Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Створення інтерактивного застосунку "Музейний помічник" засобами React 16.13.0, C++ та QML 5.12

Виконав: студент IV курсу, групи СНс-42
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

(підпис) Підбурачинський Р. А.
(прізвище та ініціали)

Керівник (підпис) Млинко Б. Б
(прізвище та ініціали)

Нормоконтроль (підпис) Шимчук Г. В.
(прізвище та ініціали)

Завідувач кафедри (підпис) Боднарчук І.О.
(прізвище та ініціали)

Рецензент (підпис) Осухівська Г. М.
(прізвище та ініціали)

Тернопіль
2022

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«__» _____ 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Підбурачинському Ростиславу Анатолійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Створення інтерактивного застосунку "Музейний помічник" засобами React 16.13.0, C++ та QML 5.12

Керівник роботи к. т. н. доцент кафедри Млинко Богдана Богданівна
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 16 » березня 2022 року № 4/7-161

2. Термін подання студентом завершеної роботи 23 червня 2022р.

3. Вихідні дані до роботи Літературні та інтернет джерела про розробку додатків за допомогою Qt і JavaScript, тестування веб та мобільних застосунків

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1 Аналіз предметної області та постановка задачі на розробку. 1.1 Аналітичний огляд наявного рішення. 1.2 Огляд рішень для написання веб-застосунку 1.2.1 Вимоги до технічних засобів. 1.2.2 Розгляд веб-фреймворків. 1.2.3 Платформи для написання серверних додатків. 1.2.4. Інструментарій для розробки мобільних додатків. 1.2.5. Порівняння веб-фреймворків Для розробки SPA 1.3 Постановка задачі. 1.4 Визначення інформаційних зв'язків. 1.5 Висновок до розділу. 2 Розробка програмного комплексу. 2.1 Стадії та етапи розробки. 2.2 Написання текстів програми. 2.3. Опис та обґрунтування вибору структури та методу організації вхідних та вихідних даних 2.4 Зовнішнє проєктування програми. 2.5 Тестування та налагодження програм. 2.6 Інструкція з інсталяції програмного забезпечення 2.7 Інструкція з використання тестових наборів. 2.8 Інструкція з експлуатації програмного комплексу. 2.9 Висновок до розділу 3 Безпека життєдіяльності та основи охорони праці. 3.1 Долікарська допомога при кровотечах. 3.2 Вплив шуму на організм людини та розробка заходів щодо його зниженню до допустимих величин для обладнання. Висновки. Перелік джерел. Додатки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
Мета задачі. Порівняння. Актуальність теми. Проблема яка потребує вирішення. Вимоги до веб-версії програмного забезпечення. Аналіз інструментів для розробки веб-інтерфейсу. та мобільного застосунку. Qt та Node.js - як інструменти для розробки програмного забезпечення. Основні компоненти. ER-діаграма зв'язків компонентів. Можливості веб-додатку. Можливості мобільного застосунку. Тестування. Результат роботи. Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи хорони праці	Гурик О. Я., доцент		

7. Дата видачі завдання 24 січня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	24.01.2022	*Виконано*
2.	Підбір джерел про отримання інформації для роботи	04.01.2022-30.01.2022	*Виконано*
3.	Переклад та опрацювання джерел про функціонал додатку	31.01.2022-06.02.2022	*Виконано*
4.	Виконання дослідження щодо можливостей роботи додатку	07.02.2022-13.02.2022	*Виконано*
	Розроблення додатку		
5.	Оформлення розділу «Аналіз предметної області та постановка задачі на розробку»	14.02.2022-06.03.2022	*Виконано*
6.	Оформлення розділу «Розробка програмного комплексу»	07.03.2022-03.04.2022	*Виконано*
7.	Виконання завдання до підрозділу «Безпека життєдіяльності»	04.04.2022-17.04.2022	*Виконано*
8.	Виконання завдання до підрозділу «Основи охорони	18.04.2022-01.05.2022	*Виконано*
	праці»		
9.	Оформлення кваліфікаційної роботи	02.05.2022-15.05.2022	*Виконано*
10.	Нормоконтроль	16.05.2022-22.05.2022	*Виконано*
11.	Перевірка на плагіат	09.06.2022	*Виконано*
12.	Попередній захист кваліфікаційної роботи	10.06.2022	*Виконано*
13.	Захист кваліфікаційної роботи	23.06.2022	

Студент _____ Підбурачинський Р. А.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Млинко Б. Б.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Створення інтерактивного застосунку "Музейний помічник" засобами React 16.13.0, C++ та QML 5.12 // Кваліфікаційна робота освітнього рівня «Бакалавр» // Підбурачинський Ростислав Анатолійович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СНс-42 // Тернопіль, 2022 // С. – 60, рис. – 46, табл. – 1, додат. - 19, бібліогр. - 26.

Ключові слова: веб-сайт, застосунок, android, material-ui, база даних, користувач, адміністратор, QR-код, javascript, c++, qt, node.js.

Кваліфікаційна робота присвячена розробці програмного комплексу «Музейний помічник» який складається з веб-застосунку та мобільного додатку для Android який допоможе власникам музеїв покращити взаємодію з відвідувачами.

Метою кваліфікаційної роботи є розробити унікальний додаток для розвитку індустрії та мистецтва який би полегшив взаємодію людей з картинами, експонатами і скульптурами та надав би можливості гіда в автоматизованому додатку.

В першому розділі кваліфікаційної роботи описується аналітичний огляд існуючих рішень, постановка задачі на створення програмного продукту, зокрема наводяться аргументи щодо вибору інструментів розробки, та порівняння інструментів для розроблення програмного комплексу.

В другому розділі кваліфікаційної роботи міститься опис процесу розробки, середовища виконання та методів виконання поставлених задач та роботи у веб-частині, встановлення та експлуатація мобільного додатку.

Розділ охорони праці включає в себе питання про долікарську допомогу при кровотечах та вплив шуму на організм людини та розробка заходів щодо його зниженню до допустимих величин для обладнання.

ABSTRACT

An interactive application "Museum Assistant" design using React 16.13.0, C++ and QML 5.12 // Diploma project // Pidburachynskyi Rostyslav Anatoliyovych // Ternopil Ivan Puluj National Technical University, group SNs-42 // Ternopil, 2022 // pages – 60, figures – 46, tables. – 1, supplements – 19, bibliography – 26.

Key words: web-site, application, android, material-ui, database, user, administrator, QR-code, javascript, c++, qt, node.js.

Qualifying work is dedicated to the development of the software package "Museum Helper" which consists of a web application and a mobile application for Android that will help museum owners to improve interaction with visitors.

The aim of the qualification work is to develop a unique application for the development of industry and art that would facilitate the interaction of people with paintings, exhibits and sculptures and would provide opportunities for a guide in an automated application.

The first section of the qualification work describes an analytical review of existing solutions, problem statement, terms of reference for creating a software product, in particular, arguments for the choice of development tools, and comparison of tools for software development.

The second section of the qualification work contains a description of the development process, execution environment and methods of performing tasks, the processes of setting up and working in the web part, installation and operation of a mobile application.

The section on labor protection includes issues of pre-medical care for bleeding and the impact of noise on the human body and the development of measures to reduce it to acceptable levels for equipment.

СПИСОК СКОРОЧЕНЬ

БД – База даних

СКБД – Система керування базою даних

HTML – Hypertext markup language

CSS - Cascading Style Sheets

JS – JavaScript

МОД – мова опису даних

МОС – мова опису схем даних

МОП – мова опису підсхем даних

АРМ - Автоматизоване робоче місце

ПЕОМ - Персональна електронна обчислювальна машин

SEO - Search engine optimization

SQL - Structured query language

ACID - Atomicity, Consistency, Isolation, Durability

BLOB - Binary Large Object

JSON - JavaScript Object Notation

UUID - Universally Unique Identifier

XML - Extensible Markup Language

OID – Object identifier

DML - Data Manipulation Language

SPA – Single Page Application

API - Application programming interface

HTTP - HyperText Transfer Protocol

CLI – Command Line Interface

ОЦК - об'єм циркулюючої крові

ЗМІСТ

ВСТУП

Актуальність теми. З появою музеїв картини та експонати в них почали зацікавлювати людей, адже саме вони можуть перенести в те минуле, де була написана картина, і вся історія її виникнення, або ж у прикладі з експонатами, те як вони були зроблені використовуючи молоточок та стамеску. Саме це і зацікавлює людей, які відвідують сучасні музеї, щоб побачити роботи художників та скульпторів свого краю чи зарубіжних. Проте не завжди є можливість подивитися на оригінал картини, адже він є тільки один та всі музеї світу хочуть його отримати. Деякі люди долають сотні кілометрів заради того, щоб побачити оригінал картини, який по вигляду не відрізняється від копії, аби пережити і відчути ті почуття і хвилювання, які переживав художник коли її писав. Тому розробка програмного комплексу на цю тематику є актуальною.

Найстаріша публічна колекція мистецтва датується 1471 роком, саме з того часу почали засновуватися музеї. З сучасних музеїв, першим був заснований Британський музей в Лондоні у 1753 рік, проте він був не публічним, з публічних першим сучасних вважається Лувр, саме він містить одні з найрідкісніших картини світу ціна яких переважає за мільйони доларів. Ціна на похід у цей музей становить близько 17 євро, а ціна за гіда від 240 євро.

Сама ж проблема для замовлені гіда залишиться, адже він переважно замовляється на групу осіб де кожна людина буде залежна від всіх та не зможе довго розглядати експонат не віддалившись від групи. Це обмежує людей у часі, щоб розглянути картину або експонат поближче, краще, а деколи й навіть точніше. Зрештою в музеях є інші люди окрім однієї групи з гідом, це означає, що довго стоячи біля одного експонату з групою осіб – ця група буде заважати іншим, і навпаки інші будуть заважати цій групі. Для вирішення цієї проблеми потрібно розробити сервіс, який дозволить не замовляючи гіда, кожній людині окремо спостерігати картину та подеколи й підходити тільки до тих картин, які цікаві їй, а не іншим учасникам групи.

Ця проблема вже існує протягом багатьох років, і для її вирішення все існує всі окрім ідеї. Саме цю проблему вирішить сервіс «Музейній помічник», який

включає в себе завдання інформації про будь-який експонат та забере залежність від гіда та спішки за іншими. Це відкриє простір для розглядання картини чи, можливо, навіть скульптури у більше триваліший час та дозволить переглядати інформацію про картини на потрібній людині мові, адже сервіс є багатомовним і дозволяє створювати переклади на інші мови без додаткових витрат. Це напряму дозволить людям чи сім’ям у навіть різні дні приходити, щоб подивитися на нові картини та почитати про них находячись біля неї без того, щоб замовляти собі гіда.

Мета і задачі дослідження. Метою даної кваліфікаційної роботи освітнього рівня «Бакалавр» є покращення досвіду відвідування музеїв завдяки використанню сучасних інструментів для розробки програмних засобів. Для досягнення поставленої мети потребують вирішення ряд наступних завдань:

– Проаналізувати стан досліджень в галузі опрацювання відомостей щодо процесу роботи музеїв та їхньої діяльності.

– Дослідити стан досліджень в галузі опрацювання відомостей щодо процесу роботи музеїв та їхньої діяльності.

– Провести аналіз засобів для розробки програмного забезпечення.

Практичне значення одержаних результатів. Практично продемонструвати роботу розробленого програмного забезпечення на основі проаналізованої для роботи програмного забезпечення інформації.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ НА РОЗРОБКУ

1.1 Аналітичний огляд наявного рішення

Серед існуючих рішень на поточному технологічному ринку можна виділити компанію «izi.TRAVEL», яка дає можливість створення власних картин та музеїв, додаванню описів та зображень. Великий мінус у вищезгаданому рішенні - це неможливість працювати без підключення до всесвітньої павутини, що одразу відкине бажання в використанні такого сервісу у країнах з малим наданим об’ємом мобільних даних. Вирішити це може Wi-fi у кожному музеї, що в свою чергу може призвести до втрати безпечності в користуванні цією мережею. Всі ж інші проєкти або зберігають все в QR-коді та видають лише текст для показу або посилають на веб-ресурс з цією картиною, що не дозволяє працювати з картинами в поза мережевому режимі.

В результаті провівши аналіз критичних рішень я дійшов висновку, що моя тема є актуальною та може бути використання у реальних комерційних проєктах. Для досягнення мети потрібно розробити програмний комплекс в який входить веб та мобільний застосунки.

1.2 Огляд рішень для написання веб-застосунку

Веб-застосунок складається з набору веб-сторінок які пов’язані між собою за допомогою інтерактивних посилань чи компонентів, який ідентифікується загальним доменним ім'ям і публікується як мінімум на одному веб-сервері.

Веб-сайти зазвичай присвячені певній темі або цілі, наприклад новинам, освіті, комерції, розвагам чи соціальним мережам.

Користувачі можуть отримувати доступ до веб-сайтів на різних пристроях, включаючи настільні комп’ютери, ноутбуки, планшети та смартфони. Додаток, використовуване на цих пристроях, називається веб-браузером.

Всі загальнодоступні веб-сайти разом поділяються на три групи:

– Статичні веб-сайти – сайти які не містять інтерактивних компонентів і призначені здебільшого для відображення постійного контенту та можуть містити прості форми для заповнення.

– Динамічні – сайти які можуть містити будь-який контент, від простих форм до складних таблиць з пошуком та фільтрацією. Компоненти на таких сайтах реагують на дію користувача та динамічно міняють своє наповнення в залежності від закладеної на сайт поведінки, також такі сайти називаються Single Page Application (SPA) через те що всі зміни графічного відображення відбуваються в користувача в браузері.

– Динамічно-статичні – поєднання попередніх двох груп веб-сайтів. Цей тип сайтів дуже схожий до динамічного типу, лише за виключенням того, що всі зміни графічного відображення компонентів відбуваються на сервері та задача браузера лише відобразити їх. У цих сайтів є перевага в тому, що швидкість їхнього завантаження на сторінці користувача вища через те що браузеру вже не потрібно робити ніяких обрахунків.

– Оскільки для написання веб-застосунків використовується мова програмування JavaScript то логічним буде і пошукати варіанти написання веб-сервера на цій ж мові програмування. Для цього існує лише одне рішення – Node.js та його пакетний менеджер npm[9].

Для розробки мобільних застосунків є багато різних вирішень, від «рідних» для кожної операційної системи мови програмування до обгорток які можуть бути написані на будь-якій мові програмування. Для розробки мобільного застосунку було вибрано засіб Qt мовою програмування C++ та з пакетом для графічного інтерфейсу QML.

1.2.1 Вимоги до технічних засобів

В якості вимог програмного комплексу найбільш важливими є наступні:

– швидкість роботи;

– зручність;

– надійність;

– доступність.

Швидкість роботи повинна бути досягнута якнайменшими кількостями запитів до сервера, що дозволить працювати навіть з поганим підключенням до мережі. Зручність повинна проявити себе у якості дизайну для спрощення розуміння та користування програмних комплексом. Надійність – можливість змінювати дані музеїв та картин тільки попередньо ввійшовши в свій профіль за допомогою логіна та пароля. Доступність – можливість з будь-якого пристрою змогти додавати, видаляти та змінювати дані про картини та музеї.

Основними вимогами програмної частини є:

– доступність;

– зручність.

Доступність повинна проявити себе в можливості завантаження додатка на практично будь-який мобільний смартфон[16], а зручність – це адаптивний дизайн який дозволить користувачу довго не мешкатись у використанні.

1.2.2 Розгляд веб-фреймворків

Для розробки динамічних веб-сайтів варто використовувати наступні рішення[22], адже вони розробляються великою кількістю людей та з відкритим вихідним кодом:

– Angular – фронт-енд фреймворк який розробляється під керівництвом компанії Google.

– React – фронт-енд фреймворк який розробляється під керівництвом компанії Meta.

– Vue.js – фронт-енд фреймворк який розробляється виключно його комюніті.

Цей фреймворк написаний мовою програмування TypeScript та міститься із відкритим кодом, що розробляється під керівництвом Google та спільнотою приватних розробників і корпорацій які зацікавлені у розвитку продукту.

Angular — це ретельно переписаний AngularJS. Та після переписання він містить наступні відмінності:

– Містить CLI, що дає змогу розпочати створення нового додатка [23].

– Angular не використовує концепцію "області видимості" або контролерів, як головну архітектурну концепцію він застосовує ієрархію компонентів.

– Angular має інакший синтаксис написання виразів для біндінгу даних властивостей і для біндінгу даних івентів [23].

– Модульність – значна частина основного функціоналу перенесена у модулі.

Angular має такі ES6-можливості:

– Анонімні функції.

– Ітератори.

– Цикли типу For/Of.

– Рефлексія.

– Динамічне завантаження.

– Асинхронна компіляція шаблонів.

– Заміна контролерів та області видимості компонентами та директивами – компонент є директивою з шаблоном.

– Ітеративні колбеки завдяки використанню RxJS.

React дозволяє розробникам створювати великі веб-програми, що використовують дані, що змінюються з часом, без перезавантаження сторінки. Його мета полягає в тому, щоб бути швидким, простим, масштабованим. React обробляє лише інтерфейс користувача в додатках. Це відповідає вигляду шаблону модель-вид-контролер, і може бути використане у поєднанні з іншими бібліотеками JavaScript.

React також може бути використаний в основі надбудов, щоб піклуватися про частини без інтерфейсу користувача побудови веб-додатків. Як бібліотеку інтерфейсу користувача React найчастіше використовують разом з іншими бібліотеками, такими як Redux[8].

Особливості:

– Одностороння передача даних.

– Віртуальний DOM.

– Вирази JavaScript.

– Не лише рендеринг HTML в браузері.

– Методи життєвого циклу.

– Вкладені елементи.

– Атрибути.

– Умовні вирази.

Vue - це JavaScript-фреймворк що використовує шаблон MVVM для створення інтерфейсів користувача на основі моделей даних [27], через реактивне зв'язування даних.

Особливості:

– Шаблони.

– Реактивність.

– Переходи.

– Роутинг.

1.2.3 Платформа для написання серверних додатків

Node.js – це платформа з відкритим кодом для виконання високопродуктивних мережевих програм, написаних мовою JavaScript. Якщо раніше JavaScript використовувався для обробки даних у браузері користувача, то Node.js надав можливість виконувати JavaScript-скрипти на сервері та надсилати користувачеві результат їхнього виконання.

Платформа Node.js перетворила JavaScript на мову загального користування з великою спільнотою розробників. Особливості:

– Потоковість – Node.js немає потоків, весь код виконується в одному потоці використовуючи цикл подій.

– Управління пакетами – в додаток до Node.js при встановленні додається зручний пакетний менеджер який вільно можуть використовувати розробники.

– Уніфікований API – оскільки найкращим форматом передачі даних є JSON – який поширено використовується в JavaScript то всі модулі реалізовують

цю підтримку.

– Цикл подій – оскільки в Node.js немає потоків, то потрібен інструмент для роботи з паралельними обчисленнями, що і надає цикл подій.

– Нативні прив'язки – Node.js також надає можливість написання підпрограм використовуючи мову програмування C++.

1.2.4 Інструментарій для розробки мобільних застосунків

Qt – крос-платформний інструментарій розробки програмного забезпечення мовою програмування C++. Дозволяє запускати написане з його допомогою на більшості сучасних операційних систем, просто компілюючи текст програми для кожної операційної системи без зміни сирцевого коду.

Містить всі основні класи, які можуть знадобитися для розробки прикладного програмного забезпечення, починаючи з елементів графічного інтерфейсу і закінчуючи класами для роботи з мережею, базами даних, OpenGL, SVG та XML. Бібліотека дозволяє керувати потоками, працювати з мережею та забезпечує крос-платформний доступ до тимчасових файлів та файлів операційної системи з використанням вбудованих можливостей кожної з підтримуваних операційних систем.

В поставці Qt є «Qt Linguist» — могутня графічна утиліта, що дозволяє спростити локалізацію й переклад вашої програми багатьма мовами, та «Qt Assistant» — довідкова система Qt, що спрощує роботу з документацією для бібліотек і дозволяє створювати крос-платформову довідку для ПЗ, розробленого на основі фреймворку Qt[15].

1.2.5 Порівняння веб-фреймворків для розробки SPA

Порівняння інструментів для використання є важливим етапом перед початком роботи на проєктом, адже від цього залежить багато чинників які варто прийняти до уваги, адже потім зміни інструменту може виявитися дорогим задоволенням і не залишиться іншого варіаунто окрім як підтримувати наявне

рішення, що призведе до гіршої розробки додатку програмістами та погіршить співпрацю з користувачами додатку через створення великої кількості багів у системі через це[14].

Нижче наведена порівняльна таблиця 1.1 для вибору фреймворку для розробки інтерактивного веб-додатку динамічного типу з використанням веб-технологій HTML, CSS та JavaScript зі стандартом роботи EcmaScript2020.

Таблиця 1.1 – Порівняння характеристик веб-фреймворків

Властивості	**Angular**	**React**	**Vue**
Підтримка TypeScript	З коробки	Потрібно додатково налаштовувати	Потрібно додатково налаштовувати
Наявність CLI	Так	Лише для створення	Так
Шаблони розробки	HTML + JS	JSX	HTML + JS
Поріг входу	Дуже високий	Низький	Низький
Зв'язування даних	В обидві сторони	Лише в сторону відображення	В обидві сторони
Стиль написання коду	ООП	Функціональне програмування	Функціональне програмування
Швидкість розробки додатків	Довго	Швидко	Швидко
Легкість підтримку коду	Середньо	Легко	Легко
Набір бібліотек для роботи	З коробки	Потрібно встановлювати власноруч	З коробки

Виходячи з порівняння описаного вище можна дійти висновку що:

– Angular – варто використовувати для написання додатків високого рівня Enterprise.

– React – варто використовувати для навчання та написання додатків не високого рівня.

– Vue – ідентичний до React, але в цей ж час дуже жваво розвивається[11].

Оцінивши інструменти які є у вільному доступі було прийнято рішення розробляти програмний комплекс використовуючи наступні інструменти:

– React версії 16.13.0[7] для написання інтерактивного веб-застосунку за допомогою мови програмування JavaScript та пакету з готовим набором графічних компонентів material-ui[9];

– Node.js версії 14.0.0 для написання серверної частини додатку[12] використовуючи мову програмування JavaScript, бібліотеку для обробки http-запитів express та базу даних SQLite[14];

– Qt C++/QML версії 5.12 та 2.12 відповідно для розробки мобільного застосунку для операційної системи Android[16] версій вище 4.3.

1.3 Постановка задачі

Метою розробки є створення унікального додатку для розвитку індустрії мистецтва та їхніх представників – музеїв. Полегшити взаємодію людей з картинами, експонатами та скульптурами.

Для досягнення даної мети, необхідно сформувати та вирішити наступні задачі:

– розробити додаток з яким будуть взаємодіяти користувачі, в якому буде відображатися всі інформація потрібна користувачам;

– створити серверну частину з якою буде взаємодіяти додаток для отримання інформації про музей, картини тощо;

– розробити веб-додаток в якому будуть наповнюватися музеї, картини, експоната та інша потрібна користувачам інформація;

– провести тестування розробленої системи на відповідність

поставленим вимогам.

Даний проєкт повинен забезпечити роботу програмного комплексу по полегшенню отримання інформації про картину в музеї користувачами.

В даний комплекс входитиме веб-частина для підтримки і обслуговування програмної системи, введення інформації про картини і музей, та мобільний додаток, який забезпечить зв'язок для отримання інформації про картину та зображення які до неї були прикріплені.

1.4 Визначення інформаційних зв’язків

Важливим етапом перед початком роботи над проєктом є визначення інформаційних зв’язків, адже саме це може допомогти краще зрозуміти ціль продукту та зразу побачити недоліки наявно спроєктованої системи та змінити її. Також це хороший спосіб передати думки іншим учасником команди при розробці, таким як: тестувальники, аналітики, менеджери та програмісти.

В разі недоцільності розробки певного компоненту системи можна тут ж це побачити і забрати з системи тим самим зменшивши час на розробку, зменшивши бюджет як вимагається для розробки цього додатку та деколи навіть зменшення кількості людей які будуть залучені до роботи над проєктом.

Інформаційним зв’язком у веб-частині вважається зв'язок компонентів на сторінці зі сховищем, детальніше наведено на рисунку 1.1

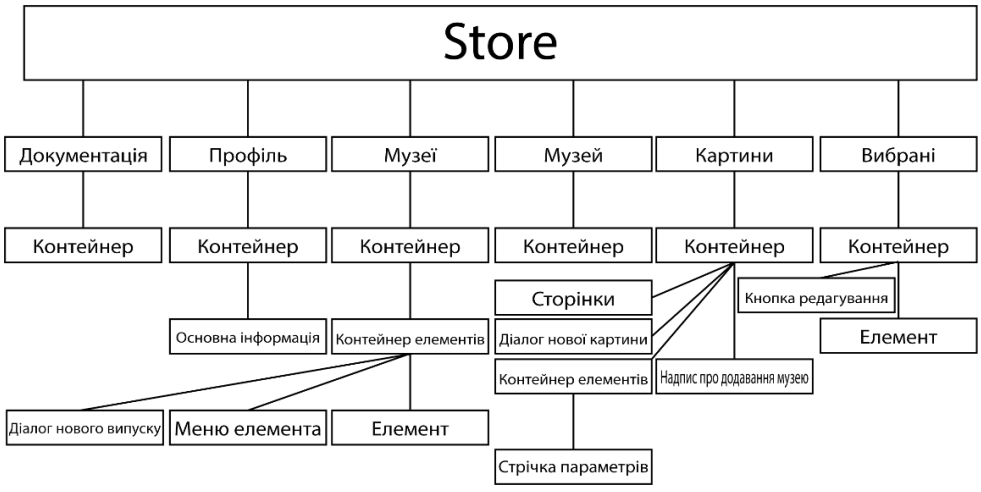


Рисунок 1.1 – Структура інформаційних зв'язків веб-частини

Інформаційним зв’язком у додатку є зв'язок компонентів відображення зі зв’язком описаних об’єктів у розділі 2.2, цей зв'язок зображений на рисунку 1.1.

Всі зв’язки описуються кругами де зовнішні круги – компоненти які відображаються на екрані, а внутрішні – об’єкти які керують логікою програми.

Взаємодія описується стрілками, де зворотній напрямок стрілки показує залежність, а двосторонній показує залежність один від одного.

Об’єкти створені за шаблоном програмування – Singleton, який дозволяє створити один екземпляр на всю програму та брати доступ до нього з любої точки програми. За цим шаблоном створено наступні об’єкти:

– logic;
– settings;
– qrcodeAnalyzer;
– pictures;
– museums;
– savedMuseum;
– currentPicture;
– currentMuseum.

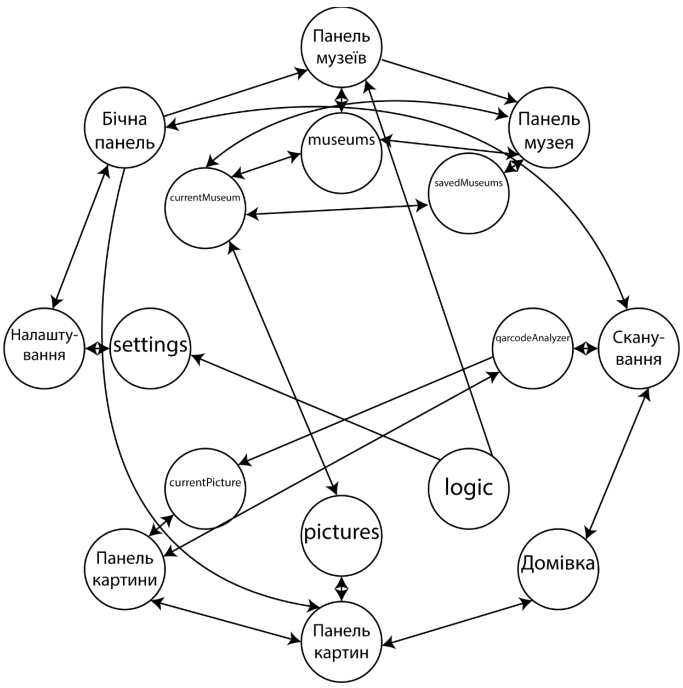


Рисунок 1.2 – Структура інформаційних зв'язків додатку

На рисунку 1.2 зображено структуру зв’язків додатку та з цього можна побачити як повинна виглядати структура додатку та це повинно бути враховано при проєктуванні його та це є хороший спосіб подивитися на додаток ще до його фактичної розробки та при потребі змінити щось.

1.5 Висновок до розділу

В розділі розглянуто наявні рішення інших компаній які надають схожий функціонал, аналітично розглядаються інструменти які доступні розробникам які хочуть розробити схожий проєкт та детально описано інструменті та технології які використовуються при розробці даного програмного комплексу.

Здійснено постановку задачі та визначено інформаційні зв’язки з обґрунтуванням їхньої структури.

РОЗДІЛ 2. РОЗРОБКА ПРОГРАМНОГО КОМПЛЕКСУ

2.1 Стадії та етапи розробки

Розробка програмного забезпечення включає різні стадії, від стадії розгляду ідеї як сутності для проєкту до стадій розробки, тестування та впровадження у роботу додатку з наступною підтримкою наявного коду та побудови поверх нового функціоналу не ламаючи поточного.

Розробка програмного комплексу «Музейний помічник» буде включати в себе наступні стадії:

– аналіз;
– проєктування;
– розробка;
– тестування;
– випуск.

Завданням стадії аналізу є опис поставленої задачі та можливі виході у її вирішенні. Тут аналізується область задачі яка включає в себе декілька факторів: визначення предметної області задачі, призначення вирішення задачі та пошук найкращих рішень. У результатах аналізу описується вибране рішення.

Завданням стадії проєктування є створення структури об’єктів, модулів та їх поведінки, продумування засобів збереження даних, створення алгоритмів, упакування та спосіб випуску продукту.

Завданням стадії розробки є розроблення всіх дій описаних в вище зазначених стадіях, поєднання їх між собою та вирішення всіх поступивших проблем.

Завданням стадії тестування є проведення всіх можливих тестів для запобіганню випуску продукту з непередбачуваною поведінкою у всіх можливих випадках.

Завданням стадії випуску є повне підтвердження готовності продукту до виходу на ринок та його роботи.

2.2 Написання текстів програми

Застосування класів у мобільному додатку:

– Logic – виконує загальні функції по керування завантаженими музеями та їх розприділення, доступний у всьому інтерфейсі.

– QRCodeAnalyzer – керує станом програми на найвищому рівня при та після сканування QR-коду, реалізовує шаблон програмування – Singletone, доступний на сторінці сканування.

– QRCodeAnalyzerDecoder – клас-обгортка для бібліотеки по розпізнаванню QR-коду.

– DBC (Database controller) – для взаємодії з базою даних, реалізовує шаблон програмування – Singletone.

– MuseumObject – для керування сторінкою музею.

– PictureObject – для керування сторінкою з картиною.

– Settings – для збереження та завантаження налаштувань з сховища, а також для отримання та змінення значення певного налаштування, реалізовує шаблон програмування – Singletone, доступний у всьому додатку.

– NetworkManager – статичний клас для роботи з сервером.

– Picture – клас для зберігання зовнішніх даних про картину.

– PictureInfo – клас для зберігання всієї інформації про заголовки та описи на різних мовах, також і самі мови якими перекладена ця картина.

– Museum – клас для зберігання зовнішньої інформації про музей.

– BigMuseum – клас для зберігання всієї інформації про музей.

– PicturesModel – клас для отримання картин у їхньому порядку на сторінці вибору картини.

– MuseumsModel – клас для отримання музеїв у їхньому порядку на сторінці вибору музеїв.

– SavedMuseumsModel – клас для отримання музеїв у їхньому порядку на сторінці вибору збережених музеїв для майбутньої взаємодії з кожним з них при виборі будь-якого з них.

– PictureIcon – клас для роботи з зображеннями у інтерфейсі.

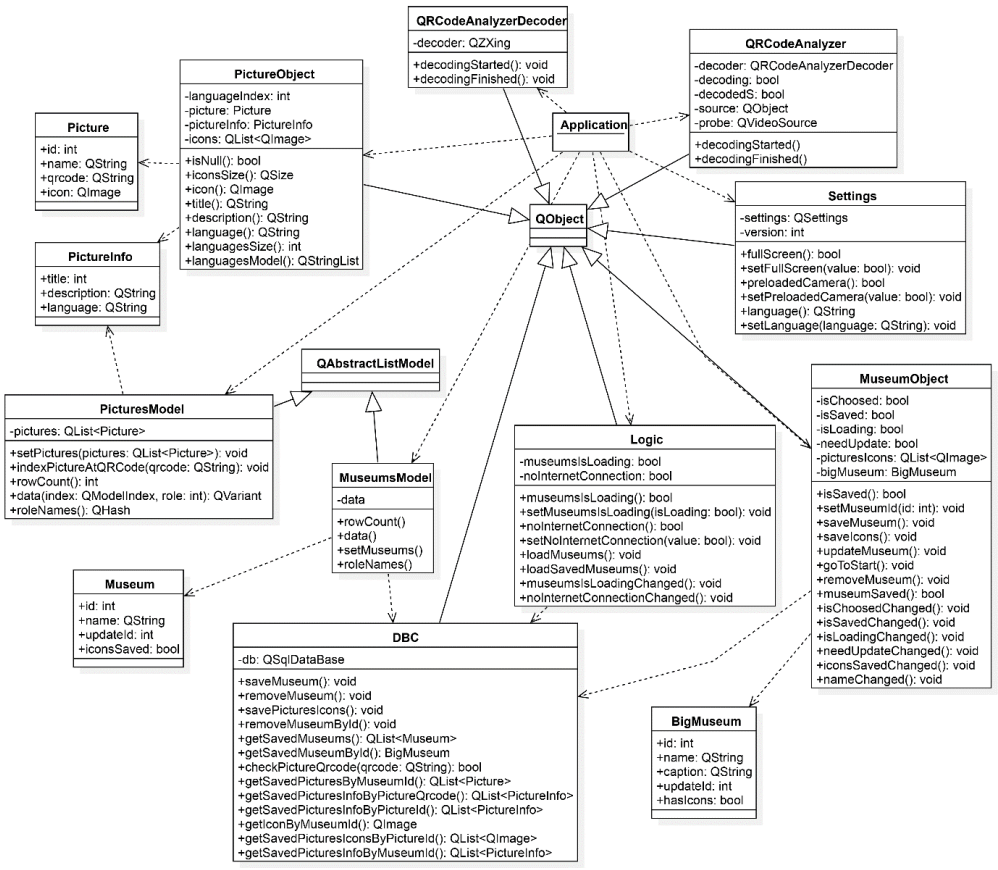


Рисунок 2.1 – UML-діаграма класів

Серверна частина була поділена на дві частини: для отримання сайту та для роботи з api самого сервера. Доступ до api сайт та додаток отримують по одному під домену “app”. Сервер реалізує наступне функціонування:

– Отримання розмітки та файлів сайту – subdomains/index.js.

– Робота з api – subdomains/api.index.js.

Під домен api включає в себе наступні шляхи які співпадають з назвами директорій:

– app – для роботи з додатком (див. дод. Г), включає в себе такі під шляхи:

– getMuseums – для отримання музеїв по шаблону імені;

– getMuseum – для отримання інформації по конкретному вибраному музеї;

– getPictures – для отримання всіх картин у конкретному музеї.

Для аутентифікації користувача використовується login компонент (див. дод. Е), включає в себе наступні під шляхи:

– loginIn – для аутентифікації в системі, тут ж користувачу

привласнюється унікальний сесійний токен;

– verifyEmail – для підтвердження e-mail адреси;

– verifyEmailAgain – для підтвердження адреси після невдалої спроби першого підтвердження.

Для реєстрації користувача використовується register (див. дод. И), включає в себе наступні під шляхи:

– registerIn – керує перевіркою даних про реєстрацію користувача та реєструє його в базі даних.

В свою чергу компонент museums керує усіма запитами щодо музеїв (див. дод. Є), включає в себе наступні під шляхи:

– getMuseum – для отримання інформації по конкретному музеї;

– getMuseums – для отримання інфомації по всіх музеях:

– changeMuseumData – змінення даних музею;

– addMuseum – для додавання нового музею;

– removeMuseum – для видалення музею;

– newReleaseMuseum – для створення новий випуск музею.

– pictures – керує зовнішньою інформацією про картини (див. дод. З), включає в себе такі під шляхи:

– getPicturesData – для отримання інформації по картинах з фільтром;

– deletePicture – для видалення картини;

– addPicture – для додавання нової картини без інформації.

– picture – керує внутрішньо інформацією по картині (див. дод. Ж), включає в себе наступні під шляхи:

– getPictureData – для отримання інформації по картині;

– savePictureInfo – для збереження випускної інформації для користувачів;

– addPictureInfo – для додавання нової локалізації в картині;

– removePictureInfo – для видалення локалізації по картині;

– savePictureData – для збереження розробницької інформації по картині;

– addIconToPicture – для додавання зображення до картини;

– deleteIconFromPicture – для видалення зображення з картини.

– favorites – для керуванням вибраних картин (див. дод. Д), включає в себе наступні під шляхи:

– getFavorites – для отримання всіх вибраних картин у вибраних групах;

– saveFavorites – для збереження вибраних картин та груп;

– addPictureToFavorites – для додавання картини до «вибраних», додається до невизначеної групи;

– deletePictureFromFavorites – для видалення картини з «вибраних»;

– addFavoriteGroup – для видалення групи у «вибраних»;

– deleteFavoriteGroup – для видалення групи з «вибраних».

Для роботи з інформацією про користувача використовується компонент user (див. дод. І), включає в себе наступні під шляхи:

– changeUserData – для змінення інфомації про користувача;

– unlogin – для видалення сесійного токену з бази даних, що призведе до виходу з системи;

– getData – для отримання інформації про поточного користувача.

Також кожен з шляхів містить функції для роботи з базою даних, всі вони написані під конкретну реалізацію шляху та обробляються різними обробниками.

Верстка була здійсненна за рекомендаціях від Google [6][11] та Mozilla[19] з виконанням всіх прописаних до цього вимог та дотримання правил при роботі з елементами верстки.

2.3 Опис та обґрунтування вибору структури та методу організації вхідних та вихідних даних

В якості вхідних даних виступають:

– для веб-частини – дані про картини та музеї;

– для додатку – QR код або ж вибір картини.

Для введення вхідних даних у веб-частині застосована структура вкладок, яка складається з:

– музеї;

– картини;

– вибрані картини.

На вкладці музеї є можливість перейти на сторінку музею, яка зберігає в собі інформацію про музей, яку можна редагувати. На вкладці картин присутні картини вибраного з випадаючого списку музею та можливість пошуку, фільтру та сортування по назві та опису. У вкладці «Вибрані картини» присутні картини, які були позначені як «Вибрані», що легко та швидко отримати до них доступ.

Вихідними даними вважаються дані, як результат інформацію про картини. У додатку вхідними даними є критерії пошуку музею, та пошуку картин у музеї за допомогою сканування QR коду або вибору з усіх картин.

2.4 Зовнішнє проєктування програми

Веб-частина програмного комплексу містить наступні основні функції:

– tr – функція для забезпечення перекладу певної стрічки, яка в якості аргументів приймає шлях у вигляді розділів розділених крапками, наприклад: «documentation.howToUse.title».

– setStore – для вказання місця збереження даних та доступу до них з будь-якого компонента.

Додаток містить наступні основні функції:

– load – функція для завантаження налаштувань користувача з .ini файлу;

– save – функція для збереження налаштувань користувача в .ini файл у папці програми;

– postSend – функція для відправлення POST-запиту на сервер з заготовленими параметрами;

– putSend – функція для відправлення PUT-запиту на сервер з заготовленими параметрами;

– getSend – функція для відправлення GET-запиту на сервер з заготовленими параметрами;

– startDecoding – фунція для запуску розпізнавання QR коду з камери;

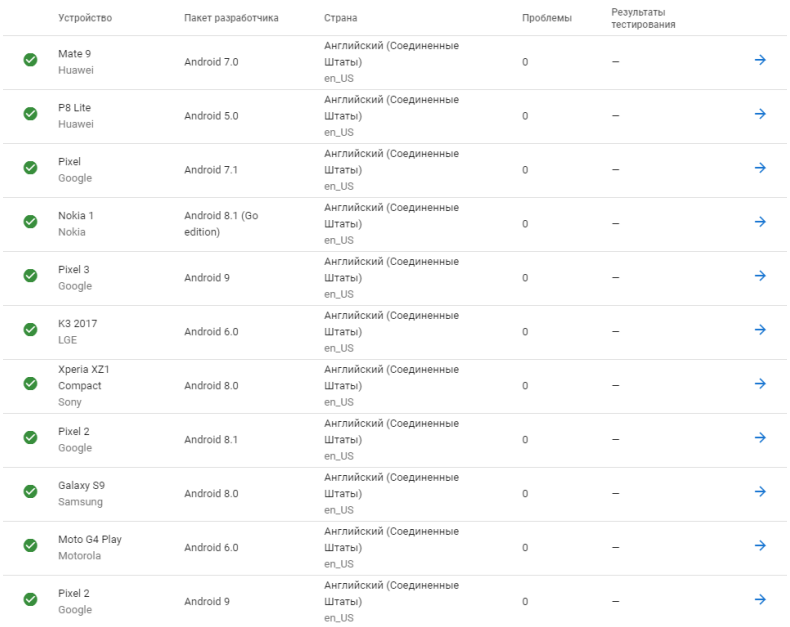
– finishDecoding – фунція для завершення розпізнавання QR коду з

камери з певним кодом в незалежності від того прив’язаний він до картини чи ні.

2.5 Тестування та налагодження програм

Для тестування веб-частини використовувався модуль Cypress [17], який дозволяє тестувати графічну частину на різні ситуації, наприклад відключення від мережі. Все інше тестувалось після завершення написання коду.

Тестування додатку виконувалось за допомогою альфа-тестування Play Console [20]. Результат тестування зображено на рисунку 2.2



Устройство	Пакет разработчика	Страна	Проблемы	Результаты тестирования
Mate 9 Huawei	Android 7.0	Английский (Соединенные Штаты) en_US	0	–
P8 Lite Huawei	Android 5.0	Английский (Соединенные Штаты) en_US	0	–
Pixel Google	Android 7.1	Английский (Соединенные Штаты) en_US	0	–
Nokia 1 Nokia	Android 8.1 (Go edition)	Английский (Соединенные Штаты) en_US	0	–
Pixel 3 Google	Android 9	Английский (Соединенные Штаты) en_US	0	–
K3 2017 LGE	Android 6.0	Английский (Соединенные Штаты) en_US	0	–
Xperia XZ1 Compact Sony	Android 8.0	Английский (Соединенные Штаты) en_US	0	–
Pixel 2 Google	Android 8.1	Английский (Соединенные Штаты) en_US	0	–
Galaxy S9 Samsung	Android 8.0	Английский (Соединенные Штаты) en_US	0	–
Moto G4 Play Motorola	Android 6.0	Английский (Соединенные Штаты) en_US	0	–
Pixel 2 Google	Android 9	Английский (Соединенные Штаты) en_US	0	–

Рисунок 2.2 – Звіт з тестування додатку з Google Play Console

Звіт містить в собі інформацію про пристрій на якому це було виконано, версію системи, країну та кількість виявлених проблем. На всіх пристроях не було знайдено жодної проблеми.

Потрібно зауважити що при кожному тестуванні не було знайдено жодної помилки в роботі додатку та перед тим як дозволити використовувати цей додаток в Play Market він має бути протестований вручну підтримкою ресурсу куди завантажується додаток.

2.6 Інструкція з інсталяції програмного забезпечення

Веб-частина програмного комплексу не потребує встановлення та працює прямо у браузері, веб-сторінка якої буде завантажуватися з сервера[13]. Сервер запускається у середовищі Node [4], і не потребує інсталювання, а лише простого копіювання файлів. Для встановлення додатку потрібно отримати його *.apk файл, наприклад у сервісі Play Market.

Інструкція з інсталяції серверної частини:

1. Встановити node.js з офіційного сайту, версію вибирати 12 або вище (див. рис. 2.3 – 2.14).
2. Завантажити вихідний код сервера в будь-яку директорію.
3. Встановити всі залежності командою *npm install [10]*.
4. Запустити сервер використовуючи команду node та вказавши повний шлях до файлу index.js який знаходиться в кореневій директорії.
5. Для перезапуску сервера потрібно зупинити його роботу використовуючи комбінацію клавіш *Ctrl+C* та знову ввести команду node та шлях до файлу index.js.

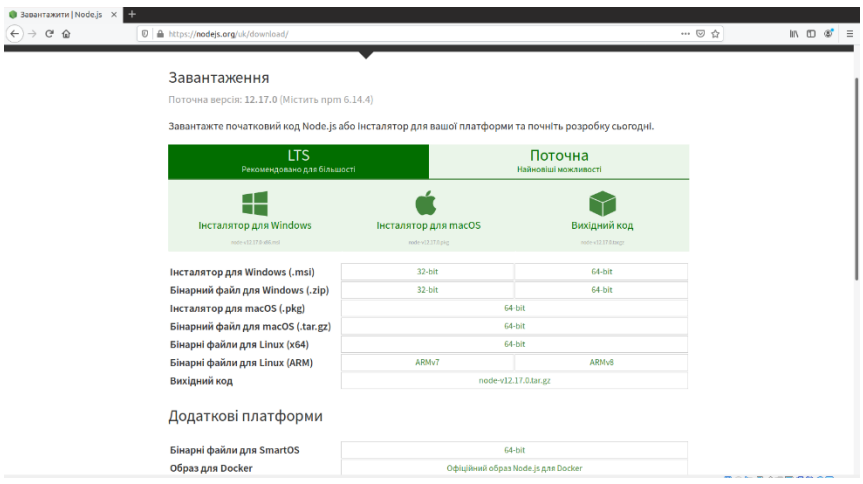


Рисунок 2.3 – Веб-сторінка для завантаження node.js

На рисунку 2.3 зображено вигляд веб-сторінки на які можна завантажити Node.js. Тут можна завантажити для всіх популярних платформ та вибрати потрібну версію для встановлення.

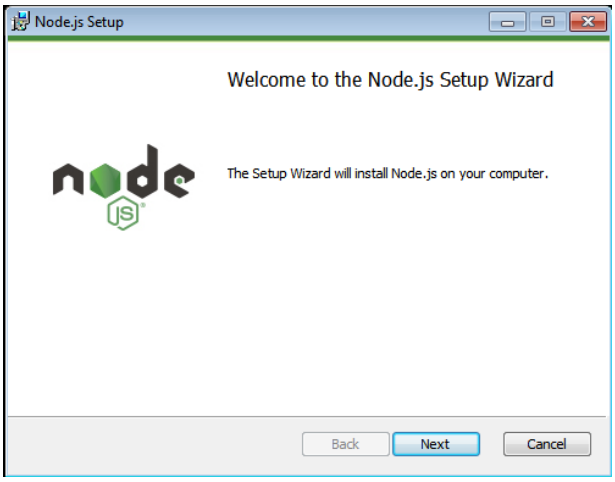


Рисунок 2.4 – Початкове вікно встановлення node.js

На початковому вікні немає корисної інформації окрім повідомлення про встановлення. Для того, щоб пройти далі потрібно натиснути кнопку «Next», після чого відкриється вікно узгодження ліцензії.

Рисунок 2.5 – Вікно узгодження ліцензії node.js

У вікні узгодження ліцензії на рисунку 2.5 зберігається сам текст ліцензії та її прив’язка до встановлюваного програмного забезпечення.

Для продовження роботи інсталятора потрібно перечитати текст ліцензії та тільки після цього вибрати чек-бокс «I accept the terms in the License Agreement», інакше встановлення програмного забезпечення не можливе, і далі натиснути кнопку «Next» для переходу у вікно вибору локації куди потрібно встановити

вибране програмне забезпечення.

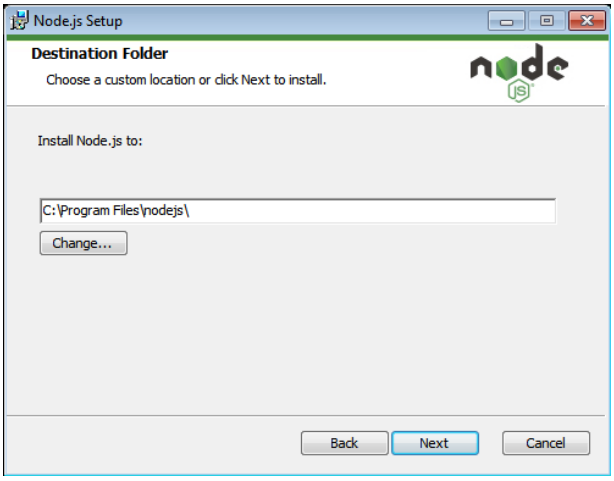


Рисунок 2.6 – Вікно вибору шляху для встановлення node.js

У вікні вибору директорії для встановлення можна вибрати будь-яку папку в операційній системі, для чого потрібно натиснути кнопку «Change» та вибрати папку з відповідного вікна операційної системи або ж вписати вручну у відповідному полі на рисунку 2.6 та натиснути кнопку «Next».

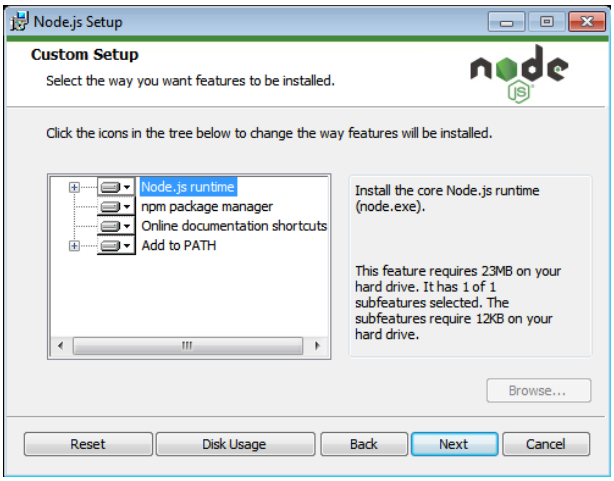


Рисунок 2.7 – Вікно вибору додаткових опцій для встановлення

Після цього відкриється вікно вибору компонентів для встановлення. Обов’язковими компонентами для встановлення є:

– Node.js та всі бібліотеки для запуску.
– npm – пакетний менеджер для встановлення пакетів.

Не обов’язковими є додаткова інформації про Node.js, наприклад література або документації. Після вибору потрібних компонентів для продовження потрібно натиснути кнопку «Next».

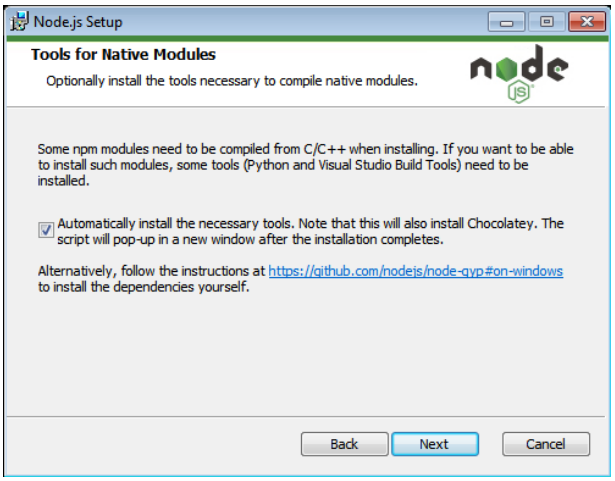


Рисунок 2.8 – Вікно підтвердження встановлення додаткових інструментів

Після цього відкриється вікно для вибору прапора для встановлення обов’язкових додаткових інструментів які не йдуть разом з Node.js, але потрібні для її роботи. Для продовження потрібно натиснути кнопку «Next».

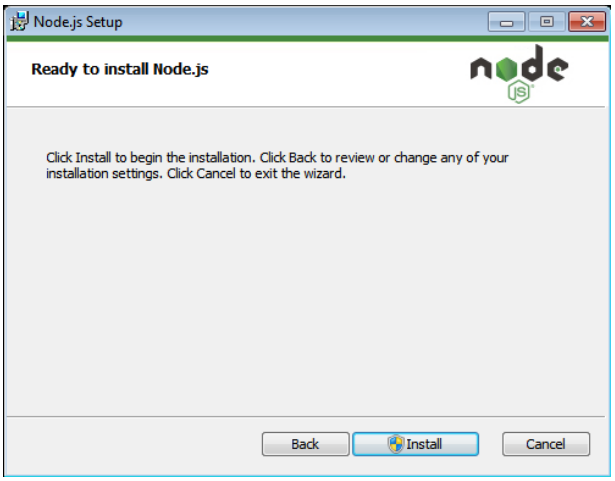


Рисунок 2.9 – Вікно підтвердження встановлення з правами адміністратора

У вікні підтвердження (рис. 2.9) потрібно натиснути кнопку «Install», що вимагає адміністраторського доступу для роботи з операційною системою, цим ж підтвердивши встановлення. Після чого відкриється вікно з прогресом

встановлення програмного забезпечення як зображено на рисунку 2.10.

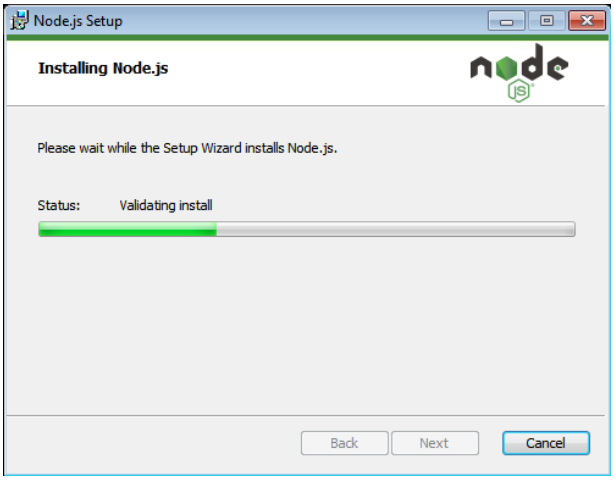


Рисунок 2.10 – Вікно прогресу встановлення node.js

У вікні прогресу є лише опція скасування встановлення, кнопка «Cancel». В разі скасування всі тимчасові файли та файли які вже були встановлені автоматично видаляться.

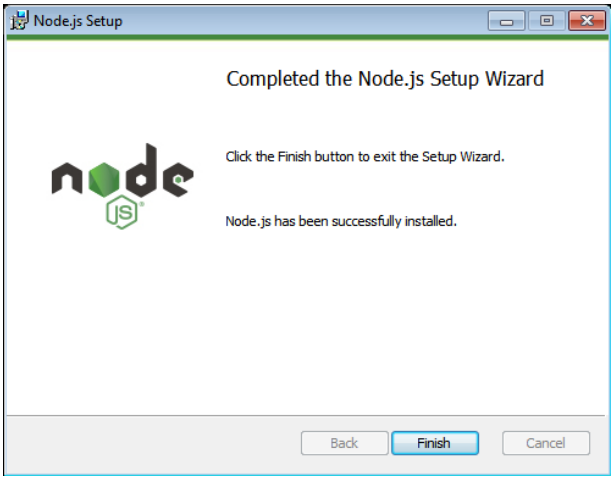


Рисунок 2.11 – Вікно завершення встановлення node.js

Після завершення встановлення відкриється відповідне вікно (рис. 2.11) на якому можна буде завершити роботу з інсталятором натиснувши кнопку «Install».

Для підтвердження того факту, що потрібно програмне забезпечення було встановлено, потрібно відкрити термінал та відповідною командою перевірити

встановлену версію. Команда для дізнавання версії Node.js та її відповідний результат зображено на рисунку 2.12. Немає різниці з якої директорії запускати виконання цієї команди, адже Node.js встановлюється глобально.

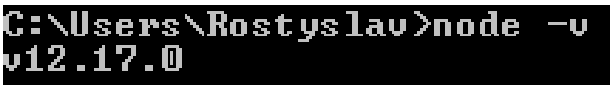


Рисунок 2.12 – Команда для отримання версії node.js

На рисунку 2.13 зображено процес встановлення додаткових залежностей до відповідного програмного коду. Для цього потрібно зайти в головну директорію та ввести відповідну команду.

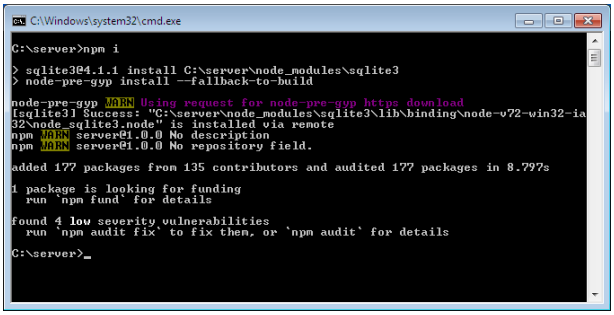


Рисунок 2.13 – Процес встановлення залежностей для сервера

Після встановлення залежностей можна запустити веб-сервер за допомогою команди яка показана на рисунку 2.14.

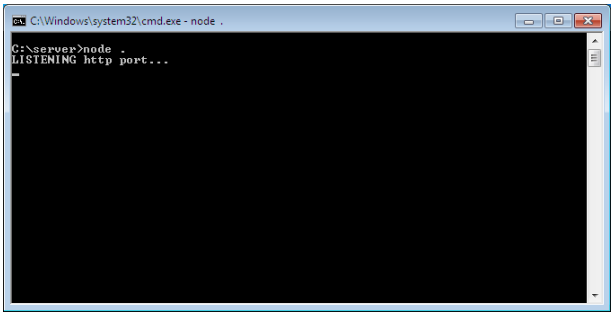


Рисунок 2.14 – Запуск сервера на порті 80

Інструкція з встановлення додатку на смартфон:

1. Зайти в Play Market [21].

2. Ввести в пошуку «Музейний помічник».
3. Встановити програму від розробника Pidburachynskyi Rostyslav.
4. Запустити програму.

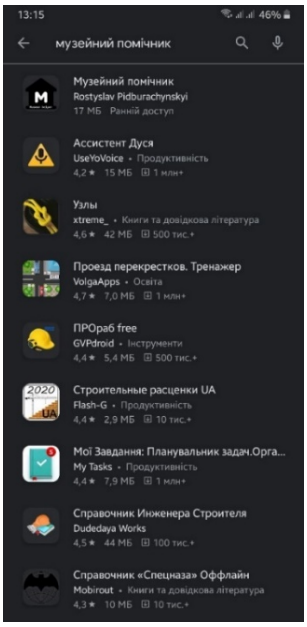


Рисунок 2.15 – Пошук додатку в play market

На рисунку 2.15 зображено вікно мобільного пристрою при пошуку додатку в Play Market. Після знаходження додатку потрібно вибрати його та перейти на його сторінку.

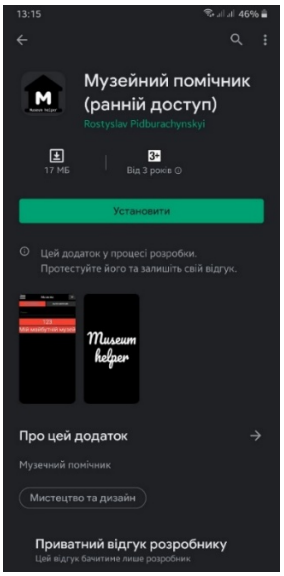


Рисунок 2.16 – Сторінка додатку в play market

На рисунку 2.16 зображено сторінку додатку. Для встановлення потрібно натиснути кнопку «Установити» після чого почнеться підготовка завантаження.

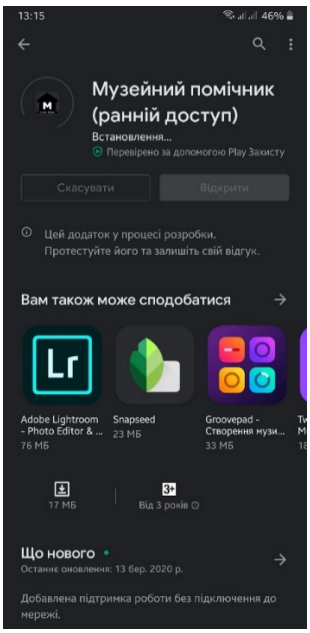


Рисунок 2.17 – Вікно початку завантаження

На рисунку 2.17 зображено вікно початку завантаження. На цьому етапі перевіряється чи підтримується даний пристрій вибраним додатком та чи є достатньо місця на пристрої для встановлення цього додатку.

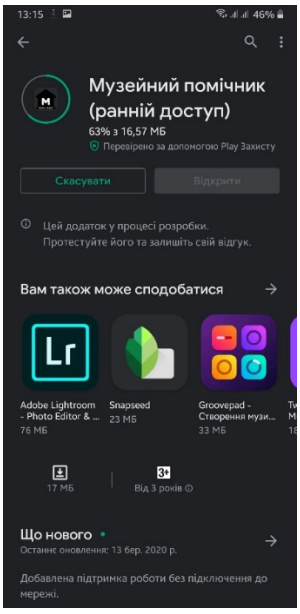


Рисунок 2.18 – Вікно завантаження

На рисунку 2.18 зображено вікно завантаження. На цьому вікно присутня інформації для прогрес завантаження та скільки ще потрібно встановити. Для скасування встановлення потрібно натиснути кнопку «Скасувати».

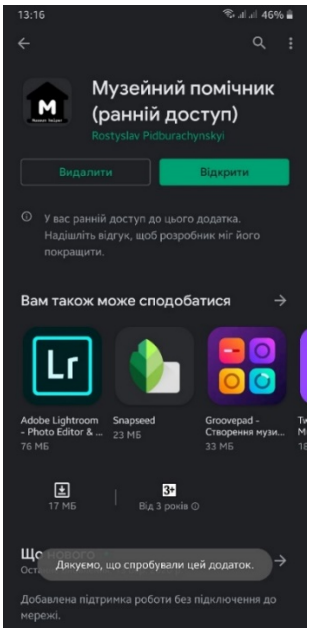


Рисунок 2.19 – Вікно завершення завантаження

Після завершення встановлення додатку вікно набуде стану як зображено на рисунку 2.19. Для відкриття додатку потрібно натиснути кнопку «Відкрити».

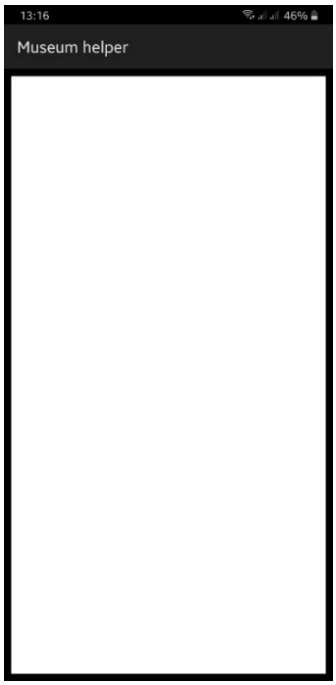


Рисунок 2.20 – Відкриття додатку

На рисунку 2.20 зображено вікно яке настає при першому відкриванні додатку. Сам екран білий адже сам під час цього ініціалізуються всі компоненти система і тільки тоді записуються в кеш для пришвидшення подальшої взаємодії користувача з додатком.

Рисунок 2.21 – Вікна дозволу на використання камери

На рисунку 2.21 зображено те як операційна система Android запитує дозвіл на використання камери, адже в додатку пристуній функіонал який її вимагає. Для дозволу потрібно натиснути кнопку «Дозволити», для заборони потрібно настиснути кнопку «Відхилити».

В раза відхилення дозволу на камеру додаток зможе продовжити роботу, лише без функціоналу який вимагає її.

2.7 Інструкція з використання тестових наборів

Для використання тестових наборів Cypress потрібно повністю завантажити вихідний програмний код веб-частини та відповідною командою

test запустити їх.

Інструкція з запуску тестів:

1. Встановити cypress використовуючи команду *npm i cypress -g*.
2. Запустити програму використовуючи команду *cypress open*.
3. Вибрати папку з проєктом.
4. Зачекати відкриття браузера та переглянути результати тестів.

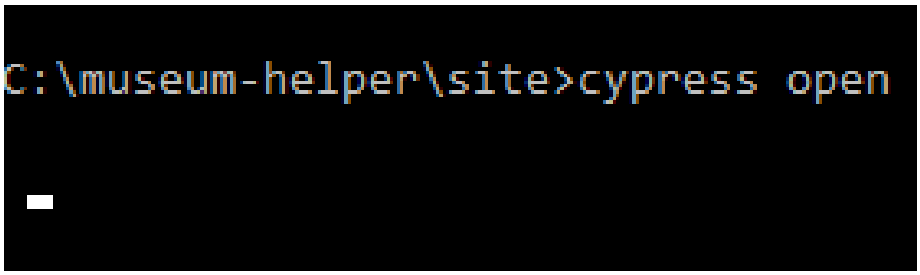


Рисунок 2.22 – Команда *cypress open* в командрій стрічці

На рисунку 2.22 зображено приклад запуску відповідної команди для відкриття тестового середовища.

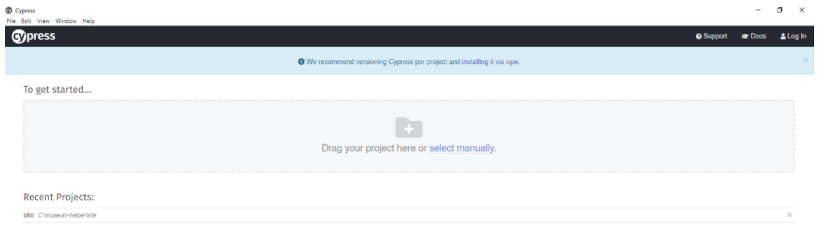


Рисунок 2.23 – Початкове вікно програми

На рисунку 2.23 зображено початкове вікно програми в якому можна вибрати папку в якій знаходяться тести для їх подальшого запуску.

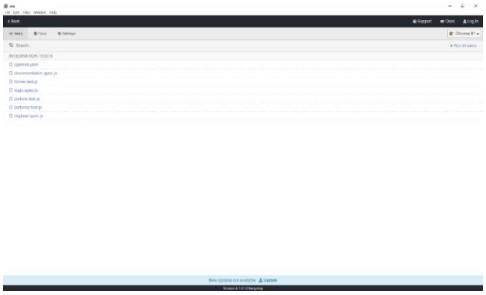


Рисунок 2.24 – Вікно з написаними тестами

На рисунку 2.24 зображено вікно з тестами які програма знайшла в папці. Для подальшого їх запуску потрібно натиснути кнопку «Run» у верхньому правому куті вікна програми.

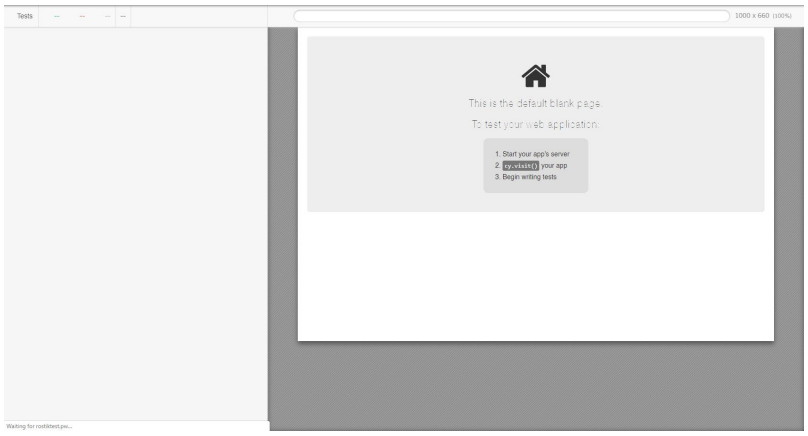


Рисунок 2.25 – Вікно браузера без тестів

На рисунку 2.25 зображено початкове вікно перед запуском тестів. Тут проходять всі потрібні для роботи ініціалізації компонентів, після чого запускається тестування.

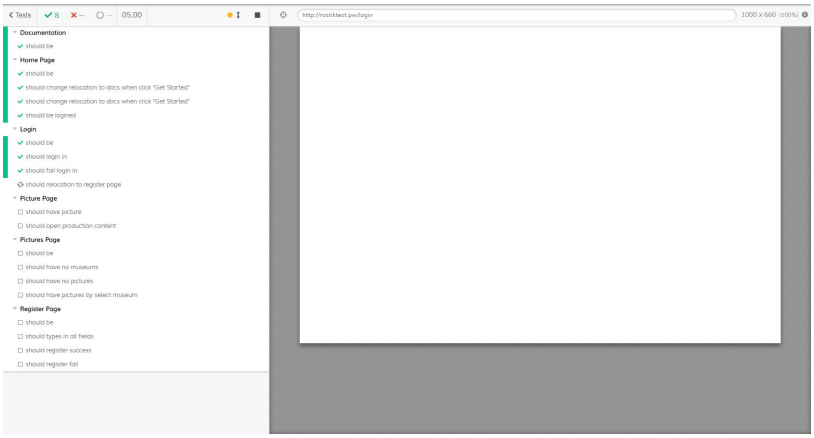


Рисунок 2.26 – Вікно браузера під час тестування

На рисунку 2.26 зображено вікно під час того як запускаються та проходять тести. В разі успішного проходження тестів зліва появляється зелений маркер, а у разі не успішного червоний. Під час виконання тестів в правій частині екрану видно їхній вигляд, а у лівій частині їхній прогрес та те на якому вони етапі зараз. Якщо один з етапів виявиться не успішний всі наступні виконуватися не будуть.

Для того щоб перезапустити або зупинити тест потрібно у верхній лівій частині вікна вибрати відповідну опцію за застосувати її. Також при зміні тесту він перезапуститься самостійно.

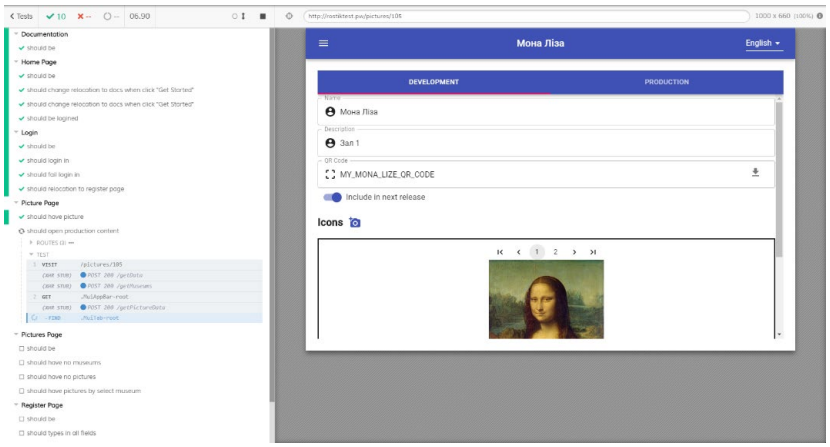


Рисунок 2.27 – Вікно браузера під час тестування конкретної сторінки

На рисунку 2.27 зображено приклад того як буде виглядати вікно браузера на відповідному етапі та з відповідним станом який був на цей момент заданий кодом.

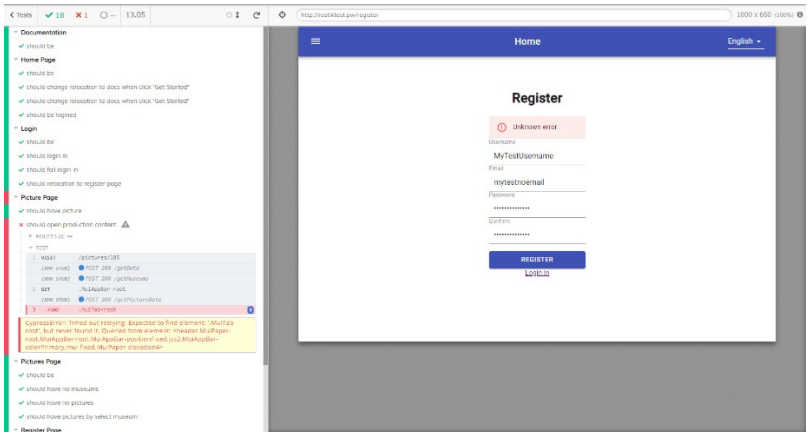


Рисунок 2.28 – Вікно браузера з непройденим тестом

На рисунку 2.28 зображено вікно браузера у якому проводиться приклад того, як буде виглядати коли тест не пройде.

2.8 Інструкція з експлуатації програмного комплексу

Для початку роботи з веб-частиною потрібно отримати html-розмітку, css-стилі та javascript код з сервера, для цього потрібно просто ввійти за певною адресою в браузері, це дозволить користувачам з графічною оболонкою у інтерфейсі пройти реєстрацію, по можливості підтвердити e-mail адресу, після чого авторизуватися у сервісі, щоб отримати можливість керувати своїми музеям та картинами у них.

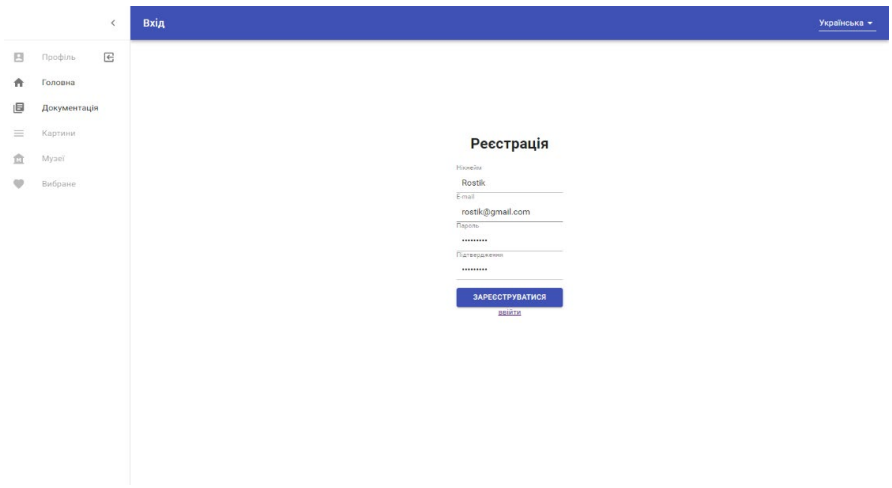


Рисунок 2.29 – Вікно реєстрації

Після реєстрації попадаємо на головну сторінку з якої можна перейти на сторінку документації та сторінку роботи з картинами або в разі не ввійшовши в профіль пройти у вікно заходу в систему.

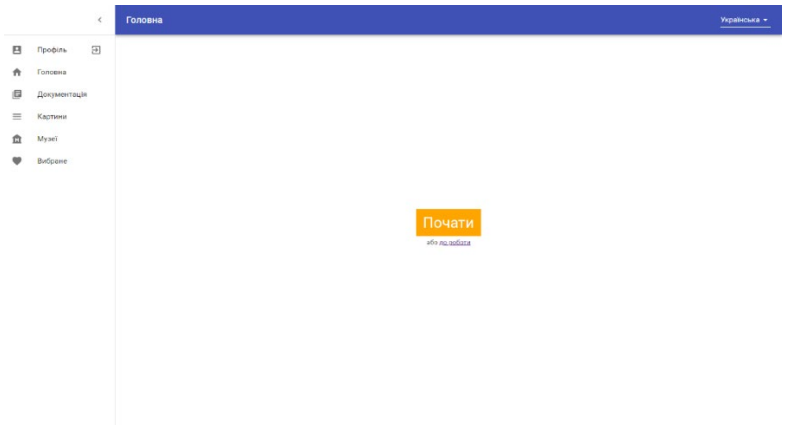


Рисунок 2.30 – Головна сторінка

Для початку роботи потрібно створити музей, це можна зробити на сторінці музеїв.

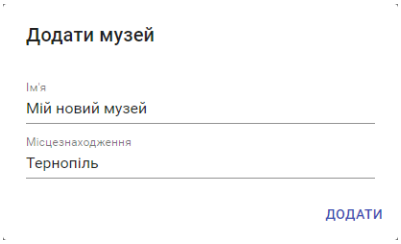


Рисунок 2.31 – Вікно додавання нового музею

На рисунку 2.31 зображено вікно з двома тестовими полями які потрібно заповнити для того щоб додати музей. Саме ця інформація про музей буде відображатися в додатку.

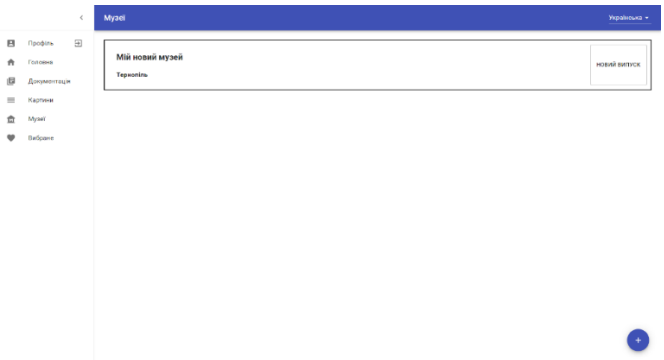


Рисунок 2.32 – Сторінка з музеями

Після створення музею потрібно перейти на сторінку з картинами та добавити картину у відповідний музей.

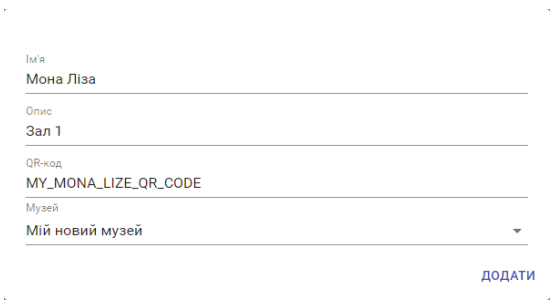


Рисунок 2.33 – Вікно додавання картини

На рисунку 2.33 зображені поля які потрібні для заповнення при додаванні картини до музею. Кожне з цих полів можна буде потім проредагувати після створення картини.

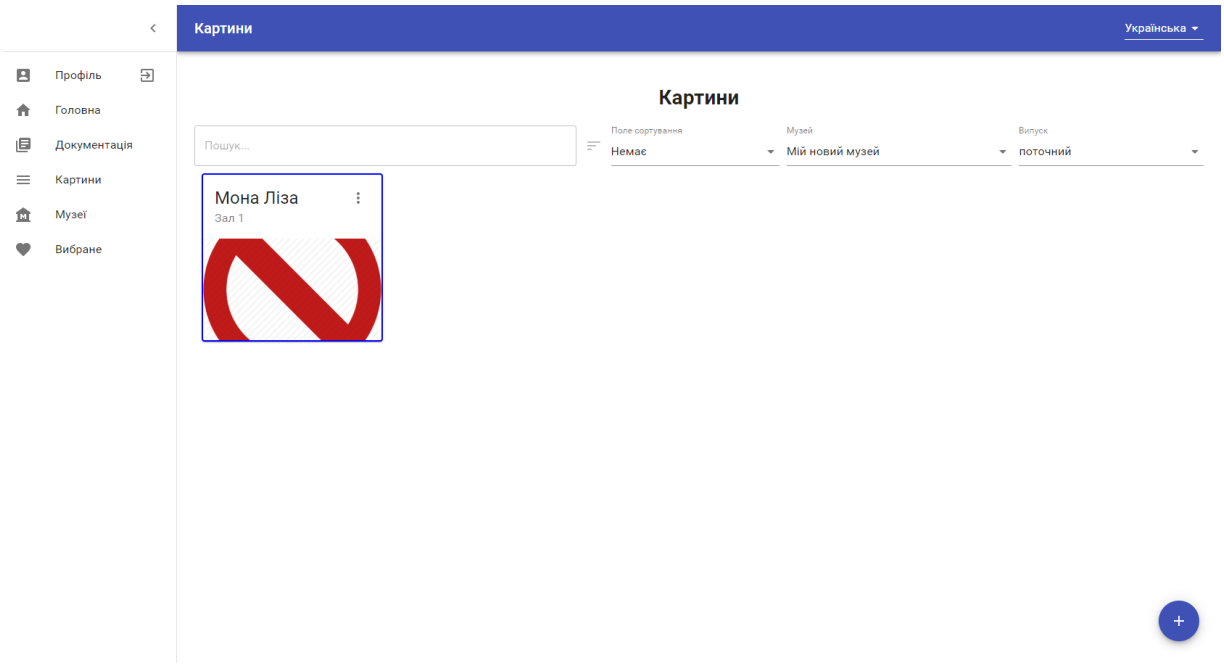


Рисунок 2.34 – Сторінка з картинами

Після додавання картини можна продовжити додавати їх, або перейти на сторінку конкретної картини та добавити переклад заголовку та опису на інші мови.

Рисунок 2.35 – Вікно додавання мови

На рисунку 2.35 зображено випадаючий список з додаванням опису та інформації про картину на вибраний мові.

Якщо ж не додати мову то картина буде в додатку відображатися без інформації, що може відлякати користувача. Те як заповнюється ця інформація зображено на рисунку 2.36, де зображено всі доступні дії які може зробити адміністратор на цій сторінці.

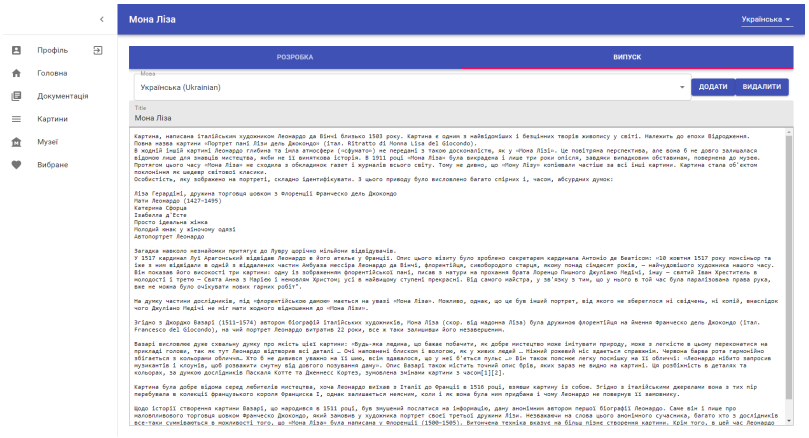


Рисунок 2.36 – Опис картини українською мовою

На рисунку 2.36 зображено вигляд вікна у якому заповнено опис картини українською мовою. Опис може додаватися на різних мовах і всі вони будуть зображені в додатку у користувача. При додаванні нової мови потрібно випустити нову версію музея, і тільки після завантаження музею в додатку користувач побачить зміни. При заповнені інформації відсутня кнопка «зберегти», адже при завершенні редагування інформація буде збережена автоматична.

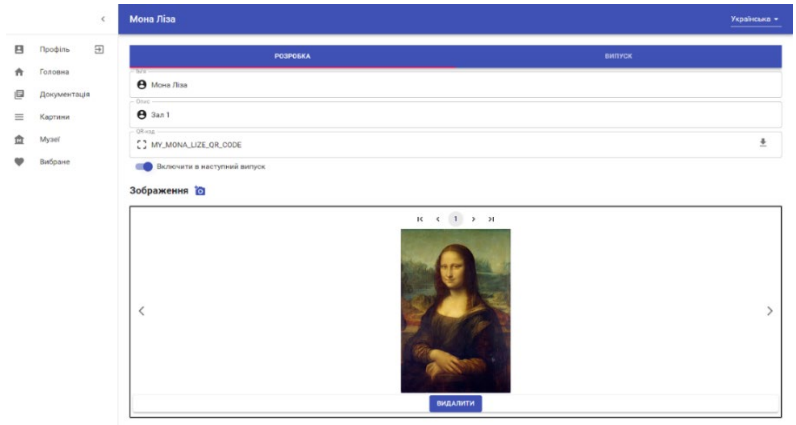


Рисунок 2.37 – Вікно картини з інформацією для адміністратора

На рисунку 2.37 зображено вікно картини з інформацією лише для адміністратора музею, адже ця інформація ніколи не буде показуватися користувачу.

Також картини можуть бути розміщені на сторінці з картинами які позначені як «вибрані».

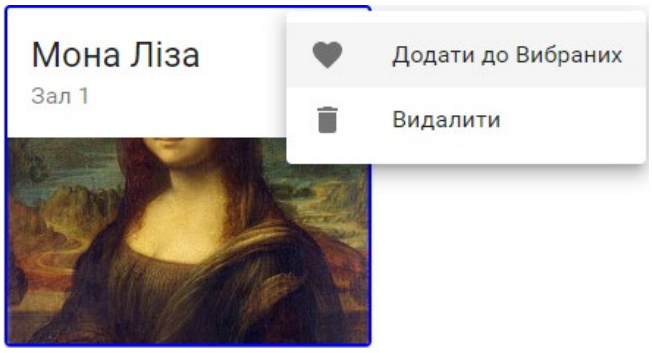


Рисунок 2.38 – Меню картини

Для цього потрібно зайти в меню картини та добавити командою «Додати до Вибраних», для видалення потрібно провести аналогічну дію вибравши опцію «Видалити».

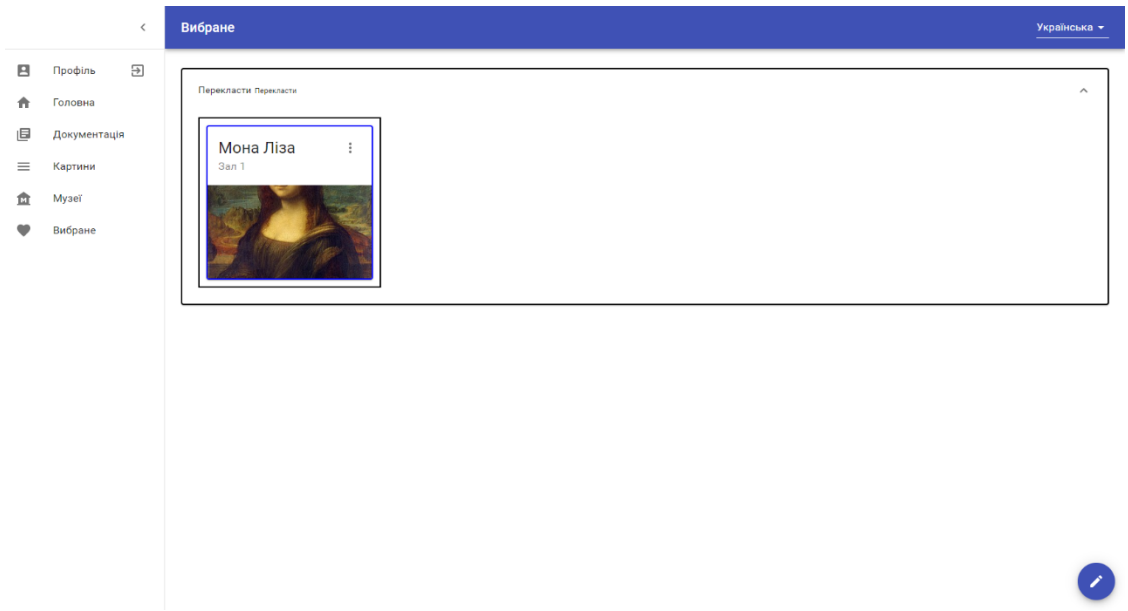


Рисунок 2.39 – Сторінка картин які по значені як «вибрані»

На цій сторінці можна створювати, видаляти та редагувати групи в яких

будуть розміщені картини. Для цього потрібно перейти в режим редагування вибравши кнопку редагування в правому нижньому куті сторінки.

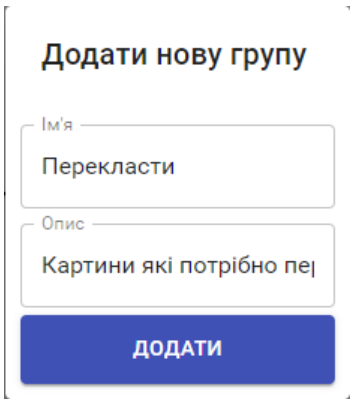


Рисунок 2.40 – Вікно додавання нової групи

Для того щоб добавити картину у групу потрібно просто перенести її туди використовуючи технології “Drag And Drop”.

Інструкція з експлуатація додатку представлена нижче. При запуску додатку вперше він перед показом зображення декілька секунд буде зберігати дані для себе. Як тільки він запуститься, на екрані покажеться список всіх музеїв та буде представлена можливість пошуку музеїв по імені.

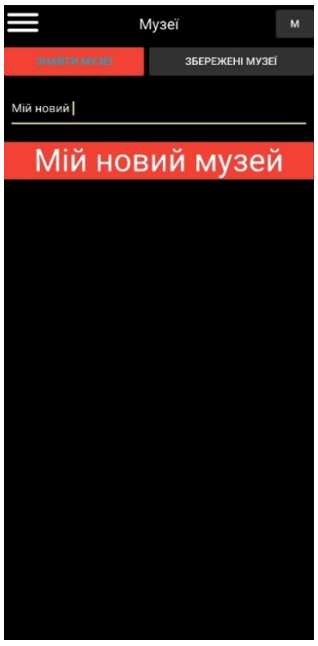


Рисунок 2.41 – Вікно пошуку музеїв

Після пошуку музею його потрібно вибрати, далі появиться вікно яке запропонує зберегти музей для подальшої роботи з ним.

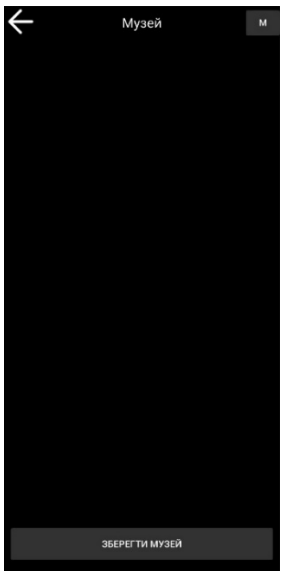


Рисунок 2.42 – Вікно музею

Після натискання кнопки «Зберегти музей» музей буде збережено зі всіма його картинами без зображень, далі буде запропоновано зберегти зображення до цих картин.

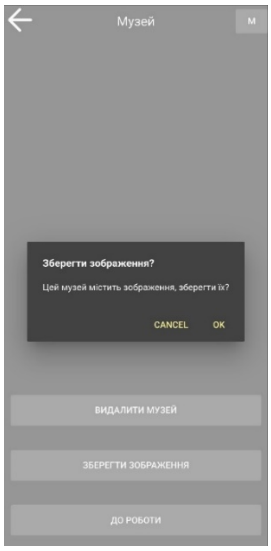


Рисунок 2.43 – Діалог завантаження зображень

Після завантаження або не завантаження зображень потрібно перейти до роботи з музеєм за допомогою клавіші «До роботи», після чого на екрані

появиться головне вікно для роботи з картинами. У ньому буде вибір між скануванням QR коду або пошуком картини у відповідному списку. В разі вибору сканування відкриється вікно «Сканування» (див. рис. 2.44) у якому потрібно буде навести камеру телефону на відповідне зображення QR коду та почекати поки додаток розшифрує його та перейде до картини (див. рис. 2.47) з відповідним кодом, в разі якщо код не буде відповідати ніякій з встановлених картин то нічого не буде відбуватися і помилок в додатку не буде траплятись.

Другий варіант дійти до зображення це вибрати клавішу з лупою де буде доступний перелік усіх картин з якого можна буде вибрати картину та перейти до її перегляду (див. рис. 2.45).

Рисунок 2.44 – Вікно «Сканування»

На рисунку 2.44 зображено вікно яке маж вигляд того що буде бачити користувач при скануванні QR-коду.

Після успішного сканування користувач попаде на відповідну картину

прив’язану до коду, а інакшому випадку користувач побачить відповідну помилку.

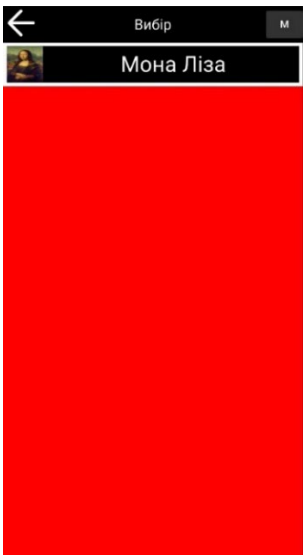


Рисунок 2.45 – Вікно пошуку картин

На рисунку 2.45 зображено вікно пошуку картин, адже не завжди у картини може бути QR-код, тому є така можливість вибору картини. Після вибору картини користувач попаде на вікно з картиною (рис. 2.46).

Рисунок 2.46 – Вікно картини

На рисунку 2.46 показано як виглядає вікно з картиною та

продемонстровано можливість вибору мови опису картини в двох мовах: українська та англійська.

2.9 Висновок до розділу

В ході роботи над розділом:

– описано процес встановлення мобільного додатку на смартфон з операційною системою Android з Play Market;

– описано процес встановлення програмного забезпечення для запуску серверної частини Node.js;

– продемонстровано процес встановлення додаткових залежностей для серверного коду;

– показано послідовний процес запуску тестів.

Також в розділі описуються послідовні кроки які має виконати користувач при взаємодії з додатком та вигляд усіх присутніх в додатках вікон.

РОЗДІЛ 3. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ ТА ОСНОВИ ОХОРОНИ ПРАЦІ

3.1 Долікарська допомога при кровотечах

В середньому в людині 5 літрів крові, вона виконує важливу роль в організмі перенесення кисню, вуглекислого газу та різних елементів. Втрата крові є небезпечним чинником, що може призвести до смерті. Тому важливо знати та розуміти як саме надати першу невідкладну допомогу. Розрізняють 4 ступні втрати крові [24]:

– перший ступінь – об'єм циркулюючої крові (ОЦК) від 1 до 15 відсотків. Свідомість людини збережена, видимі прояви крововиливу мінімальні. Такий крововилив супроводжується поблідінням покровів шкіри та пульсом 85-100 ударів в хв;

– пругий ступінь – ОЦК вік 15 до 25 відсотків. Свідомість у людини присутня але спостерігається порушення чіткого мислення. Супроводжується блідістю шкіряних покровів та слизових оболонок, пульс при цьому 100-120 ударів в хв;

– третій ступінь – ОЦК від 25 до 40 відсотків. Свідомість в людини відсутня. Спостерігається патологічна олігурія, шкірні покрови чітко видимі білим кольором, також температура тіла є низькою (можна відчути на додик). Пульс при цьому 120-140 ударів в хв;

– четвертий ступінь – ОЦК більше 40 відсотків. Свідомість відсутня, спостерігаються згасаючі рефлекси, шкірні покрови синіють, тривала анурія. В такому стані людина впадає в кому, за нею йде клінічна смерть та при відсутності апарату життєзабезпечення смерть. Пульс більше 140 ударів за хвилину, але периферичну пульсацію визначити неможливо.

При перших трьох стадіях надавши допомогу людину спасти можливо, при четвертій шансів майже немає за відсутності спеціалістів та необхідного обладнання. Кровотечу поділяють на 3 види [25]: капілярну, венозну та артеріальну.

При капілярній кровотечі із рани кров виходить із всієї рани та в більшості випадків затягується сама і не потребує обов'язкового втручання в її процес загоєння, але тим не менш слід пам'ятати, що в кров може потрапити бруд, віруси, тощо. Однак, якщо людина хворіє гемофілією або є широке пошкодження слід вжити заходів до її зупинки та дезінфекції області поранення. При венозній кровотечі кров виходить рівномірним струменем та має вишневий колір [25].

Для зупинки венозної і капілярної кровотечі слід надягнути гумові рукавиці, щоб захистити себе від можливого зараження хворобами, що передаються через кров. Далі слід виявити ділянку ураження потерпілого після чого накласти тканину або стерильну пов'язку і рівномірно притиснути, після 10 хв кров повинна зупинитися. Якщо кров не зупинилася слід натиснути сильніше або перевірити чи в тому місці є витік крові. Якщо в рані є стороннє тіло чи кістка, тоді слід використовувати кільцеву подушечку. Робиться вона з бинта або тканини, спочатку слід зробити петлю обмотавши навколо пальців та обмотати навколо петлі, щоб утворилося кільце. При пораненні руки чи ноги слід цю кінцівку підняти, щоб зменшити прилив крові до неї.

По закінченню кровотечі слід щільно перебинтувати рану але не надто сильно, щоб не порушити кровообіг.

Артеріальна кровотеча є най небезпечніша, оскільки кров втрачається дуже швидко, б'є фонтаном та має червоний колір. Таку кровотечу слід зупинити негайно по її виявленню та викликати швидку. Для цього слід накласти джуг або щільну пов'язку [25].

Джгут накладається вище місця ураження, попередньо підклавши тканину, щоб не пошкодити шкіру при накладанні. Час який можна тримати джгут від 1,5 до 2 годин. Це залежить від погодних умов та самого потерпілого. Літом це 2 години, зимою або якщо пацієнт знаходиться в шоковому стані 1 година. На тканині або окремій записці вказують час накладання джгута. При відсутності невідкладної допомоги слід послабляти джгут на кілька хвили та знову затягувати вказуючи час цих дії. При правильному накладеному джгуті відбувається зупинка кровотечі, блідніть кінцівки, відсутність пульсу та спорожнення вен.

Якщо поранення знаходиться на тій частині тіла де джгут накласти складно чи не можна слід застосувати спосіб пальцевого притискання. Він заключається в притисканні вени за пару сантиметрів від ураження, щоб зупинити кровотечу. Кожну артерію слід правильно притиснути [26]:

– A.temporalis притискають до скроневої кістки на 2 см вище і вперед від отвору зовнішнього слухового проходу.

– A.facialis притискають до нижньої щелепи на 2 см вперед від кута нижньої щелепи.

– A.carotis притискають до «сонного» горбика поперечного відростка шостого шийного хребця посередині внутрішнього краю кивального м'яза.

– A.subclavia притискають до першого ребра за ключицею в середній третині останньої.

– A.axillaris притискають до головки плечової кістки посередині переднього краю росту волосся в пахвовій западині.

– A.brachialis притискають до внутрішньої поверхні плечової кістки в середній третині медіального краю двоголового м'яза.

– A.femoralis притискають до горизонтальної гілки лобкової кістки посередині пахвинної зв'язки.

– A.poplitea притискають до задньої поверхні великогомілкової кістки у підколінній ямці при зігнутому коліні.

– Aorta abdominalis притискають кулаком до поперекового відділу хребта зліва від пупка.

Фінальну зупинку кровотечі проводять такими методами:

– механічними (перев'язка або прошивання судини);

– фізичними (гіпотермія, заморожування азотом, серветками змоченими гарячим фізіологічним розчином);

– хімічними (перекис водню, норадреналін, адреналін, інгібітори);

– біологічними (тампонування пасмом сальника, препаратами плазми, інгібіторами фібринолізу).

В незалежності від виду приміщення, чи то офісне чи то вільний зал з великою кількостю людей завжди у всіх повинен бути доступ до аптечки у якій

буду знаходитися всі необхідні засоби для зупинки кровотечі які детально описані вище.

3.2 Вплив шуму на організм людини та розробка заходів щодо його зниженню до допустимих величин для обладнання

Шум як несприятливий чинник виробничого середовища наявний на більшості промислових підприємств, на транспорті, у сільському господарстві. Джерелами шуму можуть бути системи вентиляції та кондиціонування повітря, аерогазодинамічні установки, двигуни, верстати, молоти, дробарки [24].

Інтенсивний виробничий шум може стати причиною таких професійних захворювань, як туговухість або глухота. Крім того, у працівників, які щодня перебувають під його впливом:

– знижується продуктивність праці;

– ослаблюється увага та уповільнюється реакція, спостерігається запаморочення, дратівливість, знижується працездатність, гострота зору;

– зростає кров’яний тиск, змінюється ритм дихання та серцевої діяльності, порушується працездатність клітин кори головного мозку тощо.

Звичним для людини є шумовий фон з рівнем звукового тиску в частотах 15-35 дБ. При збільшенні рівня звукового тиску до 40-70 дБ спостерігається деяке зниження продуктивності праці та погіршення самопочуття. Рівень звукового тиску в межах 75-120 дБ спричиняє враження органів слуху і серцево-судинної системи. Постійний шум з рівнем звукового тиску понад 120 дБ може призвести до акустичної травми.

Дія його на організм людини пов’язана головним чином із застосуванням нового, високопродуктивного обладнання, механізацією та трудових процесів, у тому числі переходом на великі швидкості при експлуатації різних верстатів і агрегатів. Джерелами шуму в цьому випадку можуть бути двигуни, насоси, компресори, турбіни, пневматичні та електричні інструменти, молоти, дробарки, верстати, центрифуги, бункери та інші установки, що мають рухомі деталі.

Це один із найбільш поширених несприятливих фізичних факторів

навколишнього середовища. При запуску реактивних двигунів літаків рівень шуму коливається від 120 до 140 дБ; при клепанні й рубанні листової сталі — від 118 до 130 дБ; роботі деревообробних верстатів — від 100 до 120 дБ, ткацьких верстатів — до 105 дБ [23].

Коли мова йде про вплив шуму, то зазвичай основну увагу приділяють стануоргануслуху,такякслуховийаналізаторупершучергу сприймає звукові коливання і подразнення його є адекватним дії шуму на організм.

Зміни, що виникають в органі слуху, деякі дослідники пояснюють травмуючою дією шуму на периферичний відділ слухового аналізатора внутрішнього вуха. Основною ознакою впливу шуму є зниження слуху по типу кохлеарного невриту. Професійне зниження слуху буває зазвичай двостороннім.

Захист від шуму повинен забезпечуватися розробкою шумобезпечної техніки, застосуванням засобів і методів колективного захисту, в тому числі будівельно-акустичних, застосуванням засобів індивідуального захисту.

Колективні засоби захисту поділяються на засоби, що знижують шум у джерелі його виникнення, і засоби, що знижують шум на шляху його поширення від джерела до об’єкта, що захищається [24].

Зниження шуму в джерелі здійснюється за рахунок поліпшення конструкції машини або зміни технологічного процесу. Методи і засоби колективного захисту, в залежності від способу реалізації, поділяються на будівельно-акустичні, архітектурно-планувальні та організаційно-технічні і включають в себе:

– зміну спрямованості випромінювання шуму;
– раціональне планування підприємств і виробничих
– приміщень;
– акустичну обробку приміщень;
– застосування звукоізоляції.

У низці випадків величина показника спрямованості досягає 10-15 дБ, щонеобхідно враховувати при використанні установок з направленим випромінюванням, орієнтуючи ці установки так, щоб максимум випромінюваного шуму був спрямований у протилежну сторону від робочого

місця людини [24].

Раціональне планування підприємств і виробничих приміщень дозволяє знизити рівень шуму на робочих місцях за рахунок збільшення відстані до джерел шуму.

Засоби індивідуального захисту (ЗІЗ) застосовуються в тому разі, якщо іншими способами забезпечити допустимий рівень шуму на робочому місці не вдається. Принцип дії ЗІЗ — захистити найбільш чутливий канал впливу шуму на організм людини — вухо. Застосування ЗІЗ дозволяє попередити розлад не тільки органів слуху, а й нервової системи від дії надмірного подразника.

Найбільш ефективні ЗІЗ, як правило, в області високих частот. ЗІЗ включають в себе протишумні вкладиші, навушники, спеціальні костюми [23]. Це особливо важливо у випадку якщо в музеї грає голосна музика, ведуться не значні ремонтні роботи або ж приміщення має погано звукоізоляцію, це все може нашкодити організму. Також при роботі в офісному приміщені, приміщення має бути захищене спеціальними обладнанням, щоб запобігти поганого самопочуття співробітників.

3.3 Висновок до розділу

У третьому розділі кваліфікаційної роботи бакалавра було розглянуто долікарську допомога при кровотечах та вплив шуму на організм людини та розробка заходів щодо його зниженню до допустимих величин для обладнання.

Основною метою першої допомоги у разі кровотеча є їх тимчасова зупинка на долікарському етапі, яка дозволить доправити хворого до медичного закладу для надання кваліфікованої допомоги

Для здійснення заходів з послаблення інтенсивності впливу шкідливих виробничих факторів на виробничий персонал необхідно організувати належну взаємодію служби охорони праці з виробничими структурними підрозділами підприємства, спрямовану на впровадження передових технологій у цій сфері.

ВИСНОВКИ

В першому розділі кваліфікаційної роботи освітнього рівня «Бакалавр»:

– проводиться аналіз предметної області та зовнішнє проєктування програмного комплексу;

– здійснено постановку задачу на розробку програмного комплексу для покращення досвіду відвідувачів музеїв;

– проаналізовано інструменти для розробки програмного комплексу, проведено порівняння фреймворків для розробки веб-частини комплексу та наведено аргументи на користь вибраних інструментів.

В другому розділі кваліфікаційної роботи:

– розроблено алгоритм роботи програм та їхньої взаємодії в середині програмного комплексу між собою використовуючи транспорт http;

– запропоновано опис процедур розробки, тестування та експлуатації додатку з проведеного документування під час роботи над проєктом;

– описано тестування роботи веб-частини, серверної частини програмного комплексу та роботу мобільного додатку. Для тестування деяких компонентів веб-частини використовувалися інструменти автоматизованого тестування.

У розділі «Безпека життєдіяльності та основи охорони праці» висвітлюється долікарська допомогу при кровотечах та фактору шуму як несприятливий чинник виробничого середовища наявний на більшості промислових підприємств.

В процесі роботи над кваліфікаційною роботою бакалавра було розроблено програмний комплекс «Музейний помічник», який складається з мобільного додатку, веб-частини та серверної частини для взаємодії. Цей програмний комплекс дозволяє:

– створити свій онлайн музей, у якому можна добавити картини та опис для цих картин;

– відвідувачам музею завантажити додаток, в якому можна буде вибрати

поточний музей та переглядати опис картин на смартфоні з підтримкою багатомовності, зручного інтерфейсу та можливістю сканування QR коду.

Аналізуючи пророблено роботу можна прийти до висновків, що:

– сайт комплексу є досить красивим з точки зору вигляду інтерфейсу так як використовує дизайнерські рішення від Google у вигляді Material блоків;

– мобільний додаток комплексу містить в собі зручні налаштування та інтерфейс, повний потрібний для додатку функціонал і можливість працювати без виході до мережі.

Також були прийняті наступні рішення з врахуванням деяких вимог:

– використано сайт замість додатку - для можливості працювати з музеями та картинами з будь-якої точки планети використовуючи смартфон;

– використано готову material схему для дизайну веб-сайту – для швидкості розробки продукту без затримки на розробку інтерфейсу та виявлення його багів, так як такий популярний набір контролів вже протестовано;

– використано базу даних sqlite для синхронного збереження однакових даний на сервері та у додатку для смартфону, щоб забезпечити можливість роботи зі збереженими музеями без виходу до мережі.

Даний програмний комплекс орієнтований на користувачів музеїв, які часто їх відвідують, це допоможе уникнути замовлення гіда та з більшою цікавістю розглядати експонати, а також цей програмний комплекс допоможе іншим людям, яким не подобаються походи в музей переглядати твори мистецтва не виходячи з дому.

Програмний комплекс надає можливості перегляду всієї інформації щодо картини за допомогою мобільного додатку, після сканування QR-коду картини, з детальними зображеннями, описами та назвами на різних мовах, розроблених кожним музеєм індивідуально та змогою побачити точку зору різних людей та надання можливості власникам музеїв написання власних текстів для експонатів за для збереження авторського контенту та думки автора.

ПЕРЕЛІК ДЖЕРЕЛ

1. Шлее М. Qt 5.10. Професійне програмування на C++ / Макс Шлее. – Санкт-Петербург: Питер, 2016. – 1072 с. ISBN 978-5-9775-3678-3.

2. Бенкс А. React і Redux - функціональна веб-розробка / А. Бенкс, Е. Порселло. – Санкт-Петербург: Питер, 2016. – 336 с. ISBN 978-5-4461-0668-4.

3. Стефанов С. React.js - Швидкий старт / Стоян Стефанов. – Санкт-Петербург: Питер, 2018. – 334 с.. ISBN 978-5-496-03003-8.

4. Янг А. Node.js в дії / А. Янг, Б. Мек, М. Кентелон. – Санкт-Петербург: Питер, 2018. – 432 с. ISBN 978-5-496-03212-4.

5. Браун І. Веб-розробка з використання Node і Express – Повноцінне використання стеку JavaScript / Ітан Браун. – Санкт-Петербург: БХВ, 2016. – 336 с. ISBN 978-5-4461-0590-8.

6. Керівництво з оформлення HTML / CSS коду від Google [Електронний ресурс]. – Режим доступу до ресурсу: https://habr.com/post/143452/.

7. JavaScript-бібліотека React [Електронний ресурс]. Режим доступу до ресурсу: https://uk.reactjs.org.

8. Redux [Електронний ресурс]. Режим доступу до ресурсу: https://redux.js.org.

9. Material-UI [Електронний ресурс]. Режим доступу до ресурсу: https://material-ui.com.

10. npm [Електронний ресурс]. Режим доступу до ресурсу: https://www.npmjs.com.

11. GitHub Vue.js [Електронний ресурс]. Режим доступу до ресурсу: https://github.com/vuejs/vue.

12. Node.js web application [Електронний ресурс]. Режим доступу до ресурсу: https://expressjs.com.

13. FASTVPS BILLING [Електронний ресурс]. Режим доступу до ресурсу: https://bill2fast.com/.

14. SQLite Docs [Електронний ресурс]. Режим доступу до ресурсу: https://www.sqlite.org/docs.html.

15. Qt Documentation [Електронний ресурс]. Режим доступу до ресурсу: https://doc.qt.io.

16. Android Developers [Електронний ресурс]. Режим доступу до ресурсу: https://developers.android.com.

17. Why Cypress? [Електронний ресурс]. Режим доступу до ресурсу: https://docs.cypress.io.

18. Qt Bug Tracker [Електронний ресурс]. Режим доступу до ресурсу: https://bugreports.qt.io/browse/QTBUG-103712.

19. MDN Web Docs [Електронний ресурс]. Режим доступу до ресурсу: https://developer.mozilla.org.

20. Google Play Console [Електронний ресурс]. Режим доступу до ресурсу: https://play.google.com/apps/publish.

21. Google Play [Електронний ресурс]. Режим доступу до ресурсу: https://play.google.com.

22. Вікіпедія [Електронний ресурс]. Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Каркас_вебзасосунків.

23. Засоби захисту від шуму та вібрації [Електронний ресурс]. Режим доступу до ресурсу: https://www.sop.com.ua/news/3625-dopomogu-za-e-lkarnyanim-mojna-otrimati-na-pdstav-vityagu.

24. MCFR – Цифрове видавництво [Електронний ресурс]. Режим доступу до ресурсу: https://sop.com.ua.

25. Що робити у разі ушкодження кінцівок та при кровотечі? - Портал Києва [Електронний ресурс]. Режим доступу до ресурсу: https://kyivcity.gov.ua/likarni_ta_medytsyna/persha_dolikarska_dopomoha/scho_robiti_u_razi_ushkodzhennya_kintsivok_ta_pri_krovotechi/.

26. Барвінська А. Кровотеча і крововтрата. / Методи тимчасової і кінцевої зупинки кровотечі. / Барвінська А. – Львів, 2019. – 19 с.

ДОДАТКИ

Додаток А

Лістинг файлу index.js

```
const express = require("express");
const app = express();
const subdomain = require('./modules/subdomain');
const path = require("path");
app.set('subdomain offset', 0);
app.use("/static/pictureIcons",
express.static(path.join(__dirname, "/../uploads")));
app.use(subdomain('api', require("./subdomains/api")));
app.use(require("./subdomains/_"));
app.listen(80, () => console.log("LISTENING http port..."));
```

Додаток Б

Лістинг файлу subdomains/_/index.js

```
const express = require("express");
const router = express.Router();
router.use(express.static(__dirname + "/"));
router.use("*", (req, res) => {
     res.sendFile(__dirname + "/index.html");
});
module.exports = router;
```

Додаток В

Лістинг файлу subdomains/routes/index.js

```
const { sendError } = require("./statics");
const express = require("express");
const router = express.Router();
const cookieParser = require("cookie-parser");
const bodyParser = require("body-parser");
const { getIdBySesid } = require("./routes/user/db");
router.use(cookieParser());
router.use(bodyParser.json());
router.use(/.*/, (req, res, next) => {
    if(req.headers.origin)
    res.setHeader('Access-Control-Allow-Origin',
req.headers.origin);
    res.setHeader('Access-Control-Allow-Methods', 'GET, POST,
OPTIONS, PUT, PATCH, DELETE');
    res.setHeader('Access-Control-Allow-Headers', 'X-Requested-
With,content-type');
    res.setHeader('Access-Control-Allow-Credentials', true);
    next();
});
router.use(require("./routes/register"));
router.use(require("./routes/login"));
router.use(require("./routes/app"));
router.use((req, res, next) => {
    const { sesid } = req.cookies;
    if(!sesid) {
        res.send({
            success: false,
            error: "UNLOGINED_USER"
        });
        return;
    }
    getIdBySesid(sesid)
        .then(id => {
            req._payload = { user: { id } };
```

```
  next();
        }).catch(sendError(res));
});
router.use(require("./routes/user"));
router.use(require("./routes/picture"));
router.use(require("./routes/museums"));
router.use(require("./routes/pictures"));
router.use(require("./routes/favorites"));
module.exports = router;
```

Додаток Г

Лістинг файлу subdomains/api/routes/app/index.js

```
const router = require('express').Router();
const bodyParser = require("body-parser");
const { getMuseumsByNamePattern, getMuseumById } =
require('../museums/db');
const { getReleasedPicturesByMuseumId } =
require('../pictures/db');
const { getReleasedPicturesInfoByMuseumId,
getReleasedPicturesIconsByMuseumId } = require('../picture/db');
router.use(bodyParser.json());
router.post('/app/getMuseums', (req, res) => {
    const { pattern = ""} = req.body;
    getMuseumsByNamePattern(pattern)
        .then(museums => {
            res.send({
                success: true,
                museums
            });
        });
});
router.post('/app/getMuseum', (req, res) => {
    const { id } = req.body;
    require('../museums/db').getMuseumById(id)
        .then(museum => {
            res.send({
                success: true,
                museum
            });
        });
});
router.post('/app/getPictures', (req, res) => {
    const { museumId } = req.body;
    let pictures, picturesInfo, picturesIcons;
    getReleasedPicturesByMuseumId(museumId)
        .then(p => pictures = p)
```

```
        .then(() => getReleasedPicturesInfoByMuseumId(museumId))
        .then(p => picturesInfo = p)
        .then(() => getReleasedPicturesIconsByMuseumId(museumId))
        .then(p => picturesIcons = p)
        .then(() => {
            res.send({
                success: true,
                pictures,
                picturesInfo,
                picturesIcons
            });
        });
});
module.exports = router;
```

Додаток Д

Лістинг файлу subdomains/routes/favorites/index.js

```
const express = require("express");
const router = express.Router();
const { sendAllData, sendError } = require("../../statics");
const { addFavoritePicture, deleteFavoritePictureByPictureId,
getFavorites, changeFavorites, addFavotireGroup,
deleteFavotireGroup } = require("./db");
router.post("/getFavorites", (req, res) => {
    const { user: { id }} = req._payload;
    getFavorites(id)
        .then(groups => ({ groups }))
        .then(sendAllData(res))
        .catch(sendError(res));
});
router.post("/saveFavorites", (req, res) => {
    const { groups } = req.body;
    changeFavorites(groups)
        .then(sendAllData(res))
        .catch(sendError(res));
});
router.post("/addPictureToFavorites", (req, res) => {
    const { id } = req.body;
    addFavoritePicture(id)
        .then(sendAllData(res))
        .catch(sendError(res));
});
router.post("/deletePictureFromFavorites", (req, res) => {
    const { id } = req.body;
    deleteFavoritePictureByPictureId(id)
        .then(sendAllData(res))
        .catch(sendError(res));
});
router.post("/addFavoriteGroup", (req, res) => {
    const { name, description } = req.body;
    const { user: { id }} = req._payload;
```

```
    addFavotireGroup(id, name, description)
        .then(group => ({ group }))
        .then(sendAllData(res))
        .catch(sendError(res));
});
router.post("/deleteFavoriteGroup", (req, res) => {
    const { id } = req.body;
    deleteFavotireGroup(id)
        .then(sendAllData(res))
        .catch(sendError(res));
});
module.exports = router;
```

Додаток Е

Лістинг файлу subdomains/routes/login/index.js

```
const express = require("express");
const router = express.Router();
const utils = require("../../utils");
const { sendAllData, sendError } = require("../../statics");
const { loginIn, verifyEmail, getVerifyLink } =
require("../user/db");
const mailc = require("../../mailc");
router.post("/loginIn", (req, res) => {
    const { email, password } = req.body;
    loginIn(email, password)
    .then(({ id, username, email }) => {
        res.cookie("sesid", utils.createSesid(id));
        res.send({
            success: true,
            username,
            email
        });
    }).catch(sendError(res));
});
router.post("/verifyEmail", (req, res) => {
    const { link } = req.body;
    verifyEmail(link)
        .then(sendAllData(res))
        .catch(sendError(res));
});
router.post("/verifyEmailAgain", (req, res) => {
    const { email } = req.body;
    getVerifyLink(email)
        .then(link => mailc.sendEmailVerify(email, link))
        .then(sendAllData(res))
        .catch(sendError(res))
});
module.exports = router;
```

Додаток Є

Лістинг файлу subdomains/routes/museums/index.js

```
const express = require("express");
const router = express.Router();
const { sendAllData, sendError } = require("../../statics");
const { addMuseum, removeMuseum, getMuseumById,
getMuseumsByUserId, changeMuseumData, newReleaseByMuseumId } =
require("./db");
router.post("/getMuseum", (req, res) => {
    const { museumId } = req.body;
    getMuseumById(museumId)
        .then(museum => ({ museum }))
        .then(sendAllData(res))
        .catch(sendError(res));
});
router.post("/getMuseums", (req, res) => {
    const { user: { id }} = req._payload;
    getMuseumsByUserId(id)
        .then(museums => ({ museums }))
        .then(sendAllData(res))
        .catch(sendError(res));
});
router.post("/changeMuseumData", (req, res) => {
    const { museumId, changes } = req.body;
    changeMuseumData(museumId, changes)
        .then(sendAllData(res))
        .catch(sendError(res));
});
router.post("/addMuseum", (req, res) => {
    const { name, location } = req.body;
    const { user: { id }} = req._payload;
    addMuseum(id, name, location)
        .then(addedMuseum => ({ addedMuseum }))
        .then(sendAllData(res))
        .catch(sendError(res));
});
```

```
router.post("/removeMuseum", (req, res) => {
    const { museumId } = req.body;
    removeMuseum(museumId)
        .then(sendAllData(res))
        .catch(sendError(res));
});
router.post("/newReleaseMuseum", (req, res) => {
    const { museumId } = req.body;
    newReleaseByMuseumId(museumId)
        .then(sendAllData(res))
        .catch(sendError(res));
});
module.exports = router;
```

Додаток Ж

Лістинг файлу subdomains/routes/picture/index.js

```
const express = require("express");
const router = express.Router();
const { sendAllData, sendError } = require("../../statics");
const multer = require("multer");
const { addPictureInfo, removePictureInfoById, getPictureInfo,
changePictureInfo, addIconToPicture, deleteIconFromPictureById } =
require("./db");
const { processFileToFilename } = require("../../utils");
const { changePicture } = require("../pictures/db");
router.post("/getPictureData", (req, res) => {
    const { id } = req.body;
    getPictureInfo(id)
        .then(sendAllData(res))
        .catch(sendError(res));
});
router.post("/savePictureInfo", (req, res) => {
    const { id, changes } = req.body;
    changePictureInfo(id, changes)
        .then(changes => ({ changes }))
        .then(sendAllData(res))
        .catch(sendError(res));
});
router.post("/addPictureInfo", (req, res) => {
    const { pictureId, title = "", description = "", language } =
req.body;
    addPictureInfo(pictureId, title, description, language)
        .then(addedPictureInfo => ({ addedPictureInfo }))
        .then(sendAllData(res))
        .catch(sendError(res));
});
router.post("/removePictureInfo", (req, res) => {
    const { id } = req.body;
    removePictureInfoById(id)
        .then(sendAllData(res))
```

```
        .catch(sendError(res));
});
router.post("/savePictureData", (req, res) => {
    const { id, changes } = req.body;
    changePicture(id, changes)
        .then(changes => ({ changes }))
        .then(sendAllData(res))
        .catch(sendError(res));
});
router.post("/addIconToPicture",
multer({dest:"uploads"}).single("icon"), (req, res) => {
    const { id } = req.body;
    processFileToFilename(req.file)
    .then((filename) => addIconToPicture(id, filename))
    .then(icon => ({ icon }))
    .then(sendAllData(res))
    .catch(sendError(res));
});
router.post("/deleteIconFromPicture",
multer({dest:"uploads"}).single("icon"), (req, res) => {
    const { id } = req.body;
    deleteIconFromPictureById(id)
        .then(sendAllData(res))
        .catch(sendError(res));
});
module.exports = router;
```

Додаток З

Лістинг файлу subdomains/routes/pictures/index.js

```
const express = require("express");
const router = express.Router();
const { sendAllData, sendError } = require("../../statics");
const { getPictures, deletePicture, addPicture } =
require("./db");
router.post("/getPicturesData", (req, res) => {
    const { searchParams: { searchText,
        sortedField,
        sortedType,
        museumId,
        updateId,
        pageNumber = 1,
        limit = 5 }} = req.body;
        getPictures(museumId, searchText, sortedField, sortedType,
museumId, updateId, limit, pageNumber)
        .then(sendAllData(res))
        .catch(sendError(res));
});
router.post("/deletePicture", (req, res) => {
    const { id, searchParams: { searchText,
                                sortedField,
                                sortedType,
                                museumId,
                                pageNumber,
                                limit }} = req.body;
    const userId = req._payload.user.id;
    deletePicture(id)
        .then(() => getPictures(userId, searchText, sortedField,
sortedType, museumId, limit, pageNumber))
        .then(sendAllData(res))
        .catch(sendError(res));
});
router.post("/addPicture", (req, res) => {
    const { museumId, name, description, qrcode } = req.body;
```

```
    addPicture(museumId, name, description, qrcode)
        .then(picture => ({ picture }))
        .then(sendAllData(res))
        .catch(sendError(res));
});
module.exports = router;
```

Додаток И

Лістинг файлу subdomains/routes/register/index.js

```
const express = require("express");
const router = express.Router();
const utils = require("../../utils");
const { registerUser, setVerifyLink } = require("./registerDb");
const mailc = require("../../mailc");
const { sendAllData, sendError } = require("../../statics");
const { customError } = require("../../statics");
router.post("/registerIn", (req, res) => {
    const { username, email, password, passwordConfirm } =
req.body;
    const link = utils.makeLink();
    checkPasswords(password, passwordConfirm)
        .then(() => checkEmail(email))
        .then(() => registerUser(username, email,
utils.hashPassword(password)))
        .then(setVerifyLink(email, link))
        .then(() => mailc.sendEmailVerify(email, link))
        .then(sendAllData(res, {link}))
        .catch(sendError(res));
});
const checkEmail = (email) => new Promise((resolve, reject) => {
    if(/\S+@\S+\.\S+/.test(email)) return resolve();
    return reject(customError("EMAIL_IS_NOT_VALID"));
});
const checkPasswords = (pass = "", confirm) => new
Promise((resolve, reject) => {
    if(pass.length < 8) return
reject(customError("PASSWORD_LENGTH_LESS", { field: 'password'}));
    if(pass !== confirm) return
reject(customError("PASSWORDS_IS_NOT_IDENTICAL", { field:
"confirm"}))
    resolve({});
});
module.exports = router;
```

Додаток І

Лістинг файлу subdomains/routes/user/index.js

```
const express = require("express");
const router = express.Router();
const { sendAllData, sendError } = require("../../statics");
const { hashPassword } = require("../../utils");
const { getUser, checkPassword, changeUserData, deleteSesid } =
require("./db");

router.post("/changeUserData", (req, res) => {
    const { password, changes } = req.body;
    const { user: { id }} = req._payload;
        checkPassword(id, hashPassword(password))
        .then(() => changeUserData(id, changes))
        .then(sendAllData(res))
        .catch(sendError(res))
});
router.post("/unlogin", (req, res) => {
    const { sesid } = req.cookies;
    deleteSesid(sesid)
        .then(() => {
            res.clearCookie("sesid");
            sendAllData(res)()
        }).catch(sendError(res));
});
router.post("/getData", (req, res) => {
    const { user: { id }} = req._payload;
    getUser(id)
        .then(sendAllData(res))
        .catch(sendError(res));
});
module.exports = router;
```

Додаток Ї

Лістинг файлу subdomains/routes/mailc.js

```
const { GMAIL_LOGIN, GMAIL_PASSWORD, HOST } = require("./secret");
const nodemailer = require("nodemailer");

exports.sendEmailVerify = (email, link) => {
     sendMail(
          [email],
          'Vefiry email',
          `Hello, it your link -
http://${HOST}/verifyEmail/${link}`
     );
}
const sendMail = (to, title, text) => {
     var mailOptions = {
          from: 'Museum Helper',
          to,
          subject: title,
          text: text
     };
     transporter.sendMail(mailOptions, (error, info) => {
          if (error) {
               console.log(error);
          } else {
               console.log(info);
          }
     });
}
process.env.NODE_TLS_REJECT_UNAUTHORIZED = '0';
var transporter = nodemailer.createTransport({
     service: 'gmail',
     auth: {
          user: GMAIL_LOGIN,
          pass: GMAIL_PASSWORD
     }
});
```

Додаток Й

Лістинг файлу subdomains/routes/statics.js

```
const sqlite = require("sqlite3").verbose();
const path = require("path");
exports.db = new sqlite.Database(path.resolve(__dirname,
"../../../databases/nice.db"));
const SERVER_ERROR = "SERVER_ERROR";
exports.serverError = () => ({ error: SERVER_ERROR });
exports.customError = (type, other = {}) => ({ type, ...other });
exports.sendAllData = (res, other) => data =>
    res.send({ success: true, ...other, ...data });
exports.sendError = res => (e) => {
    if(!e) e.text = SCRIPT_ERROR;
    res.send({ success: false, error: e });
};
```

Додаток К

Лістинг файлу subdomains/routes/utils.js

```
const db = require("./statics").db;
const fs = require("fs");
const path = require("path");
const sha256 = require("js-sha256").sha256;
exports.parseJStoSQLQ = (obj, prefix = '') => {
    let keys = Object.keys(obj);
    let resultKeys = "";
    let resultValues = [];
    for(let i = 0; i < keys.length; i++) {
        let key = keys[i];
        resultKeys += ` ${prefix ? `${prefix}.` :
''}${correctKey(key)}=?`;
        resultValues.push(obj[key]);

        if(i !== keys.length - 1) resultKeys += ' AND '
    }
    return { resultKeys, resultValues };
}
exports.processFileToFilename = (file) => {
    return new Promise((resolve, reject) => {
        const oldPath = path.join(__dirname, "../../uploads/",
file.filename);
        const newPath = path.join(__dirname, "/../../../icons/",
file.filename);
        fs.rename(oldPath, newPath, () => {
            resolve(file.filename);
        });
    });
}
exports.hashPassword = (pass) => {
    let result = pass;
    for(let i = 0; i < 32; i++) {
        result = sha256(result);
    }
```

```
    return result;
}
exports.makeLink = () => makeString(512);
exports.createSesid = (id) => {
     const sesid = makeString(256);
     db.run(`INSERT INTO sesids (id, sesid)
            VALUES (?, ?)`, [id, sesid], (run, err) => {
                if(err) console.log(err);
            });
     return sesid;
}
const makeString = (length) => {
    var result           = '';
    var characters       =
'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789';
    var charactersLength = characters.length;
    for ( var i = 0; i < length; i++ ) {
       result += characters.charAt(Math.floor(Math.random() *
charactersLength));
    }
    return result;
 }
 const correctKey = (key) => {
    for(let i = 0; i < key.length; i++) {
        const isUpp = key[i] === key[i].toUpperCase();
        if(isUpp) {
            const begin = key.slice(0, i);
            const end = key.slice(i + 1);

            key = begin + `_${key[i++].toLowerCase()}` + end;
        }
    }
    return key;
}
```

Додаток Л

Лістинг файлу qrcodeanalyzer.cpp

```
#include "qrcodeanalyzer.h"
#include "picturesmodel.h"
#include <QtMultimedia>
#include <QtDebug>
#include "Objects/dbc.h"
QRCodeAnalyzer* QRCodeAnalyzer::m_instance = nullptr;
QRCodeAnalyzer::QRCodeAnalyzer(QObject *parent)
     : QObject(parent), m_decoding(false)
{
     if(m_instance)
          throw std::runtime_error("Create second qrcodeAnalyzer");
     m_instance = this;
     m_probe = new QVideoProbe(this);
     m_decoder = new QRCodeAnalyzerDecoder;
     connect(m_decoder, &QRCodeAnalyzerDecoder::decodingStarted,
               this, &QRCodeAnalyzer::onDecodingStarted,
Qt::QueuedConnection);
     connect(m_decoder, &QRCodeAnalyzerDecoder::decodingFinished,
               this, &QRCodeAnalyzer::onDecodingFinished,
Qt::QueuedConnection);
     connect(m_probe, &QVideoProbe::videoFrameProbed, this,
&QRCodeAnalyzer::processFrame);
}
QObject *QRCodeAnalyzer::source() const
{
     return m_source;
}
bool QRCodeAnalyzer::setSource(QObject *source)
{
     m_source = source;
     bool b = m_probe->setSource(qvariant_cast<QMediaObject
*>(source->property("mediaObject")));
     return b;
}
```

```
void QRCodeAnalyzer::onDecodingStarted()
{
     qDebug() << "Start decoding";
}
void QRCodeAnalyzer::onDecodingFinished(const QString &result)
{
     qDebug() << "Finish decoding, result:" << result;
     if(m_decodedS) return;
     m_decodedS = false;
     if(result.isEmpty())
     {
          emit failedDecoding(NotFoundCodeInImage);
          m_decoding = true;
     }else
     {
          bool valid = DBC::instance()->checkPictureQrcode(result);
          if(!valid)
          {
               emit failedDecoding(NotFoundCodeInBase);
               m_decoding = true;
          }else
          {
               m_decoding = false;
               PictureObject::instance()-
>setCurrentPictureQrcode(result);
               emit successDecoding();
          }
     }
}
extern QImage qt_imageFromVideoFrame(const QVideoFrame &f);
void QRCodeAnalyzer::processFrame(const QVideoFrame &frame)
{
     if(!m_decoding)
          return;
     m_decoding = false;
     QImage imgbuf = qt_imageFromVideoFrame(frame);
```

```
    emit decode(imgbuf);
}
bool QRCodeAnalyzer::decoding() const
{
    return m_decoding;
}
void QRCodeAnalyzer::setDecoding(bool decoding)
{
    m_decodedS = decoding ? false : true;
    m_decoding = decoding;
}
void QRCodeAnalyzer::startDecoding()
{
    setDecoding(true);
}
void QRCodeAnalyzer::stopDecoding()
{
    setDecoding(false);
}
```

Додаток М

Лістинг файлу Objects/networkmanager.cpp

```
#include "networkmanager.h"
#include "logic.h"
#include "models/museumsmodel.h"
#include "picturesmodel.h"
#include "settings.h"
QNetworkAccessManager *NetworkManager::m_manager = new
QNetworkAccessManager;
QNetworkReply *NetworkManager::getMuseums(const QString &pattern)
{
     QJsonObject json;
     json["pattern"] = pattern;
     return postSend("/getMuseums", json);
}
QNetworkReply *NetworkManager::getMuseum(const int &id)
{
     QJsonObject json;
     json["id"] = id;
     return postSend("/getMuseum", json);
}
QNetworkReply *NetworkManager::getPictures(const int &id)
{
     QJsonObject json;
     json["museumId"] = id;
     return postSend("/getPictures", json);
}
QNetworkReply *NetworkManager::getIcon(const QString &iconName)
{
     QUrl url(ICONS_HOST + iconName);
     QNetworkRequest req(url);
     return m_manager->get(req);
}

QNetworkReply *NetworkManager::postSend(const QString &path, const
QJsonObject &json)
```

```
{
     QUrl url(API_HOST + "/app" + path);
     QNetworkRequest request(url);
     request.setHeader(QNetworkRequest::KnownHeaders::ContentTypeHe
ader, "application/json");
     auto reply = m_manager->post(request,
QJsonDocument(json).toJson());
     QObject::connect(reply, static_cast<void
(QNetworkReply::*)(QNetworkReply::NetworkError)>(&QNetworkReply::e
rror),
                           [&](QNetworkReply::NetworkError err) {
          qDebug() << err;
     });
     return reply;
}
```

Додаток Н

Лістинг файлу Objects/pictureobject.cpp

```
#include "pictureobject.h"
#include "dbc.h"
PictureObject* PictureObject::m_instance = nullptr;
PictureObject::PictureObject(QObject *parent)
     : QObject(parent), m_languageIndex(0)
{
     if(m_instance)
          throw std::runtime_error("Current picture already
created");
     m_instance = this;
}
QStringList PictureObject::languagesModel() const
{
     QStringList list;
     for(auto info : m_pictureInfo) {
          if(info.language() == "english")
          {
               list.append("English");
          } else if(info.language() == "russian")
          {
               list.append("Русский");
          } else if(info.language() == "ukrainian")
          {
               list.append("Українська");
          }
     }
     return list;
}
void PictureObject::allChanged()
{
     emit iconsChanged();
     emit titleChanged();
     emit descriptionChanged();
     emit languagesSizeChanged();
```

```
}
void PictureObject::setCurrentPictureQrcode(const QString &qrcode)
{
     m_pictureInfo = DBC::instance()-
>getSavedPicturesInfoByPictureQrcode(qrcode);
     allChanged();
}
void PictureObject::setCurrentPictureId(const int &id)
{
     m_pictureInfo = DBC::instance()-
>getSavedPicturesInfoByPictureId(id);
     m_icons = DBC::instance()-
>getSavedPicturesIconsByPictureId(id);
     allChanged();
}
```

Додаток О

Лістинг файлу Data/picture.h

```
#include <QJsonObject>
#include <QImage>
#include <QDataStream>
class PictureInfo
{
public:
     PictureInfo(const QString &title, const QString &description,
const QString &language);
     inline QString title() const { return m_title; }
     inline QString description() const { return m_description; }
     inline QString language() const { return m_language; }
private:
     QString m_title;
     QString m_description;
     QString m_language;
};
class Picture
{
public:
     Picture(const int &id = 0,
               const QString &name = "",
               const QString &qrcode = "",
               const QImage &icon = QImage());
     inline int id() const { return m_id; }
     inline QString name() const { return m_name; }
     inline QString qrcode() const { return m_qrcode; }
     inline QImage icon() const { return m_icon; }
private:
     int m_id;
     QString m_name;
     QString m_qrcode;
     QImage m_icon;
};
```

Додаток П

Лістинг файлу Data/museum.h

```
#ifndef MUSEUM_H
#define MUSEUM_H
#include <QJsonObject>
class Museum
{
public:
     Museum() { }
     Museum(const int &id, const QString &name, const int
&updateId, const int &iconsSaved);
     Museum(const QJsonObject &json);
     inline int id() const { return m_id; }
     inline QString name () const { return m_name; }
     inline int updateId() const { return m_updateId; }
     inline bool iconsSaved() const { return m_iconsSaved; }
private:
     int m_id;
     QString m_name;
     int m_updateId;
     bool m_iconsSaved;
};
#endif // MUSEUM_H
```

Додаток Р

Лістинг файлу Data/bigmuseum.h

```
#ifndef BIGMUSEUM_H
#define BIGMUSEUM_H
#include <QString>
class BigMuseum
{
public:
     BigMuseum();
     int id() const;
     void setId(int id);
     QString name() const;
     void setName(const QString &name);
     QString location() const;
     void setLocation(const QString &location);
     int updateId() const;
     void setUpdateId(int updateId);
     bool hasIcons() const;
     void setHasIcons(bool hasIcons);
private:
     int m_id;
     QString m_name;
     QString m_location;
     int m_updateId;
     bool m_hasIcons;
};
#endif // BIGMUSEUM_H
```