

43. Welcome to OpenLab [Electronic resource]. – Access mode: <https://openlab.org.ua/>.

44. Дослідження та програмна реалізація платформи для розробки віртуальної лабораторії [Текст] / [В. Д. Карнаух, О. В. Соколянський, А. В. Пархоменко та ін.] // Сучасні проблеми і досягнення в галузі радіотехніки, телекомунікацій та інформаційних технологій : XII Міжнародна науково-

практична конференція, Запоріжжя, 10-12 грудня 2024 р. : тези доповідей. – Запоріжжя: НУ "Запорізька політехніка", 2025. (подано до друку).

45. Карнаух, В. Д. Дослідження фреймворку для розробки віртуальної лабораторії [Текст] / В. Д. Карнаух // Тези доповідей науково-практичної конференції, Запоріжжя, 15–19 квітня 2024 р. : Тиждень науки-2024, Запоріжжя, 15–19 квітня 2024 р. : тези доповідей. – Запоріжжя: НУ "Запорізька політехніка", 2024. – С.88.

ДОДАТОК А
Технічне завдання

A.1 Підстави для розробки

Підставою для розробки програмного забезпечення служить завдання на дипломну кваліфікаційну роботу магістра на тему «Дослідження та програмна реалізація платформи для розробки віртуальної лабораторії», затверджене наказом № 398 від 07 жовтня 2024 р. по НУ "Запорізька політехніка".

A.2 Призначення розробки

Дане програмне забезпечення призначене для дослідження роботи фреймворку для розробки віртуальної та віддаленої лабораторії, розширення його функціоналу, аналіз його використання для навчальних цілей.

A.3 Вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Основними користувачами програмного забезпечення будуть викладачі та студенти, що будуть створювати експерименти, використовувати вже створені експерименти для розуміння роботи схем ардуїно та розроблюваного фреймворку.

Розроблюване програмне забезпечення має підтримувати наступні можливості для розробника:

- взаємодія із меню переключення між експериментами;
- взаємодія із експериментами (використання із усталеним і змінним кодом ардуїно, контроль інтерактивних елементів).

Розроблюване програмне забезпечення має підтримувати наступні можливості для адміністратора:

- взаємодія із меню огляду, створення, редагування та видалення експериментів;

- взаємодія із меню огляду, створення, редагування та видалення шаблонів.

Вхідними даними, необхідними для функціонування програмного продукту, є експерименти, основною частиною яких є код ардуїно та схема елементів, шаблони для експериментів і компоненти для шаблонів. Вихідними даними є інформація щодо успішності білду коду та відповідні зміни в елементах схеми, передбачених експериментами.

A.3.2 Вимоги до надійності

Розроблюване програмне забезпечення має повідомляти користувача, якщо введений код для ардуїно є некоректним. Система має підтримувати одночасну роботу різних експериментів без необхідності редагування коду фреймворку.

A.3.3 Вимоги до експлуатації

Умови експлуатації повинні відповідати вимогам, які пред'являються до технічних засобів. Показники температури, відносної вологості і швидкості руху повітря повинні відповідати оптимальним і допустимим значенням в робочій зоні виробничих приміщень.

A.3.4 Вимоги до складу та параметрів технічних засобів

Серед технічних засобів мають бути наявні:

- операційна система Linux, Windows або macOS;
- ОЗУ більше 4 Гбайт або більше;
- 32 МБ відеопам'яті або більше;
- наявність вільного місця на жорсткому диску більше 512 МБ;
- наявність програмного забезпечення Docker;

- наявність браузера Firefox, Opera або Chrome;
- наявність клавіатури, миші, монітору.

A.3.5 Вимоги до інформаційної та програмної сумісності

Розроблюване програмне забезпечення має підтримувати спільний режим роботи із різними операційними системами (Windows, Linux, macOS, Android) та надавати необхідні версії використаного набору інструментів і технологій (Laravel, Next.js, Nginx тощо) за допомогою контейнерів Docker для однакового розгортання серверу на різних пристроях. Система має бути готова для можливого покращення.

Для реалізації програмного продукту необхідно використовувати мови програмування PHP, JavaScript, мову розмітки гіпертексту HTML та середовище розробки Visual Studio Code.

A.3.6 Вимоги до програмної документації

Включно із розроблюваним програмним забезпеченням повинно бути розроблено керівництва програміста та оператора, а також автоматично генерована документація щодо структури проєкту.

A.4 Стадії та етапи розробки

Розробка та модифікація даного програмного забезпечення повинна включати наступні етапи:

- дослідження та аналіз роботи існуючих фреймворків;
- формулювання завдання та опис архітектури;
- розробка програмного забезпечення та програмної документації;
- тестування програми.

A.5 Порядок приймання і тестування

Тестування даного програмного забезпечення проводить у вигляді виконання основних можливих сценаріїв оператора (взаємодія із сторінками меню та експериментів).

Дане програмне забезпечення вважається прийнятним, якщо воно задовольняє усім висунутим вимогам до нього і успішно пройшло етап тестування методологіями TDD і BDD.

ДОДАТОК Б
Текст програми

Б.1 Код файла client/src/components/templates/Button.vue

```

<template>
  <div class="exp-view">
    <div class="led-editor">
      <codemirror                                v-model="code"
:options="editorOptions"></codemirror>
      <v-btn color="primary" @click="sendCode">Build and simulate</v-
btn>
    </div>

    <div class="led-scheme">
      <!-- Placeholder for LED scheme -->
      <wokwi-led v-for="led in leds" :key="led.id" :color="led.color"
:pin="led.pin" :value="led.value && buttons[led.id].value ? true : false" />
      <wokwi-pushbutton v-for="button in Object.values(buttons)"
:key="button.id" @button-press="clickBtn(button.id)" @button-
release="buttonRelease(button.id)"></wokwi-pushbutton>
    </div>
  </div>
  <h1>Build output:</h1>
  <div class="build-output">
    <pre>{{ buildResult }}</pre>
  </div>
</template>

<script>
import { Codemirror } from 'vue-codemirror';
import { AVRRunner } from '@utils/helpers/avr/execute';
import api from '@utils/helpers/api/api';
import '@wokwi/elements';

export default {
  name: 'Button',
  components: {
    Codemirror
  },
  data() {
    return {
      status: "",
      buildResult: "",
      hex: null,
      avrRunner: null,
      code: "",

```

```

editorOptions: {
  mode: 'javascript',
  theme: 'base16-dark',
  lineNumbers: true
},
code6LED: `
  // LEDs connected to pins 11..13
  byte leds[] = {13, 12, 11};
  void setup() {
    for (byte i = 0; i < sizeof(leds); i++) {
      pinMode(leds[i], OUTPUT);
    }
  }

  int i = 0;
  void loop() {
    digitalWrite(leds[i], HIGH);
    i = (i + 1) % sizeof(leds);
  }`,
leds: [
  { id: 0, pin: 13, value: false, color: 'red' },
  { id: 1, pin: 12, value: false, color: 'blue' },
  { id: 2, pin: 11, value: false, color: 'green' },
],
buttons: {
  0: { id: 0, value: false },
  1: { id: 1, value: false },
  2: { id: 2, value: false }
}
};
},
created() {
  this.code = this.code6LED;
  this.avrRunner = new AVRRunner();
  this.avrRunner.portD.addListener((value) => {
    this.updateLEDs(value, 0);
  });
  this.avrRunner.portB.addListener((value) => {
    this.updateLEDs(value, 8);
  });
},
methods: {
  buttonRelease(id) {
    this.buttons[id].value = false;
  }
}

```

```

    },
    clickBtn(id) {
        this.buttons[id].value = true;
    },
    async sendCode() {
        this.hex = null;
        try {
            const response = await api.experiment.build(
                this.code
            );
            if (response) {
                this.buildResult = response.output;
                this.hex = response.hex;
                this.avrRunner.uploadHex(this.hex || '');

                this.avrRunner.execute(() => {})
                this.status = response.code == 0 ? 'Success' : 'Error';
            } else {
                this.avrRunner.stop();
                console.error('Failed to build and upload:', response.statusText);
                this.buildResult = 'Failed to build and upload';
                this.status = 'Error';
            }
        } catch (error) {
            console.error('Error during API call:', error);
            this.buildResult = 'Error during API call';
        }
    },
    updateLEDs(value, startPin) {
        for (const led of this.leds) {
            const pin = led.pin;
            if (pin >= startPin && pin <= startPin + 8) {
                led.value = (value & (1 << (pin - startPin))) ? true : false;
                this.leds.map((led) =>
                    led.pin === pin ? { ...led, value: value & (1 << (pin - startPin))
? true : false } : led
                )
            }
        }
    }
};
</script>

```

```

<style scoped>
.exp-view {
  display: flex;
}
.led-editor {
  display: flex;
  flex-direction: column;
  align-items: center;
}

.codemirror {
  width: 100%;
  height: 300px;
  margin-bottom: 20px;
}

button {
  margin-bottom: 20px;
}

.led-scheme {
  margin-left: 100px;
  width: 100%;
  border: 1px solid #ccc;
  display: flex;
  justify-content: center;
  align-items: center;
}
.build-output {
  width: 100%;
  border: 1px solid #ccc;
  display: flex;
  height: 200px;
  justify-content: center;
  align-items: center;
}
</style>

```

Б.2 Код файла client/src/views/Experiment.vue

```

<template>
  <v-container>

```

```

<v-row>
  <v-col cols="12">
    <v-card v-if="experiment">
      <v-card-title >
        <span class="headline">Experiment - {{ experiment.title
}}</span>
      </v-card-title>
      <v-card-text>
        <v-list>
          <v-list-item>
            <v-list-item-content>
              <v-list-item-title>Description:</v-list-item-title>
              <div v-html="experiment.description"></div>
            </v-list-item-content>
          </v-list-item>
        </v-list>
      </v-card-text>
    </v-card>
    <v-card v-else>
      <v-card-title>
        <span class="headline">Experiment</span>
      </v-card-title>
      <v-card-text>
        <p>Experiment not found.</p>
      </v-card-text>
    </v-card>
  </v-col>
</v-row>
<v-row>
  <v-col cols="12">
    <v-card>
      <v-card-title>
        <span class="headline">Code editor</span>
      </v-card-title>
      <v-card-text>
        <component :is="experiment.template.component" v-
if="experiment && experiment.template && experiment.template.component" />
      </v-card-text>
    </v-card>
  </v-col>
</v-row>
</v-container>
</template>

```

```

<script>
import api from '@/utils/helpers/api/api';
import LED from '@/components/templates/LED.vue';
import Button from '@/components/templates/Button.vue';

export default {
  components: {
    LED, Button
  },
  data() {
    return {
      experiment: null
    };
  },
  created() {
    this.fetchExperiment();
  },
  watch: {
    '$route.params.id': 'fetchExperiment'
  },
  methods: {
    async fetchExperiment() {
      try {
        const experiment = await
api.experiment.getExperiment(this.$route.params.id);
        this.experiment = experiment;
      } catch (error) {
        console.error('Error fetching experiment:', error);
      }
    }
  }
};
</script>

<style scoped>
.schema-placeholder {
  height: 300px;
  border: 1px dashed #ccc;
  display: flex;
  align-items: center;
  justify-content: center;
  color: #999;
}
</style>

```

ДОДАТОК В
Слайди презентації



Дипломна кваліфікаційна робота магістра

ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ПЛАТФОРМИ ДЛЯ РОЗРОБКИ ВІРТУАЛЬНОЇ ЛАБОРАТОРІЇ

Виконала:
ст. групи КНТ-113м

Вероніка КАРНАУХ

Керівник:
к.т.н., доцент

Анжеліка ПАРХОМЕНКО

Рисунок В.1 – Слайд №1

Актуальність роботи

Джерела		Технології
1	Implementation of reusable solutions for remote laboratory development (A. Parkhomenko et al.)	Методологія повторного використання
2	The remote laboratory management system (RLMS) pattern (L. F. Rivera)	RLMS
3	Towards a microRLMS approach for shared development of remote laboratories (P. Orduna et al.)	
4	Content management system for remote laboratory development (O. Tereshchenko et al.)	Система керування вмістом
5	Design and development of remote laboratories with Internet of Things setting (M. K. A. Azad)	Фреймворки
6	Development of a remote industrial laboratory for automatic control based on node-RED (M. Dominguez et al.)	
7	A flexible framework for the deployment of STEM real remote laboratories in digital electronics and control systems (R. G. Vera et al.)	
8	Cloud based remote virtual prototyping platform for embedded control applications (S. Werner et al.)	Хмарна інфраструктура
9	Development of a Remote Laboratory for Engineering Education (N. Wang et al.)	Хмарні та мобільні технології

Рисунок В.2 – Слайд №2

Мета та завдання роботи

Мета роботи – дослідження та програмна реалізація платформи для розробки віртуальної лабораторії для підвищення рівня автоматизації проектних процедур за допомогою ряду сучасних технологій.

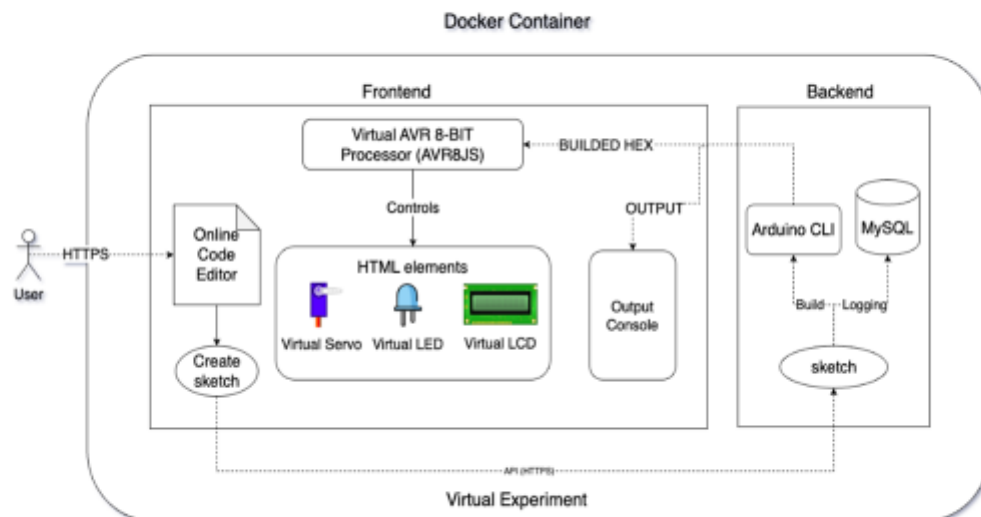
Завдання роботи:

- дослідити структурні та функціональні особливості прототипу платформи;
- виконати аналіз та вибір інструментарію розробки;
- виконати моделювання системи;
- розробити методику створення віртуальної лабораторії;
- виконати практичну реалізацію віртуальної лабораторії з використанням модифікованої платформи.

3 /13

Рисунок В.3 – Слайд №3

Архітектура платформи



4 /13

Рисунок В.4 – Слайд №4

Компоненти платформи

Модулі		Функції
1	Frontend (Vue.js)	HTTP запити до Backend для отримання JSON даних, потрібних для візуалізації користувацького інтерфейсу
		обробка даних і динамічне відображення інтерфейсу із HTML коду
2	Backend (Laravel API)	отримання запитів від Frontend та відправлення JSON відповідей до Frontend
		оброблення запитів через виконання необхідного PHP коду і звернення до MySQL бази даних
3	База даних (MySQL)	зберігання всіх даних, що використовує веб-застосунок
		витягання, додавання, оновлення або видалення даних за допомогою SQL запитів від Backend

5 /13

Рисунок В.5 – Слайд №5

Аналіз та вибір інструментарію розробки

Порівняння мов програмування

	PHP	JavaScript	Python
Компільована/інтерпретована	I	I	I
Мультипарадигмальність	++	+++	+++
Крос-платформність	+	+	+
Типізація	Динамічна	Динамічна	Динамічна
Стандартизованість	+	+	-
Фреймворки	Laravel, Symfony	React, Angular, Vue	Django, Flask

Порівняння середовищ розробки

	VS Code	PyCharm	PhpStorm
Багатоплатформність	+	+	+
Візуалізація даних	Плагіни	Плагіни	Плагіни
Інтерактивний режим	+	+	+
Безкоштовність	+	+-	-

Порівняння СКБД

	MySQL	MongoDB	PostgreSQL
Тип	реляційна	документно-орієнтована	реляційна
Типи даних	дані SQL	неструктуровані дані	різні
Особливість	швидкість, низька затримка	гнучкі дані	складні запити та транзакції
Масштабованість	вертикальна і горизонтальна	горизонтальна	вертикальна
Безпека	+	+-	+
Застосування	веб-додатки, електронна комерція	гнучкі структури даних, неструктуровані або напівструктуровані дані	реляційні дані, складні запити, звіти, аналітика

6 /13

Рисунок В.6 – Слайд №6

Аналіз та вибір інструментарію розробки

Порівняння IMDB

	Laravel	Symfony	CodeIgniter	Yii
Документація	++	-	+	+
Модульність	++	+++	+	+++
Безпечність		Redis	Memcached	Apache Ignite
Підтримка API		рядки, кеш, списки, множини	рядки	кеш, SQL таблиці
Масштабованість	Типи даних			
Застосування	Запис даних на диск	+	-	+
	Швидкість	+	++	+-
	Кластеризація	-		.
	Підтримка інтеграції		React.js	Vue.js
	Застосування	Тип	бібліотека	фреймворк

Порівняння засобів для backend

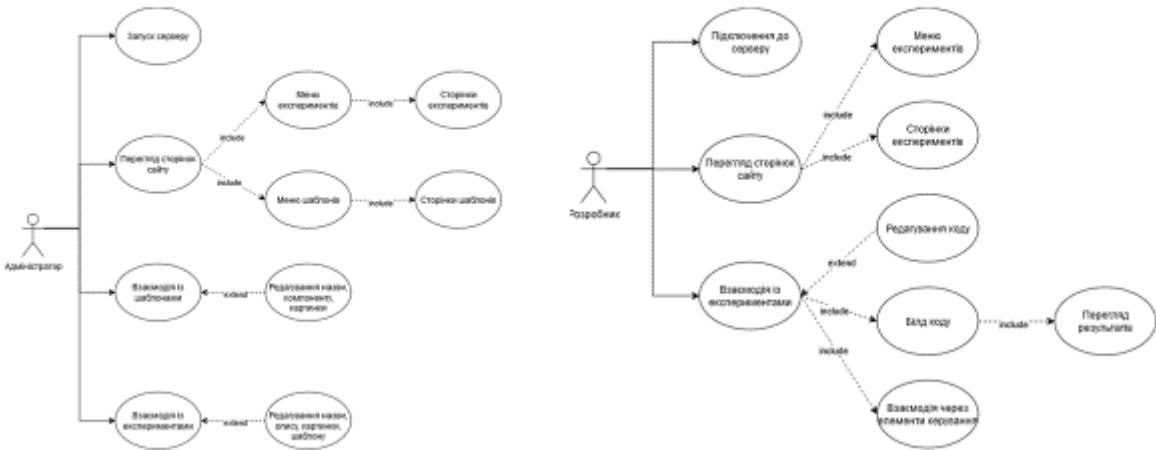
	React.js	Vue.js	Wokwi
Документація	+	+	+-
Тестування	+	+	-
Гручкість	+	+	-
Масштабованість	+	+	-
Навички	JavaScript, JSX	JavaScript	електроніка
Застосування	вебдодатки	додатки	симуляція електроніки

Порівняння засобів для frontend

7 / 13

Рисунок В.7 – Слайд №7

Моделювання системи

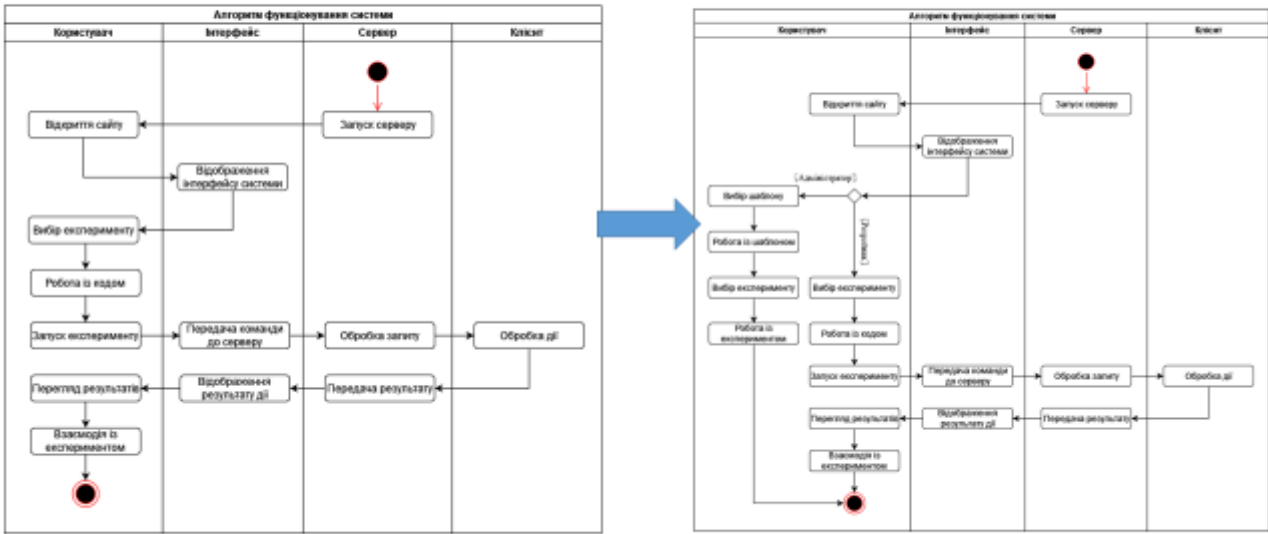


Діаграми прецедентів

8 / 13

Рисунок В.8 – Слайд №8

Модельовання системи

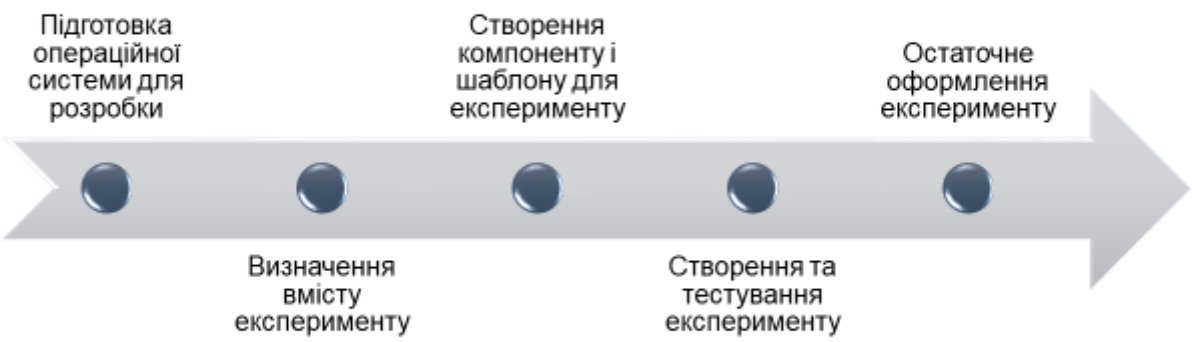


Діаграма активностей

9 /13

Рисунок В.9 – Слайд №9

Методика створення віртуальної лабораторії



10 /13

Рисунок В.10 – Слайд №10

Практична реалізація віртуальної лабораторії



11 /13

Рисунок В.11 – Слайд №11

Висновки

- Наукова новизна роботи полягає в тому, що розроблено платформу для створення віртуальної лабораторії, яка відрізняється від існуючих використанням стеком технологій розробки програмного забезпечення та рівнем їх інтеграції.
- Практична цінність роботи полягає в тому, що використання розробленої платформи дозволяє підвищити рівень автоматизації процесу розробки віртуальних лабораторій та знизити часові витрати.
- У майбутньому планується розширення системи: додавання згенерованої автоматично документації платформи; наповнення платформи експериментами різного типу; підключення реального обладнання для створення віддалених експериментів.

12 /13

Рисунок В.12 – Слайд №12

Перелік публікацій

1. Research and Practical Implementation of a Framework for Teaching Students the Development of a Virtual Laboratory / [V. Kamaukh,, A. Sokolyanskii, A. Parkhomenko et al.] // Interactive Collaborative Learning: 27th International Conference, Tallinn, Estonia, September 24-27, 2024: proceedings. – 2024. – P. 519-526 (SCOPUS).
2. Дослідження та програмна реалізація платформи для розробки віртуальної лабораторії [Текст] / [В. Д. Карнаух, О. В. Соколянський, А. В. Пархоменко та ін.] // Сучасні проблеми і досягнення в галузі радіотехніки, телекомунікацій та інформаційних технологій : XII Міжнародна науково-практична конференція, Запоріжжя, 10-12 грудня 2024 р. : тези доповідей. – Запоріжжя: НУ "Запорізька політехніка", 2025. (подано до друку).
3. Карнаух, В. Д. Дослідження фреймворку для розробки віртуальної лабораторії [Текст] / В. Д. Карнаух // Тези доповідей науково-практичної конференції, Запоріжжя, 15–19 квітня 2024 р. : Тиждень науки-2024, Запоріжжя, 15–19 квітня 2024 р. : тези доповідей. – Запоріжжя: НУ "Запорізька політехніка", 2024. – С.88.

13 /13

Рисунок В.13 – Слайд №13