МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП " ОДЕСЬКИЙ ПРОФЕСІЙНИЙ ТЕХНІЧНИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 121 «Інженерія програмного забезпечення»

Освітня програма:»розробка програмного забезпечення"

Група: 4РП-05

Дипломний проект

здобувача освіти денної форми навчання
РП.05.23.000.ДП

ТОКАРЧУК
ГЛІБ
СЕРГІЙОВИЧ

м. Одеса
2022 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП " ОДЕСЬКИЙ ПРОФЕСІЙНИЙ ТЕХНІЧНИЙ КОЛЕДЖ ОНТУ»

Спеціальність: **121 «Інженерія програмного забезпечення»**

Освітня програма:**"розробка програмного забезпечення"**

Група: **4РП-05**

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проекту (роботи) на тему:

Розробка веб-додатків із застосуванням сучасних технологій (PWA)

Проектний матеріал складається з пояснювальної записки на _____ сторінках і графічного (презентаційного матеріалу на _____ аркушах (слайдах).

Дипломник _____ (Токарчук Г. С.)

Керівник _____ (Сологуб К.В.)

Консультанти:

з економічної частини _____ (Копайгородська Т. Р.)

з охорони праці_____ (Чорновол Н. В.)

щодо дотримання вимог ЄСКД _____ (Петрашова в. І.)

старший консультант _____ (Скорнякова О. В.)

До захисту допущений

Голова циклової комісії _____ (Скорнякова О. В.)

Завідувач відділенням _____ (Сулима Ю. Ю.)

Захист«____»_____ 2022 р. Протокол ДКК №_____

Оцінка ДКК _____

Секретар ДКК _____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП " ОДЕСЬКИЙ ПРОФЕСІЙНИЙ ТЕХНІЧНИЙ КОЛЕДЖ ОНТУ»

Відділення_____Комісія_____

Спеціальність_____121 «Інженерія програмного забезпечення»_____

Освітня програма_____«Розробка програмного забезпечення»_____

Стверджую:

Зам. дір. з НВР_____

_____

"_____"_____2022 р.

ЗАВДАННЯ

на дипломний проект (роботу)

Здобувачеві (здобувачці) освіти _____

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) _____

_____

_____

затверджена наказом по коледжу від"____"_____202___р. №_____

2. Термін здачі закінченого проекту (роботи) _____

3. Вихідні дані до проекту (роботи) _____

_____

_____

_____

_____

_____

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

_____

_____

_____

_____

_____

5. Перелік графічного (презентаційного матеріалу (з точним зазначенням обов'язкових креслень, кількості слайдів)

_____

_____

_____

_____

_____

6. Консультанти по проекту (роботі), із зазначенням розділів проекту, що їх стосується

Розділ	Консультант	підпис, дата	
		завдання видав	завдання прийняв
Основний	Сологуб К.В.		
Економічний	Копайгородська Т.Г.		
Охорона праці	Чорновол Н.І.		
Нормоконтроль	Петрашова В.І.		
Старший консультант	Скорнякова О.В.		

7. Дата видачі завдання _____

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з / р	назва етапів дипломного проекту (роботи)	термін виконання етапів дипломного проекту (роботи)	відмітка про виконання
1	Розділ 1. Розробка PWA-додатка із застосуванням	17.05.2022	
	сучасних технологій		
2	Розділ 2. Економічний розрахунок	23.05.2022	
3	Розділ 3. Охорона праці	24.05.2022	
4	Розробка презентації до дипломної роботи	01.06.2022	
5	Чистове оформлення пояснювальної записки	04.06.2022	
6	Підготовка доповіді до захисту	08.06.2022	
7	Отримання рецензії, відповіді на зауваження	10.06.2022	
	рецензента		
8	Захист роботи	24.06.2022	

Дипломник _____
(підпис)

Керівник _____
(підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			РП 05.23.000 ДП ПЗ	*Дипломний проект*		
А4			*РП 05.23.000 ДП ПЗ*	*Пояснювальна*	*1*	
				записка		
				Презентація		

Ізм.	Арк.	№ докум.	Підпис	Дата	*РП 05.23.000*		
Розробив		Токарчук Г.С.			Розробка веб-додатків із застосуванням сучасних технологій (PWA)	Літ.	Аркуш 5 / Аркушів
Перевірив		Сологуб К.В.				НДП	
Н. Контр.		Петрашова В.І				ВСП «ОТФК ОНТУ» гр.РП-05	
Затверд.		Скорнякова О.В.					

АНОТАЦІЯ

Об'єкт дослідження: Об'єктом дослідження є PWA-додатки

Предмет дослідження: Предметом дослідження стала розробка сайту та використання його як PWA-додатки

Мета роботи: Дослідити процес створення сучасних веб-сайтів з використанням технологій складання проектів

Досягнуті результати:

- Вибрано необхідний стек для розробки програми
- Розроблено front-end та back-end веб-сайту
- Створені необхідні таблиці у базі даних.
- Створено повноцінний PWA додаток, який може використовуватись на всіх сучасних веб-браузерах

Ключові слова: Vue 3, TypeScript, Restify, REST-API.

Обсяг: 54 стор., 21 Рисунок, 10 табл., 1 додаток, 6 джерел.

ЗМІСТ

ВСТУП

На сьогодні найголовніша проблема будь-якої програми - підтримка мультиплатформенності у поєднанні з простотою розробки. Існує кілька способів зробити програму простою і кросплатформною, наприклад створити її із застосуванням веб-технологій, проте даний спосіб позбавлений підтримки нативних функцій пристрою, наприклад повідомлень, що робить веб-програму не зовсім повноцінною. На допомогу приходить Progressive Web App програма, яка працює на веб-технологіях і має можливість використовувати нативні особливості пристрою, а також може бути використана як звичайна програма, що включає ярлик на робочому столі та рамку вікна без панелі керування браузера.

Технологія PWA була створена корпорацією Microsoft у 2000 році. Широку популярність технологія PWA набула у 2015 році завдяки розширенню можливостей браузера Google Chrome та просуванню Service Worker та Web App Manifest. На даний момент саме Google просуває цю технологію та займається активною її розробкою

Мета роботи - Створити веб сайт із застосуванням сучасних технологій та фреймворків, а також розробити Rest-Api та структуру БД для нього. Створити модель авторизації та автентифікації для програми Після створення перетворити сайт на PWA програму шляхом додавання маніфесту та підключення Service Workers

Для виконання завдання необхідно виконати наступні цілі:

- Створити макет сторінки сайту за допомогою адаптивної верстки під кілька видів пристроїв
- Розробити логіку front-end, зробити додаток динамічним та підключити його до API
- Організувати сховище стану програми для збереження активної сесії та даних користувача при перезавантаженні PWA
- Створити реляційну схему бази даних з усіма сутностями, що використовуються в додатку
- Розробити Rest API для взаємодії клієнта та сервера із застосуванням авторизації та захищених шляхів
- Створити маніфест та Service Workers для PWA-програми

1.1. Розробка PWA-додатка із застосуванням сучасних технологій

При створенні дипломного проекту необхідно розробити PWA-додаток, який буде працювати на всіх відомих платформах, що підтримують сучасні браузери

1.2. Вибір технологій та бібліотек для розробки

Всі програмні засоби та технології, які використовувалися при розробці поширюються на безоплатній основі у вигляді open-source бібліотек або мають trial-період 30 днів.

Під час розробки використовувалися такі програми:

- IDE Jetbrains WebStorm (trial)
- Система контролю версій Git
- Менеджер контейнерів Docker Desktop

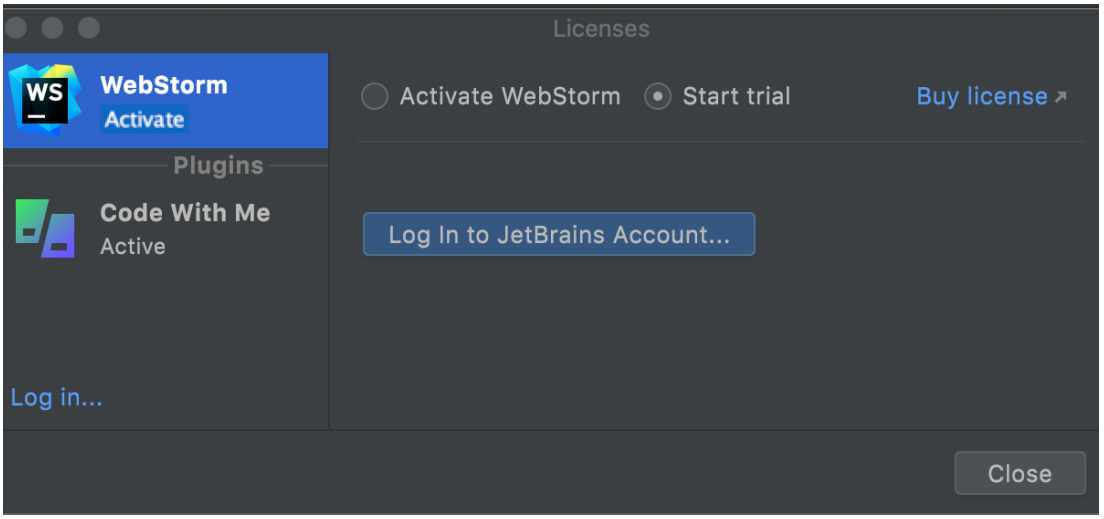


Рисунок 1.1 - Активація тріалу WebStorm

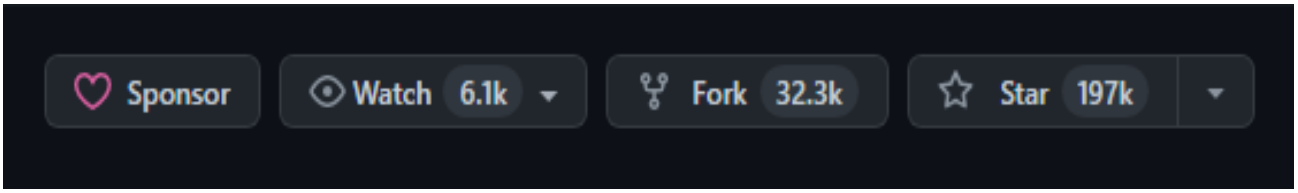
Також використовувалися бібліотеки

- Vue 3 як основний front-end фреймворк
- Tailwind 3 як css-фреймворк для полегшення верстки
- Restify як бекенд фреймворк
- Webpack як збирач проекту
- Eslint для підтримки стилю коду

Для розробки фронтенду краще бібліотека Vue через свою популярність і простоту в освоєнні.

Плюси цієї бібліотеки

- Реактивність. Vue відловлює зміну змінної у коді та перемальовує html відповідно до заданої логіки
- Компонентний підхід. У Vue весь html описується за допомогою компонентів. Програміст може створювати компоненти та інкапсулювати їхню логіку в залежності від потреби. Крім того, компоненти мають реюзабельність, тобто можуть бути використані 2 і більше разів у будь-якій частині програми
- Single-file component. Vue дозволяє розбивати свої компоненти на файли, таким чином ми обмежуємо себе декларуванням одного компонента в одному файлі, що покращує читабельність коду і відповідає концепції інкапсуляції логіки.
- Великий набір плагінів. Оскільки Vue є найпопулярнішим front-end фреймворком у світі, спільнота встигла створити великий набір плагінів, які дозволяють спростити роботу програмістів



. Рисунок 1.2 - Статистика репозиторію Vue на сайті github

1.3. Створення базового шаблону проекту

Для швидкого старту проекту ми використовуємо CLI-утиліту vue-cli. Щоб встановити її, виконайте команду npm install -g @vue/cli. У ній ми використовуємо пакетний менеджер node package manager, який є основним під час роботи з Javascript-проектами. Далі налаштувати проект, вибрати його назву та встановити основні бібліотеки.

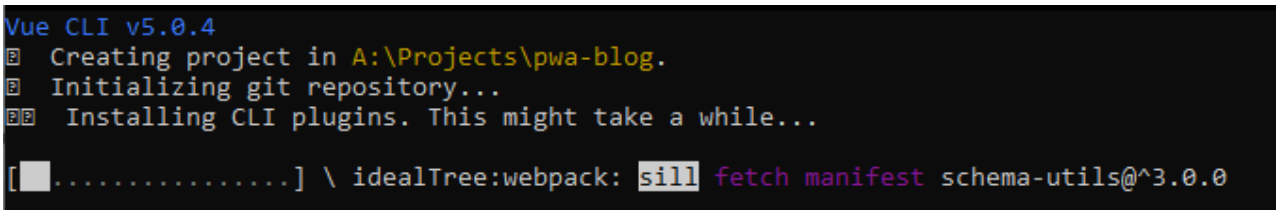


Рисунок 1.3 - Процес створення проекту та встановлення бібліотек

Після створення ми отримуємо порожній та готовий до початку розробки проект, який ми відкриваємо за допомогою IDE Webstorm

1.3.1. Архітектура проекту

Vue-cli створює готову структуру проекту, яку ми можемо використати та доповнювати у майбутньому.

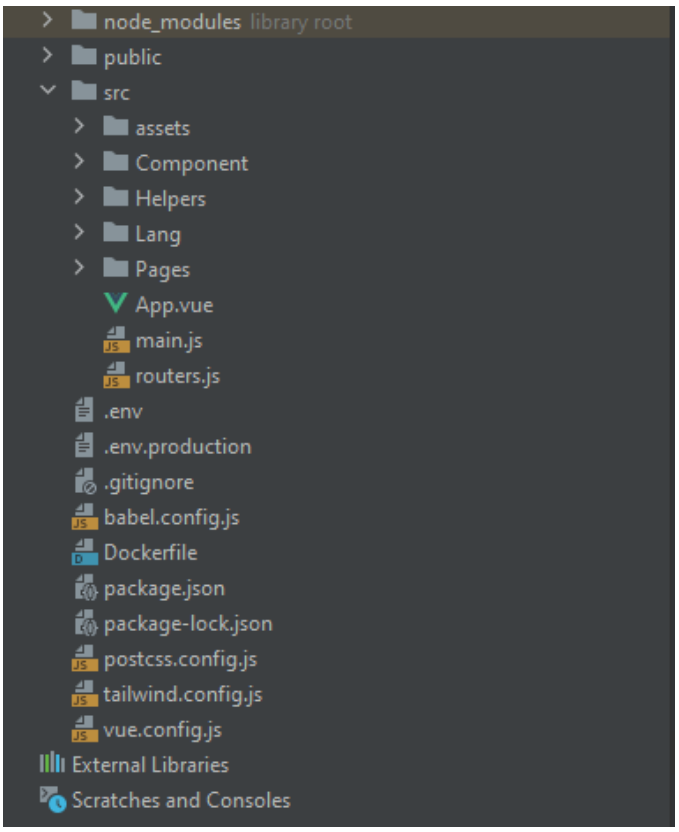


Рисунок 1.4 - Структура директорій у IDE Webstorm

Структура виконана як логічних директорій, саме:

- Public. Використовується для зберігання статичних файлів, які не змінюватимуться і можуть бути в майбутньому використані поза vue, наприклад сторонні бібліотеки та html-файли
- src. Саме файли проекту vue. У свою чергу, ця папка також ділиться на кілька інших директорій
- Assets. Тут зберігаються статичні файли, які імпортуються та компілюються при складанні, наприклад картинки та css-файли
- Component. Тут зберігаються реюзабельні компоненти, що використовуються кілька разів у проекті

- Helpers. Тут зберігаються різні js-файли з логікою, які можна використовувати у будь-якому компоненті. Служать для спрощення та інкапсуляції деякого коду компонентів
- Lang. Використовується плагіном vue-i18n для інтерналізації програми. Містить різні версії тексту програми різними мовами
- Pages. Містить компоненти vue, які служать як сторінки програми та використовуються плагіном vue-router для створення роутингу
- node_modules. Службова директорія, що містить сторонні залежності проекту

Також у проекті використовуються кілька декларативних файлів, а саме

- package.json.що вносяться в проект і build для складання проекту з подальшого його використання в продакшені.
- vue.config.js. Для конфігурації vue. Містить набір опцій для веб-сервера, webpack та eslint.

1.3.2. Панель керування та інструментів у Webstorm

Панель керування у Webstorm виконана у вигляді випадаючого списку та містить необхідні інструменти для роботи з проектом. Наприклад, ми можемо виконати пошук необхідного фрагмента коду за допомогою Navigate->File і вписати назву файлу. У комбінації з парадигмою один компонент - один файл це дає можливість швидкого і точного пошуку передбачуваного файла. Крім навігації, панель управління містить такі можливості

- File. У цьому вікні ми можемо виконати будь-які дії з файлом, наприклад зберегти його під іншим ім'ям, або відкрити/створити новий файл. Крім

цього ми можемо відкрити меню налаштування IDE, де, наприклад, налаштувати лінтер, або шрифт і тему коду

- Edit. Ця вкладка використовується для редагування коду. Дозволяє редагувати файл, наприклад дублювати лінію, копіювати та вставити код, відсортувати лінії коду
- View. Дозволяє налаштувати відображення виду IDE, відкрити додаткові панелі, наприклад, контролер версій Git та панель структури файлу
- Navigate. Полегшує навігацію в проекті, дозволяє шукати потрібний фрагмент коду, файл, або клас
- Code. Використовується для форматування коду, дозволяє очистити код від непотрібних відступів, зайвих пробілів та рядків, що не використовуються. Найчастіше використовується після написання готової частини коду
- Refactor. Містить інструменти, спрямовані на автоматизацію процесу рефакторингу засобами перейменування файлу/класу/змінної у всьому проекті. Webstorm сам знаходить, де саме використовується необхідний об'єкт і перейменовує його
- Run. Служить для запуску скриптів, що декларуються в package.json, або внутрішніх скриптів, які допомагають у розробці виконуючи різні функції
- Tools. Вкладка, що використовується для інших інструментів. Найчастіше в ній розміщуються налаштування різних плагінів Webstorm. Також містить інструменти для дебагу, наприклад, HTTP-Client, який дозволяє тестувати backend за допомогою написання запитів до нього
- VCS. Version Control System. Окрема вкладка для контролера версії. Дозволяє більш детально працювати з Git.
- Window. Налаштування відображення вікон, вкладок, панелей Webstorm

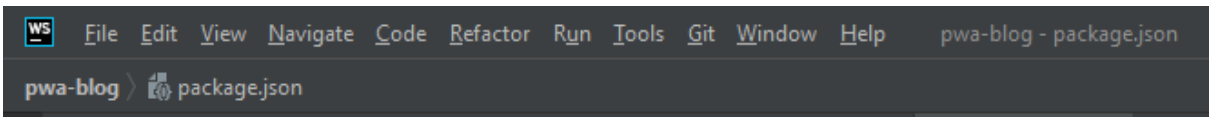


Рисунок 1.5 - Панель керування Webstorm

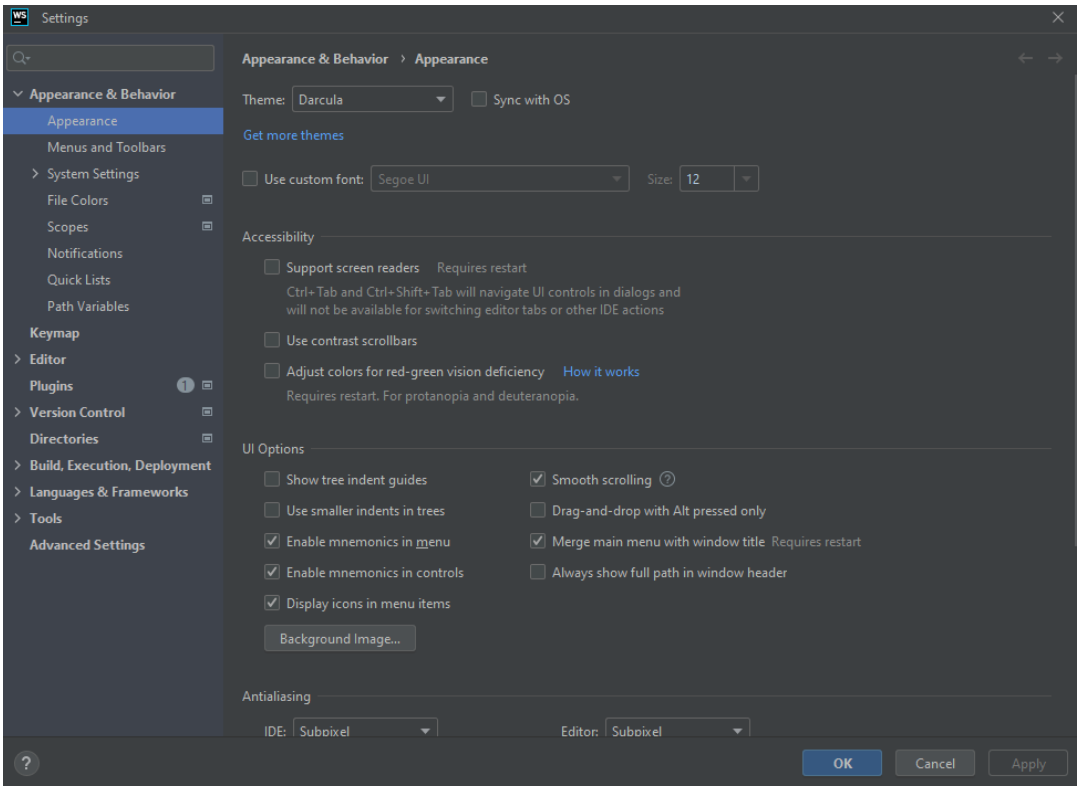


Рисунок 1.6 - Налаштування IDE Webstorm

1.4. Розробка front-end програми за допомогою фреймворку Vue

Компонент Vue - базовий елемент, що перевикористовується, який містить інкапсульовану логіку для своєї роботи. Також можна використовувати інші компоненти vue для спрощення власної логіки.

Створення компонента vue виконується за допомогою контекстного меню менеджера файлів Webstorm і вибору категорії New->Vue Component, після чого вказується його назва. Як правило, компоненти Vue називаються за допомогою

CamelCase, коли кожне слово у назві пишеться з великої літери. Приклад такої назви буде AuthPage.vue.

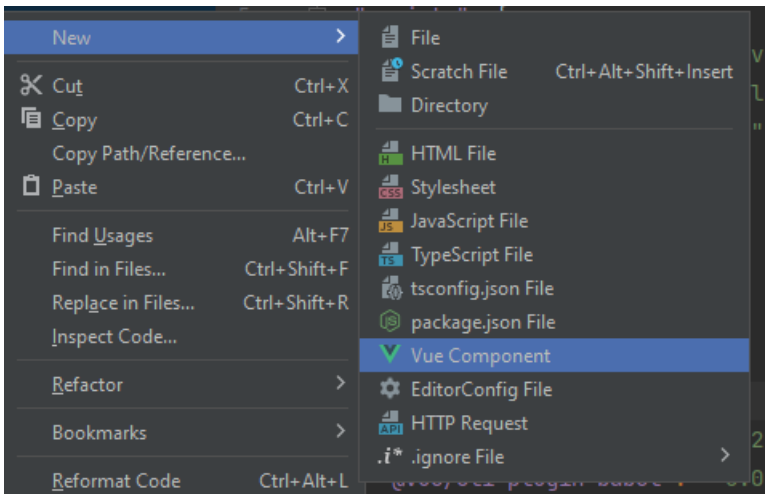


Рисунок 1.7 - Контекстне меню директорії Webstorm

Після створення Webstorm сам генерує шаблон компонента, що дозволяє програмісту економити час. Структура самого компонента виконана за принципом W3C, що також використовується в HTML, що дозволяє нам використовувати один стиль при розробці проекту у всіх його частинах, на відміну, наприклад, від React, іншого популярного front-end фреймворку.

Компонент vue повинен містити тег <template>, у якому буде міститися HTML-код для рендерингу. У цьому вся тезі ми використовуємо стандартні теги HTML крім деякого розширення синтаксису фреймворком vue.

Вся логіка компонента повинна бути описана в тегу script, який служить об'єктом, що дозволяє задекларувати методи, змінні, компоненти на які ми посилаємось і використовувані бібліотеки в коді. Декларація виконана як об'єкт JS з полями:

- Name. ім'я компонента

- Data. Функція, яка дозволяє створити змінні, що використовуються у компоненті
- Methods. об'єкт створення локальних функцій використовуваних всередині компонента
- Computed. Обчислювані властивості. Є однією з особливостей vue і спрощує читабельність коду. Служить до створення віртуальних змінних, описуваних як функцій. Може змінювати значення при зміні інших, що з нею, властивостей. Є хорошим прикладом роботи реактивності у Vue
- Components. Об'єкт, який виконує функції оголошення компонентів, що імпортуються, що використовуються в тезі <template>

pwa-blog - App.vue

```
<template>
  <img alt="Vue logo" src="./assets/logo.png">
  <HelloWorld msg="Welcome to Your Vue.js App"/>
</template>

<script>
import HelloWorld from './components/HelloWorld.vue'

export default {
  name: 'App',
  components: {
    HelloWorld
  }
}
</script>

<style>
#app {
  font-family: Avenir, Helvetica, Arial, sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  text-align: center;
  color: #2c3e50;
  margin-top: 60px;
}
</style>
```

Рисунок 1.8 - Стандартний шаблон компонента при створенні проекту за допомогою vue-cli

Крім цього компонент vue може використовувати різні події малювання, створення та видалення компонентів для того, щоб програміст міг створити/видалити різні інстанси об'єктів для уникнення витоків пам'яті або будь-яку іншу логіку програми.

<style> всередині компонента який описує стилі css. Використовується це для стилізації кнопок, форм, елементів навігації, списків, тексту і т.д. Якщо до тега не доданий атрибут scoped, то стиль буде застосовуватися до всього проекту, інакше стиль буде локальним для компонента.

1.4.2. Робота з CSS-фреймворком tailwind для верстки

Для стилізації ми використовуватимемо фреймворк Tailwind. Особливістю даного фреймворку є декларативний підхід до написання стилів замість стандартних препроцесорів sass/less, які використовують імперативний підхід до написання стилів.

У випадку з Tailwind ми можемо надавати кожному тегу html класи, що описують необхідний вид елемента. Наприклад, за допомогою класу w-[1-96] ми можемо задавати будь-яку довжину контейнера, цифри в кінці вказують на ширину в rem, що дорівнює 1rem = 16px.

Для встановлення Tailwind ми можемо скористатися vue-cli. Для цього необхідно відкрити папку з програмою в консолі. Варто врахувати, що Webstorm дозволяє безпосередньо працювати з терміналом без згортання IDE, що в нашому випадку дуже зручний варіант. Відкрити термінал можна за допомогою комбінації клавіш alt+f11. У терміналі виконуємо команду vue add-h для перегляду доступних параметрів команди. Нам буде потрібний пакет tailwind, встановимо його за допомогою vue add tailwind. vue-cli сам зробить всі необхідні

дії, а саме: Додасть залежності до package.json, створить конфігураційні файли в корені проекту, виконає встановлення всіх залежностей з репозиторіїв npm.

```
A:\Projects\pwa-blog>vue add tailwind
 WARN  There are uncommitted changes in the current repository, it's recommended to commit or stash them first.
? Still proceed? Yes

   Installing vue-cli-plugin-tailwind...


added 1 package, and audited 938 packages in 2s

96 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
 ✓  Successfully installed plugin: vue-cli-plugin-tailwind

? Generate tailwind.config.js full

   Invoking generator for vue-cli-plugin-tailwind...
   Installing additional dependencies...


added 17 packages, and audited 955 packages in 4s
```

Рисунок 1.9 - Встановлення пакету tailwind за допомогою vue-cli у терміналі webstorm

Після встановлення tailwind створить конфігураційний файл tailwind.config.js. У ньому ми можемо налаштувати колірну палітру, задати аліаси для розмірів екрану в px, налаштувати інші класи, які мають параметри, наприклад, до класів w-* ми можемо додати w-100, який дорівнюватиме 100 rem. Це досягається шляхом того, що tailwind не зберігає готового css коду, а генерує його щоразу після складання проекту залежно від налаштувань та використовуваних класів у проекті. Таким чином, у кінцевий білд потрапляє лише використовувані нами класи, що зменшує вагу програми.

1.4.3. Роутінг сторінок за допомогою Vue (vue-router)

Роутінг - одна з особливостей сучасного підходу до проектування програм. Дозволяє без зайвих завантажень переміщатися сайтом із збереженням стану сторінки. Це означає, що ми можемо не використовувати back-end для віддачі статичних html файлів, а "вшити" їх у нашу програму. Такі програми називаються SPA (Single Page Application). Так як при складанні проекту ми отримуємо один html файл і папку з js-скриптами, це дає можливість нам зручно кешувати весь наш додаток за допомогою service-workers, що є одним із найголовніших завдань PWA – роботою без підключення до інтернету.

За роутинг у vue відповідає плагін vue-router. Установка його нічим не відрізняється від tailwind, за винятком назви пакету vue-router. Після встановлення нам необхідно створити об'єкт роутера для подальшого використання.

Робиться це за допомогою імпорту методу createRouter з пакета vue-router.

Даний метод-конструктор приймає 2 параметри, routes та history.

Routes реалізує масив, що містить об'єкти віртуальних сторінок, які містять назву, віртуальний url і посилання на компонент vue, що відображається на цій сторінці

- History, задає режим роботи роутера з віртуальними url. На сьогоднішній день існують такі режими
- Hash History. Є стандартним навігаційним режимом. Додає до кінця url символ # і після нього пише віртуальний шлях до сторінки. Це дозволяє серверу не реагувати на роутинг з боку vue при запитах. Найоптимальніший режим роботи

- Web History Mode. Використовує нативно url рядок, змінюючи його залежно від поточної віртуальної сторінки. Вимагає певних настройок веб-сервера. Однак дозволяє використовувати URL рядок, наприклад, при редиректі з стороннього веб сайту.
- Memory Mode. Є віртуальним адресним рядком. Не зберігає свій стан у браузері при перезавантаженні. Необхідна лише використання на сервері для реалізації т.зв. Server Side Rendering

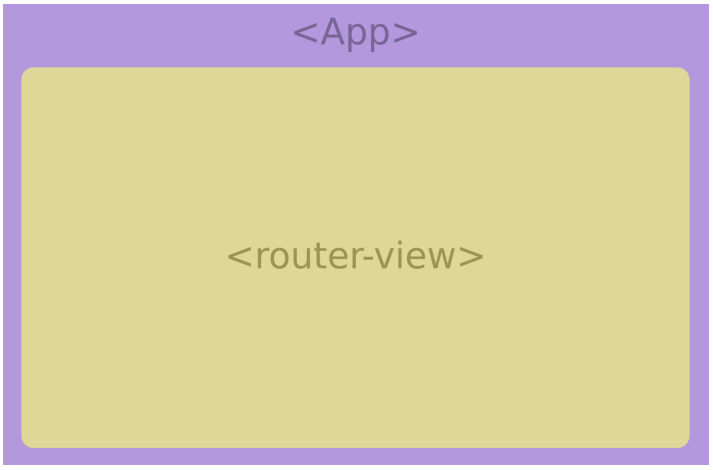
Як видно, наприкінці ми експортуємо інстанс роутера (мал. 1.10), який у майбутньому необхідно буде додати до main entry (точку входу) для подальшого використання нашим додатком.

```
1  import { createRouter, createWebHashHistory } from 'vue-router'
2  import MainPage from "@/Pages/MainPage";
3  import RegisterPage from "@/Pages/RegisterPage";
4
5  const routes = [
6      { path: '/', component: MainPage },
7      { path: '/register', component: RegisterPage },
8  ]
9
10 const router = createRouter({
11     history: createWebHashHistory(),
12     routes
13 })
14
15 export default router
```

Рисунок 1.10 - Приклад використання пакета vue-router в коді

Роутінг також вимагає початковий компонент, який буде завантажуватися в першу чергу. Це може бути шаблон програми, або його навігація, головне щоб у цьому компоненті був використаний тег <router-view>, який вказує vue-router де

саме відображати посилання компоненти за віртуальним посиланням. Схематично розміщення роутера у шаблоні виглядатиме таким чином (рис. 1.11)



. Рисунок 1.12 - Схематичне зображення розміщення роутера віртуальних сторінок усередині статичного шаблону

Навігація за віртуальними посиланнями здійснюється через доступний для всіх компонентів об'єкт “$router”. З його допомогою ми можемо через код керувати рухом між сторінками всередині <router-view>. Наприклад, нам доступні функції програмної навігації .push(:object) і .go(:int). push дозволяє перенаправити віртуальну сторінку на інший url (компонент), а go дозволяє переміщатися між історією відвідувань сторінок як вперед, якщо вказати позитивне число, так і назад, якщо негативне.

```
<a
  class="font-medium text-blue-600 hover:text-blue-500 cursor-pointer"
  @click="$router.push('/register')">
  Зарегистрироваться
</a>
```

Рисунок 1.13 - Приклад використання програмної навігації всередині компонента для переходу на сторінку реєстрації

1.4.4. Складання фронтенду

Оскільки в проекті ми використовуємо ECMAScript 6(ES6) , який дозволяє використовувати нові можливості мови (import, export), необхідно перетворити його на код, зрозумілий браузеру . Для цього потрібно використовувати плагін babel для webpack. Вебпак – це збирач модулів. Він аналізує модулі, створює списки залежностей, потім збирає модулі в правильному порядку в один бандл.

Процес збирання починається з точки входу. У ній ми імпортуємо всі необхідні модулі, плагіни vue, допоміжні бібліотеки. Також webpack дозволяє імпортувати не тільки JS-файли, але ще й різні статичні залежності, такі як png картинки, jpg картинки, svg файли, Cascading Style Sheets (CSS), Leaner Style Sheets (LESS), Syntactically Awesome Style Sheets(SASS) згодом компілюючи та його. Картинки webpack оптимізує, стискаючи їх, різні мови стилів (less, sass), він збирає в зрозумілий браузеру css. Це все досягається окремими плагінами (лоадерами), тому програміст може завантажити додаткові лоадери з репозиторіїв для будь-якого випадку.

Webpack може збирати додаток у двох режимах, development та production. Поділ на режими виконано більш логічно, вони не мають майже жодного впливу на зібраний проект (крім стиснення та зменшення ваги), а створені лише для поділу змінних середовищ. Це дозволяє зручно створювати середовище розробки та середовище продакшену, в яких, наприклад, використовуються різні сервери API або будь-які інші параметри.

Змінні зберігаються у файлах .env та .env.production у форматі NAME=VAUE.

Точка входу виглядає як звичайний JS-файл, в ньому прописаний імпорт vue-router, tailwind-css та самого vue. Крім того, webpack дозволяє реалізувати ліниве завантаження (lazy-loading) модулів бібліотек. Такий підхід дозволяє зменшити вагу кінцевого бандла і, відповідно, збільшити швидкість завантаження кінцевого користувача. При лінивому завантаженні модулі підвантажуються при необхідності для клієнта динамічно. Однак мінусом такого способу є те, що користувачеві необхідно чекати перед використанням якоїсь функції, якщо модуль, що підвантажується, багато важить, а у користувача повільний інтернет, це може істотно зменшити комфорт користувача від використовуваної програми. Тому необхідно дотриматися балансу, між початковою швидкістю завантаження і модулями, що підвантажуються, за допомогою лінивого завантаження

Так як наш проект не використовує занадто багато сторонніх бібліотек і у зв'язку з чим важить досить мало, ліниве завантаження у цьому проекті не використовувалось

```
1  import { createApp } from 'vue'
2
3  import App from './App.vue'
4  import routers from "@/routers";
5
6  import './assets/tailwind.css'
7
8  let app = createApp(App);
9  app.use(routers);
10 app.mount('#app');
```

Рисунок 1.14 - Точка входу у проект

Для того, щоб не компілювати весь проект щоразу при тестуванні програми, так як це займає досить багато часу в порівнянні зі звичайними проектами без webpack (близько 10 секунд для базового проекту), можна використовувати вбудований сервер розробки (dev-server) з плагіном швидкої заміни модулів (HMR) він збирає проект один раз при запуску і далі перезбирає тільки змінені частини коду.

Підхід з HMR є загальноприйнятим у розробці, оскільки дозволяє заощадити багато часу на компіляції програми. Однак основним мінусом такого способу є те, що деякі частини коду не можуть бути перезібрані, наприклад, точка входу, або якісь конфіги, оскільки вони є "основними", на яких будуються інші модулі.

1.5. Розробка back-end з технологією Restify

Restify - один із найпопулярніших фреймворків для розробки back-end на NodeJs. Він створений для побудови правильних проектів REST-API. Поєднує в

собі методи розробки, що використовуються в express і високу швидкість роботи. Плюсами даного фреймворку є

- Швидка швидкість роботи, набагато вища, ніж у конкурентів в особі express і hapi, це досягається шляхом того, що restify не перевантажений зайвими модулями, дозволяючи програмісту самому вирішувати які модулі і коли використовувати.
- Підтримка TypeScript(TS). Це дозволяє програмісту використовувати ООП концепти при розробці на JS. Restify типізує всі свої методи, завдяки чому полегшує розробку в рази.
- Підтримка модулів Express. Незважаючи на те що restify має велику підтримку спільноти, яка розробила багато своїх плагінів, то число все одно менше, ніж кількість плагінів для express. У зв'язку з чим розробниками було прийнято рішення розробляти restify з використанням інтерфейсів express, що дає сумісність їх плагінів та суттєво розширює кількість доступних плагінів
- Зручність навчання. Так як restify має схожий синтаксис з express, перенавчитися на нього не становить великої проблеми для розробника-початківця.

Однак не дивлячись на всі плюси restify, він все ще вважається досить "перевантаженим" фреймворком через безліч особливостей, що не використовуються, в більшості проектів на ньому. Зокрема restify реалізує застарілі технології, наприклад SPDY, що зараз замінюється HTTP/2.

Структура проекту на restify є “вільною”, програміст сам може вирішувати у яких папках та які частини коду зберігати. Однак у співтоваристві прийнято використовувати наступний патерн структури для типізації проектів.

Як і vue, весь код зберігається в папці src, ця папка містить наступні частини

- Controller. Реалізує всю "бізнес-логіку". Тут програміст управляє спілкуванням з моделями, обробкою вхідних даних та віддачею відповіді користувачам.

- Helpers. Допоміжні модулі, що використовуються для інкапсуляції окремих, логічних частин коду, наприклад, JWT-модулів.

- middleware. Є загальноприйнятим стандартом для розробки API на будь-яких мовах програмування. Потрібна для інкапсулювання деяких функцій для обробки запиту. Викликаються по ланцюжку від першого до останнього middleware. Зазвичай найпоширенішим прикладом застосування middleware є перевірка авторизації/прав користувача.

- models. Зберігає різні моделі. Використовується для типізації різноманітних форм, компонентів, таблиць БД. Останнє зберігається в окремій папці та декларується певною структурою завдяки ORM-системі TypeORM.
- routes. Реалізує роутинг бек-енду. У ній декларується різні шляхи, форми валідації, прописуються необхідні middleware та методи запити та url обробки даних.
- validation. Створено для зберігання форм валідації даних. Реалізується за допомогою бібліотеки express-validation. Усі форми мають декларативний вигляд, тобто кожному полю прописуються певні вимоги до виду. Крім цього express-validation може змінювати вступні поля, наприклад, для випадків, коли користувач міг поставити зайву прогалину у формі авторизації

Всі бібліотеки та плагіни підключаються у файлі app.ts, в якому програміст імпортує сам restify, всі роути та middleware, ініціалізує підключення до БД і прописує параметри веб-сервера, такі як порт, що прослуховується, і IP-адресу. Як і в vue, проект restify має файли package.json і package-lock.json.

1.5.1. Робота з базою даних за допомогою TypeORM

TypeORM – ORM бібліотека для роботи з БД. Створена для роботи з TypeScript, на якому програміст може суворо типізувати певні поля в БД та в моделях, завдяки чому, TypeORM має одну з найголовніших переваг на мій погляд – можливість створення таблиць прямо з коду TS. Це реалізується за рахунок створення програмістом моделей та припису до їх полів декораторів - однією з особливостей TS. Декоратори повідомляють TypeORM, що дане поле в моделі - це поле в БД, а також дозволяє задати деякі налаштування, такі як значення за замовчуванням, можливість поля бути порожнім, або тип даних, що зберігалися в полі.

TypeORM також має дуже важливу особливість, яка дозволяє працювати з безліччю СУБД (у тому числі і об'єктними) в одному стилі і без зміни синтаксису. Це досягається шляхом патерну реалізує стандартний інтерфейс для взаємодії з базою даних. Також це дозволяє створити програмісту власний контролер для сторонньої СУБД.

Щоб почати використовувати TypeORM, необхідно створити т.зв. DataSource, в якому прописуються всі необхідні параметри, включаючи використовувану СУБД, її дані для підключення, використовувані моделі, що імпортуються з папки models і налаштування синхронізації. Синхронізація – можливість вимкнути синхронізацію моделей та БД. Якщо синхронізація увімкнена, TypeORM змінюватиме типи полів та їх властивості залежно від моделей у коді. Ця функція дозволяє зручно тестувати програми без необхідності вручну вносити зміни до БД. Як правило, синхронізацію відключають під час використання програми у продакшені для недопущення випадкових змін у БД, які можуть спричинити втрату всіх даних.

Замість синхронізації на продакшен прийнято використовувати міграції. Вони дозволяють виконувати безпечну зміну таблиць без увімкнення синхронізації. Міграції виконані у вигляді класу, що реалізує інтерфейс MigrationInterface. У класі є 2 методи, up та down. Обидва повинні бути заповнені SQL-кодом, що виконує міграцію. Up це метод, що виконується при виконанні міграції, безпосередньо вносить правки до БД. Down потрібен для відкату БД до минулого стану. Міграції прописуються як migration в datasource. TypeORM сам версіонізує міграції і зберігає вже скоєні в спеціальній таблиці migrations, це дозволяє будь-якої миті скасувати відразу кілька міграцій і відкотити структуру БД до будь-якої точки часу.

При роботі з даними TypeORM реалізує патерн Builder, це ланцюжок викликів функцій із заданими параметрами для пошуку/вставки даних. Такий патерн максимально спрощує роботу із СУБД, оскільки збільшує читабельність коду.

```
const firstUser = await dataSource
    .getRepository(User)
    .createQueryBuilder("user")
    .where("user.id = :id", { id: 1 })
    .getOne()
```

Рисунок 1.15 - Приклад QueryBuilder використовує datasouce. У прикладі ми отримуємо користувача з ID 1

Для створення нового запису необхідно створити інстанс моделі за допомогою ключового слова new і виконати метод .save реалізується класом BaseModel, який

успадковується моделлю. Цей метод повертає Promise, який необхідно обробити за допомогою ключового слова await, після чого ми можемо бути впевнені що запис вже був доданий в БД. Таким чином TypeORM реалізує один з головних концептів коду на JS - асинхронність, що дозволяє під час того, як виконується збереження в БД, виконати якесь інше завдання, наприклад, створити інший запит і виконувати їх паралельно.

Також для реляційних СУБД програміст може прописати в моделі relations і отримувати за зовнішнім ключем записи, що належать до моделі. TypeORM підтримує всі існуючі типи відносин, а саме

- One-To-One. Коли 1 запис має 1 пов'язану з нею запис іншої таблиці. Декларується декоратором @OneToOne. Прикладом такого запису є користувач до фото профілю
- One-To-Many. Один запис має відношення до багатьох інших записів. Декларується декоратором @OneToMany. Прикладом є користувач до публікацій
- Many-To-One. Є зворотним ставленням до One-To-Many. Декларується декоратором @ ManyToOne
- Many-To-Many. Безліч записів мають відношення до багатьох інших записів. Декларується декоратором @ ManyToMany. Прикладом такого зв'язку будуть передплатники/друзі користувача.

```typescript
import {Entity, Column, PrimaryGeneratedColumn, BaseEntity} from "typeorm"


@Entity()
export class User extends BaseEntity {
    @PrimaryGeneratedColumn()
    id: number

    @Column({
        unique: true,
    })
    login: string

    @Column()
    password: string

    @Column({ type: "timestamp", default: () => "NOW()" })
    createdAt: number;
}
```

Рисунок 1.16 - Приклад моделі користувача з використанням типізації полів

Також TypeORM має одну перевагу перед іншими ORM-бібліотеками, це передплата різних подій. Завдяки цьому програміст може інкапсулювати різну логіку, що виконується при створенні/оновленні/видаленні будь-яких даних у БД. Це дозволяє, наприклад, створити систему повідомлень користувача при нових повідомленнях, або постах на сторінках, на які він підписаний, або застосувати патерн оновлень у реальному часі, наприклад, за допомогою socket.io. Дані події прописуються у властивості subscribers. Клас передплатника повинен реалізовувати інтерфейс EntitySubscriberInterface, у якому можна прописувати будь-які обробники подій як методів. TypeORM дозволяє реалізувати наступні події

- AfterLoad. Викликається щоразу при запиті моделі у будь-якому контексті. Дозволяє, наприклад, оновлювати лічильник переглядів посту.
- AfterInsert. Викликається після створення нового запису БД. Як приклад застосування може бути оповіщення користувачів про нові посади, або передплатників

- AfterUpdate. Викликається під час оновлення даних. Дозволяє повідомити користувачів, наприклад, про зміну пароля
- AfterRemove. Викликається при видаленні даних.

Для оптимізації запитів до бази даних TypeORM дозволяє кешувати результат запитів. Ця властивість є спірною, оскільки може мати нечітку логіку кешування і може призвести до багів у проекті. Натомість краще вручну кешувати необхідні частини, наприклад, за допомогою БД Redis. Це key-value БД, тобто на 1 ключ може бути лише 1 значення, завдяки чому Redis має вкрай швидку швидкість обробки запитів у порівнянні з типовими СУБД. Однак redis зберігає дані в ОЗУ, що може спричинити втрату даних при перезапуску сервера, у зв'язку з чим його зазвичай використовують як прошарок між СУБД із гарантованим записом на диск та програмою. Оскільки для кеша втрата даних не є проблемою, redis ідеально підходить для таких завдань.

1.5.2. Використання middleware у ланцюжку запитів

Middleware виконує роль посередника між вхідним запитом та його обробкою роутингом. Реалізується як функції, що приймає об'єкти req, res і метод next. Після виконання логіки middleware на основі прийнятого рішення необхідно викликати функцію next, яка повідомить restify що можна перейти до наступного посередника, або перейти до виконання методу роутингу, або скасувати запит повернувши користувачеві повідомлення про помилку за допомогою res.send. Найчастішим прикладом застосування middleware є виконання перевірки авторизації користувача для підтвердження його доступу до захищених URL. Для цього можна використовувати технологію JWT, яка дозволяє створити спеціальний ключ для верифікації користувача. Структура JWT складається із 3 частин. Використовуваного методу підпису, даних та цифрового підпису.

Middleware з jwt під час виклику повинен перевірити заголовки на наявність ключа, далі перевірити його підпис та розшифрувати дані. У випадку, якщо ключа немає, або підпис невалідний, middleware повинен віддати користувачеві помилку, робиться це за допомогою res.code(401) який віддасть клієнту помилку 401(Unauthorized), в іншому випадку, якщо перевірка JWT пройшла успішно, посередник повинен передати керування наступним у ланцюжку викликів функції за допомогою методу next().

```
1  import { Jwt } from "../helpers/jwt";
2  import {IJwtAccessToken} from "../models/JwtTokenModel";
3  import {Next, Request, Response} from "restify";
4
5  export default (req: Request, res: Response, next: Next) => {
6      if(!req.headers.authorization) return res.send(401);
7
8      const encodedToken = req.headers.authorization.replace("Bearer ", "");
9      const token = Buffer.from(encodedToken, "base64").toString("utf8");
10
11     if (!token) return res.send(401)
12
13     try {
14         const payload = Jwt.verify<IJwtAccessToken>(token);
15         if(!payload) throw new Error("Invalid token");
16
17         next();
18     } catch (err) {
19         return res.send(401);
20     }
21 }
```

Рисунок 1.17 - Приклад проміжного ПЗ для перевірки JWT-токена відправленого клієнта

Крім виконання middleware до роутингу, в деяких випадках може знадобитися виконати якісь дії вже після виконання основної логіки коду, для цього існують механізм post-routing, який виконується останнім у ланцюжку викликів і може на

основі якихось даних міняти, або скасовувати відповідь користувачеві. Як правило, така методика застосовується для логування та метрики якихось даних із сервера, що дозволяє проводити моніторинг помилок або іншої інформації на back-end.

1.5.3. Авторизація з використанням JWT-токену

JWT – Json Web Token, відкритий стандарт для створення токенів доступу. Використовується для перевірки користувачів на доступ до запитуваного ресурсу. Створено на противагу звичному механізму "сесій" та API-ключів. Головною перевагою JWT є те, що серверу немає необхідності робити запити до БД або будь-якому іншому сховищу даних для перевірки валідності та даних користувача за рахунок того, що всі необхідні дані зберігаються прямо в токені, а сервер лише перевіряє їх цифровий підпис. Для перевірки серверу потрібен лише секретний ключ, який використовується для хешування даних. Так як немає потреби в сховищі даних для JWT, він є хорошим рішенням для проектів, які використовують мікро-сервісну архітектуру, кожен сервіс зберігає лише необхідні для його роботи дані в бд і може бути гео-віддаленим від інших без втрати продуктивності.

У контексті перевірки авторизації застосування JWT виглядає так. Існують 2 токена, маложивучий (access token) і довгоживучий (refresh token). Access token передається при кожному запиті для ідентифікації користувача, має життя близько 5-15 хвилин і повинен перезапускатися щоразу при закінченні. Refresh token служить для перевипуску access token, має довгий термін життя, зазвичай 1-7 днів. Для перевипуску токенів використовується спеціальний ендпоінт на back-end, який робить запит із refresh token до БД, інвалідує старий refresh token і випускає новий.

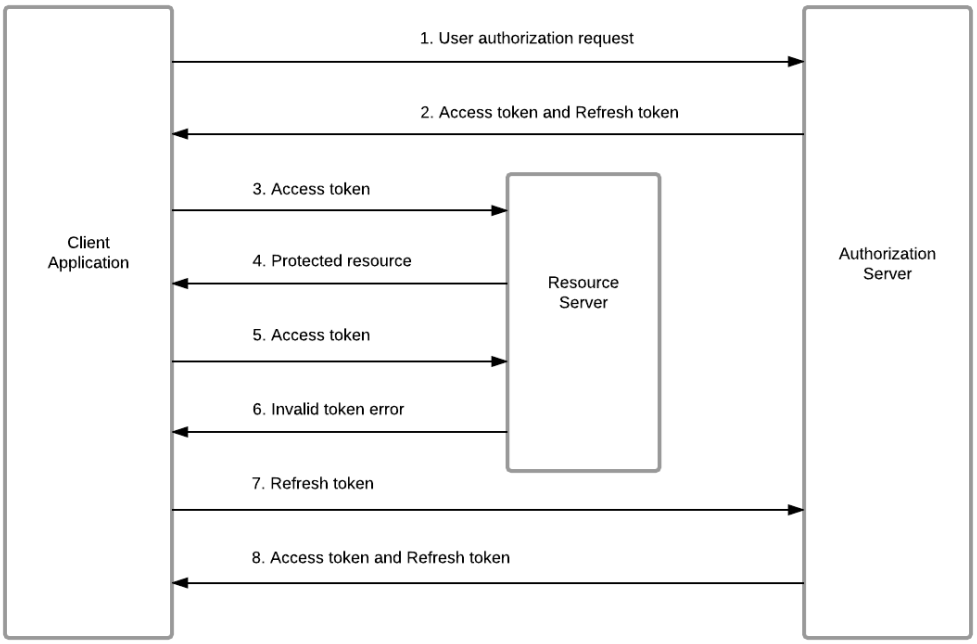


Рисунок 1.18 - Схема авторизації з використанням access token та refresh token

JWT це відкритий стандарт, тому може використовувати різні методи шифрування даних, зокрема і асиметричні. Щоб не зберігати секретний ключ у кожному сервісі, ми можемо розділити ключ на приватний та публічний. Приватний буде використовуватися сервером авторизації для створення підписів і публічний для їх перевірки. Таким чином ми убезпечимо програму від компрометації ключа і не дамо зловмисникам можливості випускати будь-які токени авторизації.

1.6. Перетворення сайту до PWA-додатку

Після створення сайту необхідно привести його до стандартів PWA. На даний момент PWA підтримується багатьма веб-браузерами, проте "двигуном" цієї технології є Google Chrome, який активно просуває її в маси.

Критерії, які дають сайту право називатися "прогресивним додатком", є наявність маніфесту та обробка кешу сервісним працівником.

Маніфест служить маркером для браузера, що цей сайт варто сприймати як прогресивну програму. Підключається маніфест через заголовки HTML. Він повинен бути виконаний у стандарті json-файлу та мати наступні параметри:

- start_url корінь програми, потрібен для завантаження статики
- name. Ім'я PWA-програми, відображається в “меню додатків” у браузері
- display. Режим відображення програми. від цього параметре залежить кількість відображених елементів керування браузерів у вікні програми
- icons. Додаток значок, аналог favicon для вебсайту

Крім маніфесту, PWA повинен мати щонайменше 1 service worker (сервісний працівник). Це одна з особливостей сучасних сайтів, яка вміє реєструватися на рівні програм та перехоплювати/модифікувати запити навігації та ресурсів. Являє собою JS-файл, що реалізує події обробки. Найпростішим способом застосування service worker є кешування сторінки та відповідей від api, або модифікація логіки роботи для доступності програми без підключення до інтернету.

Після виконання ключових дій програма може називатися PWA і бути встановлена як нативна на платформах Android/Windows.

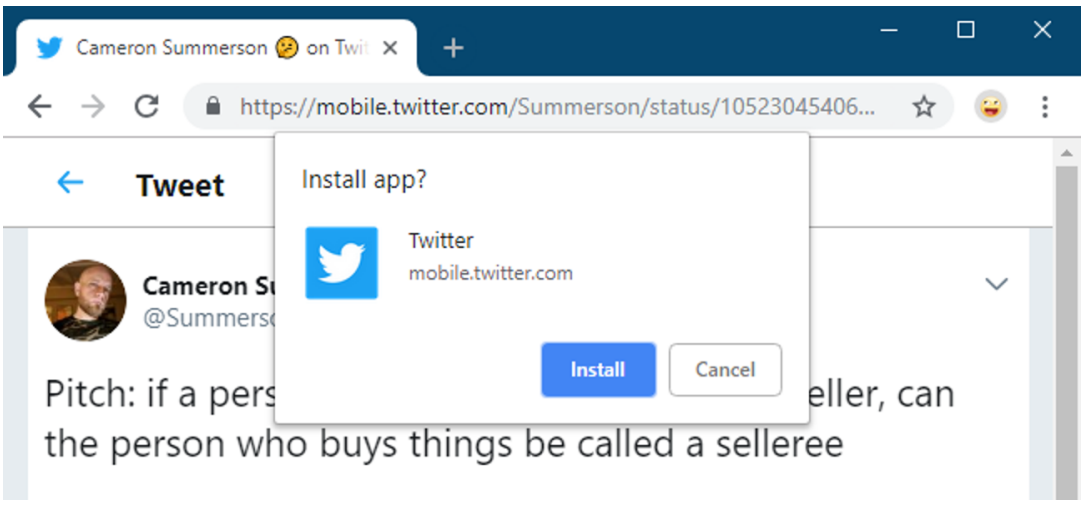


Рисунок 1.19 - Приклад установки програми PWA

Після встановлення на пристрій, PWA буде доступно для запуску як ярлик на робочому столі або в меню програм браузера. Запускатися PWA буде згодна з настройками з маніфесту, або з інтерфейсом браузера, або як самостійна програма

2. ЕКОНОМІЧНА ЧАСТИНА

2.1. Резюме

В даному дипломному проекті створена розробка веб-додатка із застосуванням сучасних технологій [PWA]

Ефективність кожного програмного продукту визначається його якістю та ефективністю процесу розробки. Якість ПП визначається наступними складовими: з точки зору користувача; з позиції використання ресурсів; виконання вимог до програмного забезпечення.

Оцінка якості програмного продукту з точки зору користувача визначається необхідним на стадії функціонування розміром оперативної пам'яті ЕОТ, витратами машинного часу, пропускною спроможністю каналів передачі даних. Оцінка якості програмного продукту включає визначення трудомісткості і вартості його створення.

2.2. Визначення трудомісткості розробки програмного забезпечення.

Тривалість розробки програмного продукту залежить від його обсягу, трудомісткості розробки, кваліфікації виконавців, а також планових термінів, визначених умовами ринку. Методом структурної аналогії по відповідних каталогах аналогів програмного забезпечення визначається обсяг програмних засобів, у тисячах умовних машинних команд програми аналога

Каталог аналогів

Таблиця 2.1

Найменування ПП	Обсяг функції ПП – V_o, усл. машинних командах.
1. ПП СУБД	2500 – 9800
2. ПП імітаційного моделювання	1800 – 8800
3. ПП введення інформації	1060 – 5750

У таблиці 2.1 представлені аналоги програмного забезпечення, функції яких, у більшому або меншому ступені, виконує розроблений програмний продукт. Для нашого варіанта виділено сірим кольором.

Вибравши аналог ПП, що містить Vo в умовних машинних командах, трудомісткості визначати на основі табл.2.2

Таблиця.2.2

Обсяг ПП, тис.умов.машинних команд	Норма часу, люд/год
1.00	229
2.00	244
3.00	262

На підставі отриманого значення, по довіднику, визначається укрупнена норма часу на розробку аналога програмного забезпечення (коректується поправочним коефіцієнтом враховуючої умови розробки ПП, тобто в умовах комп'ютера, Кк=0,7÷0,8): $T^a = = 244$ x 0,7 = 170,08 (люд/годин).

Трудомісткість програмного продукту визначається по кожному етапу розробки окремо на підставі трудомісткості аналога з урахуванням складності

розробки, ступеня новизни і ступеня використання в розробці стандартних модулів на підставі формул:

$$T_{ТЗ} = T^a p \times L_1 \times K_H \qquad (2.1)$$

$$T_{ТП} = T^a p \times L_2 \times K_H \qquad (2.2)$$

$$T_{РП} = T^a p \times L_3 \times K_H \times K_T \qquad (2.3)$$

Для розрахунку необхідні наступні коефіцієнти:

Li – питома вага і-го етапу розробки (див. табл. 2.2.);

Кн – поправочний коефіцієнт, що враховує ступінь новизни (див. табл.2.3.);

Кт – поправочний коефіцієнт, що враховує ступінь використання в розробці типових програм (див. табл. 2.4.).

Таблиця 2.2. Значення питомих коефіцієнтів трудомісткості стадії в загальній трудомісткості розробки ПП.

Код стадії	Ступінь новизни		
	А	Б	В
ТЗ (L_1)	0,15	0,12	0,12
ТП (L_2)	0,16	0,15	0,11
РП (L_3)	0,55	0,58	0,61

Для нашого варіанта виділено сірим кольором.

Таблиця 2.3. Значення поправочного коефіцієнта, що враховує ступінь новизни

Код ступ новизни	Ступінь новизни	Значення $К_н$
А	Принципово нові ПП	1,75 – 1,2
Б	ПП – розвиток визначеного параметричнс ряду	1,0 – 0,8
В	ПП маючий аналог	0,7

Для нашого варіанта виділено сірим кольором.

Таблиця 2.4. Значення коефіцієнта ступеня використання в розробці типових програм

Ступінь охоплення реалізованих функцій розроблювально ПП типовими програмами, %	Значення К
60 і вище	0,6
40-60	0,7
20-40	0,8
До 20	0,9

Для нашого варіанта виділено сірим кольором.

Тепер розраховуємо трудомісткість по кожному етапу окремо:

Трудомісткість технічного завдання

$T_{тз}=T^{a}*L_1*K_н$= 170,08*0,12*0,7= 14,35 (люд/годин) (2.1)

Трудомісткість розробки технічного проекту

$T_{тп}=T^{a}*L_2*K_н$=170,08*0,11*0,7 = 13,10 (люд/годин) (2.2)

Трудомісткість розробки робочого проекту

$T_{рп}=T^{a}*L_3*K_н*K_т$=170,08*0,61*0,7*0,7= 50,81 (люд/годин) (2.3)

Для подальших розрахунків визначили кількість папера, витраченого на кожен етап: технічне завдання $N_{тз}$= 2 (стр), розробка ТП $N_{тп}$=12 (стр), розробка робочого проекту $N_{рп}$=24(стр), пояснювальна записка відповідно $N_{пз}$=22(стр)Розрахунок зведений у таблицю 2.5

Таблиця 2.5. Розрахунок трудомісткості ПП

Найменування етапів	Розрахунок, годин.		
1	2	3	4
1.ТЗ	$T_{Pтз}$=14,35	$T_{кк}=0,7*N_{тз=}$ 0,7*2=1,4	$T_{нк}=0,15*N_{тз}$=0,15*2= 0,3
2.Розробка ТП	$T_{Pтп}$=13,10	$T_{кк}=0,7*N_{тп}$=0,7*12=8,4	$T_{нк}=0,15*N_{тп}$=0,15*12=1,8
3.Розробка РП	$T_{Pрп}$=50,81	$T_{кк}=0,7*N_{рп}$=0.7*24=16,8	$T_{нк}=0,15*N_{рп}$=0,15* 24=3,6
4.Розробка ПЗ	$T_{пз}$=1,5*22 $*N_{пз}$=33	$T_{кк}=0,7*N_{тз}$=0,7*22=15,4	$T_{нк}=0,15*N_{пз}$=0,15*22=3,3
Усього, в т.ч.:	ΣT=162,26		
- на розробку	$\Sigma T_р$=111,26		
- контроль керівника		$\Sigma T_{кк}$=42	
- нормоконтрол			$\Sigma T_{нк}$=9,0

2.3. Розрахунок ціни програмного продукту.

У цьому розділі для визначення ціни розраховуємо основну заробітну плату виконавців, матеріальні витрати, вартість машино – години і витрати на розробку ПО. Розрахунок основної заробітної плати виконавців приведений у таблиці 2.6. Відповідно до статті 8 «Закону про Державний бюджет України на 2022» встановлено мінімальну заробітну плату у місячному розмірі з 1 січня 2022 року - 6500 гривень; мінімальну погодинну тарифну ставку – 39.26 грн.

Таблиця 2.6 Розрахунок основної заробітної плати виконавців.

Найменування робіт	Трудомісткість робіт, години	Погодинна тарифна ставка,	Розрахунок, гр
1.Розробка ПП	111,26	45,00	5006,25
2.Контроль керівника	42	60,00	2520
3.Нормоконт-роль	9	60.00	540
Усього			8066.25

Зробимо розрахунок матеріальних витрат на розробку ПП. Розрахунок зведемо в таблицю 2.7

Таблиця 2.7 Розрахунок матеріальних витрат на розробку ПО

Найменува матеріальних витрат	Тип, моде	Кількість	Ціна оди грн.	Вартість, грн.
Папір	Лист А4	60	2.5	150
				$В_{м\,і}$=150
Транспортно– заготівельні витрати (10%)				$Втр\_з = 0{,}1 \times Вм1 = 15.0$
Усього				$В_м=В_{мі}+В_{тр\_з}$ = 165,0

На підставі отриманих даних по окремих статтях витрат складена калькуляція планової собівартості в цілому ПП за формою, приведеною в таблиці 2.8.

Таблиця 2.8. Розрахунок статей витрат планової собівартості

Стаття витрат	Значен грн.	Формула розрахунку
1. Матеріали	165,0	$В_м$ (див. табл. 2.7)
2. Основна заробітна плата	8066.2	$З_о$ (див. табл. 2.6)
3.Додаткова заробітна плата	1209,9	$Зд = 0{,}15 \times Зо$ =0,15х8066.25
4.Відрахування до єдиного фонду соціального внеску	2040,7	$Вє.с.в. = 0{,}22 \times (Зо + Зд)$ =0,22х (8066.25+1209,93)
5. Накладні витрати	2419,8	$Внак. = 0{,}3 \times Зо$ =0,3х 8066.25
6. Повна собівартість	13901,	$С_{пов}$= $В_м$ +$З_о$ + $З_д$ +Вє.с.в. + Внак.=

Розмір прибутку, що включається в ціну, визначаємо по наступній формулі: $П=(С_{пов}*Р)/100=$ 13901,82* 10/100 =1390,18 грн (2.4)

Де Р– плановий рівень рентабельності (10-15%).

Оптова ціна (кошторисна вартість) визначається по формулы:

$Ц_о=С_{пов}+П$=13901,82+1390,18=15291,99 грн (2.5)

Податок на додану вартість визначаємо по наступній формулі ПДВ = 0.2 * $Ц_о$ = 0,2* 15291,99 = 3058,40 грн (2.6)

Виходячи з отриманих даних, ціна реалізації розробленого програмного продукту на основі наступної формули, становитиме:

$Ц_р=Ц_о$+ПДВ=15291,99+3058,40=18350,39 грн (2.7)

3. Охорона праці

3.1. Вступ

Створення безпечних умов праці на виробництві усіх форм власності було і залишається одним із головних пріоритетів держави.

Одним із найважливіших державних завдань є охорона праці та здоров'я громадян у процесі їх трудової діяльності.

Роботодавець зобов'язаний відповідно до Закону України «Про охорону праці» створити на кожному робочому місці умови праці відповідно до норм чинного законодавства, а також забезпечити додержання вимог законодавства щодо прав працівників у галузі охорони праці

3.2. Аналіз небезпечних та шкідливих чинників, що впливають на працівника.

Забезпечення безпечних і здорових умов праці в значній мірі залежить від правильної оцінки небезпечних та шкідливих виробничих факторів. Однакові по складності зміни в організмі людини можуть бути викликані різними причинами. Це можуть бути фактори виробничого середовища, надмірне фізичне і розумове навантаження, нервово-емоційна напруга, а також різне сполучення цих причин.

В даному розділі дипломного проекту розглядається питання охорони праці програміста . Оператори і програмісти зіштовхуються із впливом таких фізично небезпечних і шкідливих виробничих факторів, як підвищений рівень шуму, підвищена температура зовнішнього середовища, відсутність або недостатня освітленість робочої зони, електричний струм, статична електрика тощо.

3.3. Розробка заходів з охорони праці

На робочому місці програміста повинні бути створені умови для безпечної та високопродуктивної праці.

3.4. Виробничі приміщення

Площа на одне робоче місце становить не менше 6,0 м^2, а об'єм – не менше ніж 20,0 м^3. У приміщеннях слід щоденно робити вологе прибирання. Вони повинні бути оснащені аптечками першої медичної допомоги. При приміщеннях мають бути обладнанні побутові приміщення для відпочинку.

При кольоровому оформленні виробничих і допоміжних приміщень необхідно враховувати орієнтацію їхніх вікон стосовно частин світу і використовувати гармонійне сполучення кольорів. Для стін і робочих поверхонь використовують мало насичені (основні) кольори, для невеликих помешкань або ділянок, що рідко потрапляють у поле зору працюючих, а також для створення контрастності –у всіх приміщеннях стелі повинні бути білими. Поверхні устаткування в приміщеннях повинні бути матовими або напівматовими, для виключення кольори середньої насиченості (допоміжні), для маленьких по площі поверхонь – насичені (акценти) – як функціональне фарбування. Стелі у випадку відблисків світла в очі працюючого, а стіни бути пофарбованими фарбами пастельних тонів.

3.5. Мікроклімат робочої зони працівників, вентиляція.

Для створенная нормальних умов виробничої діяльності необхідно забезпечити чистоту повітря у виробничих приміщеннях.. Внаслідок виробничої діяльності у повітряне середовище приміщень можуть надходити різноманітні шкідливі речовини, що використовуються в технологічних процесах.

Шкідливі речовини, що потрапили в організм людини спричиняють порушення здоров'я лише в тому випадку, коли їхня кількість у повітрі перевищує граничну для кожної речовини величину.

У виробничих приміщеннях на робочих місцях мають забезпечуватись оптимальні значення параметрів мікроклімату: температури, відносної вологості й рухливості повітря – ГОСТ 12.1.005-88, СН 4088-86.

Параметри мікроклімату	Значення параме	значення параметрі
	взимку	влітку
Температура, C^0	22-24	23-25
Відносна вологість, %	40-60	40-60
Швидкість руху повітря, м	0,1	0,1-0,2

Для підтримки в приміщеннях нормального, що відповідає гігієнічним вимогам, складу повітря, видалення з нього шкідливих газів, пилу використовують вентиляцію. Механічна вентиляція (кондиціонери, вентилятори і т.д.) залежно від напрямку руху повітряних потоків, може бути витяжною, припливною і припливно-витяжною. При природній вентиляції (за допомогою вікон) повітря надходить у приміщення і видаляється з нього внаслідок різниці температур і тиску.. Механічна вентиляція забезпечується вентиляторами, що забирають повітря зовні і направляє його до будь-якого робочого місця. або устаткування, а також видаляють забруднене повітря.

3.6. Освітлення робочого місця, шум, вібрація

У приміщеннях, призначених для роботи з відео терміналами, доцільно, щоб вікна були орієнтовані на північ або північний захід. На вікнах повинні бути

штора або жалюзі, що регулюють рівень освітленості захищають від прямого влучення сонячних променів на робоче місце. Приміщення для роботи з ВДТ повинні мати природне та штучне освітлення. відповідно до ДБН В.2.5-28-2006. У приміщеннях, призначених для роботи з відео терміналами, доцільно, щоб вікна були орієнтовані на північ або північний захід. На вікнах повинні бути штора або жалюзі, що регулюють рівень освітленості і захищають від прямого влучення сонячних променів на робоче місце.

Для штучного освітлення у приміщенні використовуються люмінесцентні лампи типу ЛБ, які в порівнянні з лампами розжарювання мають ряд істотних переваг: за спектральним складом світла вони близькі до природного світла, мають підвищену світлову віддачу (у 2-5 разів вищу, ніж у ламп розжарювання); мають триваліший термін служби – до 10 тис годин.. Допускається застосування ламп розжарювання у світильниках місцевого освітлення.

У робочому приміщенні основними джерелами акустичних шумів є шуми ПЕОМ. Це коливання елементів електромеханічних пристроїв під впливом змінних магнітних полів тощо. Шум може викликати стомлення слуху й ослаблення звукового сприйняття, а також значне стомлення всього організму.

Оптимальні показники рівня шумів у робочих приміщеннях визначаються за ГОСТ 12.1.003-83. Припустимий рівень шуму при розумовій праці, що вимагає зосередженості – 50 дБ.

Рівні позитивних і негативних іонів у повітрі приміщень з ВДТ мають відповідати санітарно-гігієнічним нормам N2152

3.7. Електробезпека.

Для попередження поразок електричним струмом необхідно чітко й у повному обсязі виконувати правила провадження робіт і правил технічної експлуатації. Необхідно виключити можливість доступу оператора до частин устаткування, що

працює під небезпечною напругою, до неізольованим частинам, призначеним для роботи при малій напрузі й не підключеним до захисного заземлення, а також підводити електроживлення до ПЕОМ від розетки за допомогою спеціальної вилки із заземлюючим контактом.

3.8. Організація робочого місця користувача ПК

При організації робочого місця користувача ЕОМ з ВДТ необхідно врахувати:

- просторове розміщення і фарбування помешкань;
- особливості розміщення робочих місць у помешканні;
- параметри робочого столу і розміщення устаткування на ньому;
- конструктивні особливості робочого сидіння.

Просторове розміщення робочих місць у приміщенні повинне бути таким, щоб забезпечувати розміщення працюючого людини з урахуванням робочих рухів і переміщень відповідно до технологічного процесу, змін робочої пози, вільного доступу до місць профілактичного огляду і налагодження устаткування. Робочі місця повинні бути розташовані так, щоб у поле зору працюючого не попадали поверхні, що мають властивість віддзеркалювання, вікна й освітлювальні прилади. Відеотермінали повинні встановлюватися під кутом 90 – 100 градусів від вікон, так щоб світло падало з боку.

Робочі місця з ВДТ доцільно розміщати в глибині приміщення. При використанні в загальному освітленні світильників прямого світла робочі місця з ВДТ повинні бути обов’язково організовані в ряди паралельно стіні з вікнами. Розташування відеотермінала, при якому працюючий звернений обличчям або спиною до вікон, неприпустимо при будь-якому способі реалізації загального висвітлення, як прямим, так і відбитим світлом.

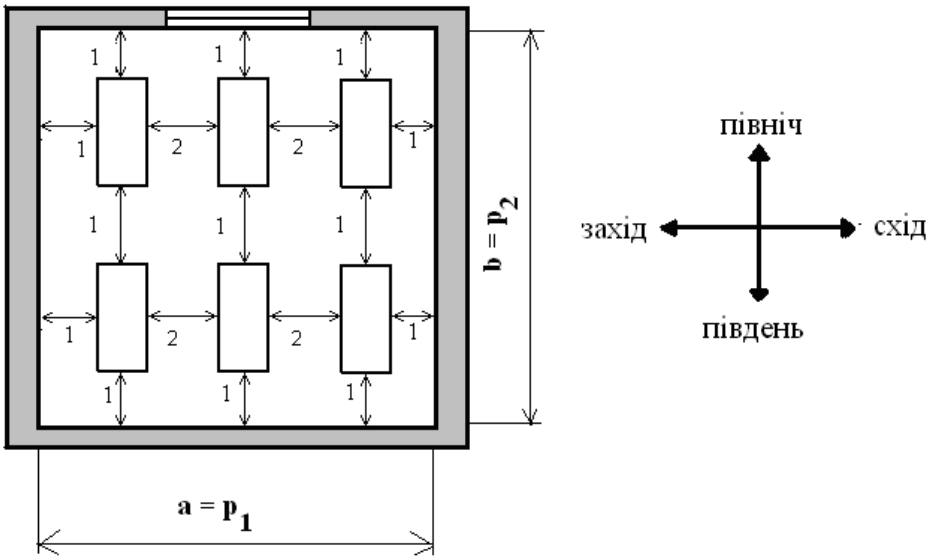


Рисунок 3.1 - Розташування робочих місць від інших об’єктів, м.

Робочий стіл повинний регулюватися по висоті в границях 680 – 800 мм, а ширина – забезпечувати можливість виконання операцій в зоні досяжності моторного поля. Рекомендовані розміри столу: висота: 725 мм, ширина 600 – 1400 мм, глибина 800 – 1000 мм. Оптимальні розміри робочого столу – 1600 х 900 мм (рисунок 2).

Розташування екрана ВДТ має забезпечувати зручність зорового спостереження у вертикальній площині під кутом +30° до нормальної лінії погляду працюючого.

Для забезпечення захисту і досягнення нормованих рівнів комп’ютерних випромінювань необхідно застосовувати приекранні фільтри, локальні світлофільтри (засоби індивідуального захисту очей) та інші засоби захисту, що пройшли випробування в акредитованих лабораторіях і мають щорічний гігієнічний сертифікат.

При оснащенні робочого місця з ВДТ лазерним принтером параметри лазерного випромінювання повинні відповідати вимогам ДСанПіН 3.3.2.007-98

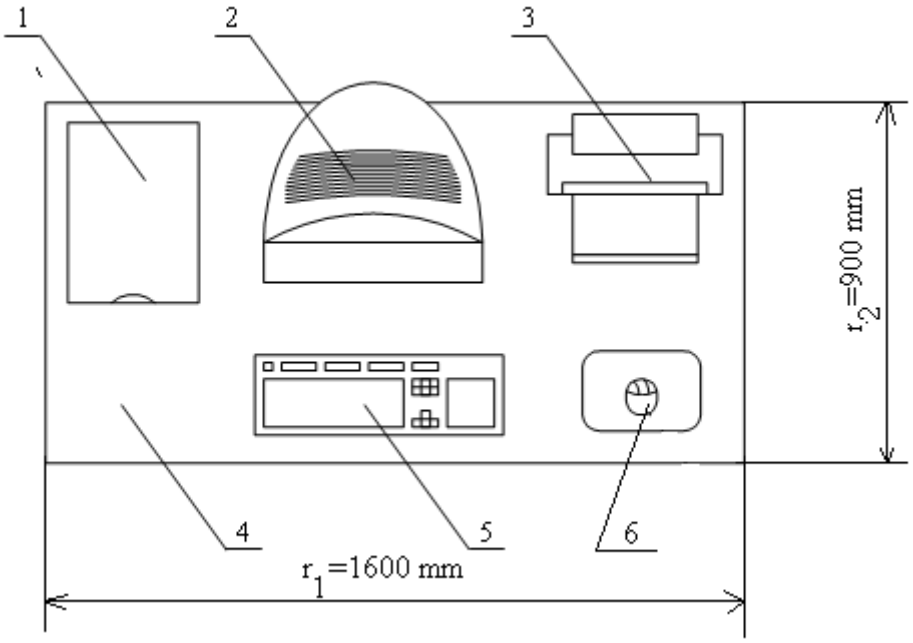


Рисунок 3.2 - Розміри робочого столу та розміщення устаткування на ньому
1 – сканер, 2 – монітор, 3 – принтер, 4 – поверхня робочого столу
5 – клавіатура, 6 – маніпулятор типу «миша».

Елементи робочого місця рекомендується розміщати на рівній відстані від очей користувача до екрана, клавіатури, підставки для документів. Конструкція столу повинна забезпечувати можливість оптимального розміщення на робочій поверхні використовуваних приладів, з урахуванням їх кількості, розмірів, конструктивних особливостей і характеру виконуваної роботи. Доцільно мати модульне робоче місце, яке легко можна змінювати.

3.9. Пожежна безпека

Пожежна безпека приміщень, що мають електричні мережі, регламентується ГОСТ 12.1.033-81, ГОСТ 12.1.004-85. Робота оператора ЕОМ повинна вестися в приміщенні, що відповідає категорії Д пожежної безпеки (негорючі речовини й матеріали в холодному стані).

Пожежна безпека забезпечується:

- системою запобігання пожежі;

- системою протипожежного захисту;

- організаційно-технічними заходами.

Протипожежний захист приміщення забезпечується застосуванням установки автоматичної пожежної сигналізації, наявністю засобів пожежогасіння, організацією своєчасної евакуації людей.

Для ліквідації невеликих осередків пожеж, а також для гасіння пожеж у початковій стадії їх розвитку силами персоналу об’єктів, застосовуються первинні засобі пожежогасіння. Це вогнегасники (вуглекислотні та порошкові), пожежний інвентар (покривала з негорючого полотна, ящики з піском, бочки з водою), пожежний інвентар.

ВИСНОВОК

В результаті виконання цього проекту всі цілі були досягнуті. Були виконані такі роботи:

Вибрано найбільш підходящі технології та бібліотеки для розробки

- Розроблено архітектуру програми для оптимальної взаємодії клієнтської частини з серверною
- Створено макет клієнтської частини, віртуальний роутинг та всі необхідні компоненти для клієнтської частини
- Розроблено модель авторизації та аутентифікації користувача з використанням JWT-токену
- Створено необхідні реляційні таблиці у базі даних для всіх сутностей у додатку
- Розроблено логіку роботи серверної частини для взаємодії з базою даних
- Створено маніфест та service worker для оптимальної роботи PWA

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ekaterinaryabukha. WebStorm 2020.3: обновленный интерфейс, поддержка Tailwind CSS и другие улучшения [Електронний ресурс] / ekaterinaryabukha // JetBrains. – 01.12.2022 – Режим доступу до ресурсу: https://habr.com/ru/company/JetBrains/blog/529708/.
2. Max R. Как структурировать крупномасштабное приложение Vue.js [Електронний ресурс] / Rokatansky Max // OTUS. – 04.08.2021 – Режим доступу до ресурсу: https://habr.com/ru/company/otus/blog/571272/.
3. Вступление во Vue [Електронний ресурс] // Vue documentation. – 07.06.2021 – Режим доступу до ресурсу: https://vuejs.org/guide/introduction.html.
4. Документация TypeORM [Електронний ресурс] – Режим доступу до ресурсу: https://typeorm.io/.
5. TypeScript in Visual Studio Code [Електронний ресурс]. – 609. – Режим доступу до ресурсу: https://code.visualstudio.com/docs/languages/typescript.
6. Parsons N. In Depth Guide on Building a REST API with Node.js, Restify & MongoDB [Електронний ресурс] / Nick Parsons // Medium. – 1612. – Режим доступу до ресурсу: https://medium.com/@nparsons08/in-depth-guide-on-building-a-rest-api-with-node-js-restify-mongodb-a8e92efbb50f.