

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему РОЗРОБКА ВЕБЗАСТОСУНКУ

ДЛЯ СОЦІАЛЬНОЇ СЛУЖБИ

DEVELOPMENT OF A WEB APPLICATION

FOR SOCIAL SERVICE

Виконав(ла): студент(ка) 4 курсу, групи КНТ-131

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ШАХОВ І.В.

(ПРИЗВИЩЕ та ініціали)

Керівник ГЛАДКОВА О.М.

(ПРИЗВИЩЕ та ініціали)

Рецензент ДЬЯЧУК Т.С.

(ПРИЗВИЩЕ та ініціали)

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

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

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2025 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ШАХОВА Івана Васильовича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка вебзастосунку для соціальної служби.
Development of a Web Application for Social Service

керівник проєкту (роботи) к.т.н., ГЛАДКОВА Ольга Миколаївна,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 28 ” квітня 2025 року № 209

2. Строк подання студентом проєкту (роботи) 02 червня 2025 року

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

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Основні рішення щодо реалізації компонентів системи. 4. Експлуатація, тестування та експериментальне дослідження програми.

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

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ГЛАДКОВА О.М., доцент		
Нормоконтроль	БЄЛОВА А.В., асистент		

7. Дата видачі завдання «14» квітня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

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

Студент(ка)

(підпис) Іван ШАХОВ
(Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Ольга ГЛАДКОВА
(Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
82 с., 9 табл., 41 рис., 3 дод., 16 джерел.

СОЦІАЛЬНА СЛУЖБА, ВЕБЗАСТОСУНОК, ОБЛІК, ПІЛЬГОВІ
КАТЕГОРІЇ.

Об'єкт дослідження – процес обліку осіб пільгових категорій у соціальних
службах .

Предмет дослідження – програмне забезпечення для соціальної служби.

Мета роботи – створення програмного забезпечення у вигляді
вебзастосунку для соціальної служби.

Матеріали, методи та технічні засоби: мова програмування TypeScript,
бібліотека React, Supabase, система керування базами даних PostgreSQL.

Результати. Розроблено програмне забезпечення у вигляді вебзастосунку
для соціальної служби.

Практична цінність полягає у розробці програмного забезпечення для
соціальної служби.

Висновки. Розроблено програмне забезпечення у вигляді вебзастосунку для
соціальної служби.

Галузь використання – соціальні служби, працівники.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 82 pages, 9 tables, 41 figures, 3 appendixes, 16 sources.

SOCIAL SERVICE, WEB APPLICATION, ACCOUNTING, BENEFIT CATEGORIES.

The object of the study is the process of accounting for persons of privileged categories in social services.

The subject of the study is software for social services.

The purpose of the work is to create software in the form of a web application for social services.

Materials, methods and technical means: TypeScript programming language, React library, Supabase, PostgreSQL database management system.

Results. Software was developed in the form of a web application for social service.

The practical value lies in the development of software for social service.

Conclusions. Software has been developed in the form of a web application for social services.

Field of use – social services, employees

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ.....	9
1 Аналіз предметної області.....	10
1.1 Предметна область та основні поняття.....	10
1.2 Актуальність роботи	11
1.3 Постановка завдання	12
1.4 Висновки за розділом 1	13
2 Матеріали і методи.....	14
2.1 Вибір мов програмування для розробки програмного забезпечення	14
2.2 Обґрунтування вибору програмної надбудови на стороні клієнта.....	15
2.3 Обґрунтування вибору програмної надбудови на стороні сервера	17
2.4 Вибір застосування рішень для виконання серверних запитів	18
2.5 Обґрунтування вибору бази даних	20
2.6 Вибір середовища розробки	21
2.7 Допоміжні засоби у розробці проєкту	22
2.8 Висновки за розділом 2	22
3 Основні рішення щодо реалізації компонентів системи.....	24
3.1 Алгоритм функціонування розробленої програми	24
3.1.1 Алгоритм роботи сторінки авторизації.....	24
3.1.2 Алгоритм роботи домашньої сторінки	25
3.1.3 Алгоритм роботи сторінки пільговика	26
3.2 Структура бази даних	28
3.3 Структура застосунку	30
3.4 Проєктування дизайну	31
3.5 Програмна реалізація	33
3.6 Висновки за розділом 3	41

4 Експлуатація, тестування та експериментальне дослідження програми	42
4.1 Опис застосування програми	42
4.2 Характеристики програми.....	43
4.3 Інструкція по експлуатації програми	43
4.4 Тестування та результати роботи програмного забезпечення	43
4.5 Висновки за розділом 4	46
Висновки	48
Перелік джерел посилання	49
Додаток А Технічне завдання	52
Додаток Б Текст програми	56
Додаток В Слайди презентації.....	75

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

- API –Application Programming Interface;
DOM – Document Object Model;
ОС – Операційна система;
JS – Java Script;
СКБД – Система керування базами даних.

ВСТУП

У сучасних умовах цифровізації важливою складовою ефективної роботи соціальних служб є впровадження зручних і доступних онлайн- інструментів.

Вебзастосунок для соціальної служби створений для оптимізації процесів надання соціальної допомоги, спрощення взаємодії між працівниками та громадянами, а також для забезпечення прозорості та автоматизації обліку даних.

Завдяки використанню інтернету працівники служби мають можливість оперативно обробляти запити, вести облік наданих послуг і контролювати перебіг соціальних програм. Це не лише підвищує якість обслуговування населення, а й дозволяє ефективно використовувати ресурси установи.

Через це було вирішено дослідити можливість створення вебзастосунку для соціальної служби.

Основний функціонал, який буде виконувати програма:

- надавати працівнику можливість заходити до свого профілю;
- у таблиці дивитись на списки пільговиків з короткою інформацією про них;
- обрати конкретного пільговика та перейти на його сторінку з повною інформацією;
- редагувати інформацію пільговиків, видаляти та додавати нових.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Предметна область та основні поняття

Національна соціальна сервісна служба України (Нацсоцслужба) — центральний орган виконавчої влади, діяльність якого спрямовується та координується Кабінетом Міністрів України через Міністра соціальної політики і який реалізує державну політику у сфері соціального захисту населення, здійснення державного нагляду (контролю) за дотриманням вимог законодавства під час надання соціальної підтримки, соціальних виплат та за дотриманням прав дітей [1].

Пільга — встановлена законодавством перевага, що надається визначеним категоріям осіб [2].

Категорії пільгових осіб:

- малозабезпечена сім'я - сім'я, члени якої постійно проживають в Україні і зареєстровані за однією адресою (щоб зрозуміти чи є сім'я малозабезпеченою, порівнюється середній дохід всіх членів сім'ї за 6 місяців, що передують зверненню, з прожитковим мінімумом для відповідних категорій, до яких відносяться члени сім'ї, до розрахунку не включається допомога на дітей, над якими встановлена опіка чи піклування) [3];
- багатодітні сім'ї - це сім'ї, в яких подружжя (чоловік та жінка) перебуває у зареєстрованому шлюбі, разом проживає та виховує трьох і більше дітей (також багатодітною сім'єю вважається сім'я, в якій є тільки один з подружжя, який самостійно виховує трьох і більше дітей) [3];
- учасники бойових дій - особи, які брали участь у виконанні бойових завдань по захисту Батьківщини у складі військових підрозділів, з'єднань, об'єднань всіх видів і родів військ Збройних Сил діючої армії (флоту), у партизанських загонах і підпіллі та інших формуваннях як у воєнний, так і у мирний час [3];

- внутрішньо переміщена особа - громадянин України, іноземець або особа без громадянства, яка перебуває на території України на законних підставах та має право на постійне проживання в Україні, яку змусили залишити або покинути своє місце проживання у результаті або з метою уникнення негативних наслідків збройного конфлікту, тимчасової окупації, повсюдних проявів насильства, порушень прав людини та надзвичайних ситуацій природного чи техногенного характеру [4];
- людина з обмеженими можливостями (інвалід) - особа зі стійким розладом функцій організму, зумовленим захворюванням, наслідком травм або з уродженими дефектами, що призводить до обмеження життєдіяльності, до необхідності в соціальній допомозі і захисті [5];
- пенсіонер - особа, що отримує пенсію — регулярну грошову виплату, що виплачується особам, які працювали до пенсійного віку, стали інвалідами або втратили годувальника [6].

1.2 Актуальність роботи

Основною метою дипломної роботи є створення вебзастосунку, який буде допомагати працівникам соціальних служб вести облік людей пільгових категорій.

На сьогодні в місті Запоріжжя дуже багато людей пільгових категорій, а саме: внутрішньо переміщені особи, учасники бойових дій, багатодітні і малозабезпечені сім'ї, і саме тому створення застосунку для їхнього обліку є дуже затребуваним. Це дозволяє соціальним службам ефективно організовувати та оновлювати інформацію про отримувачів пільг - більше не потрібно вести паперові журнали чи таблиці. Достатньо мати доступ до Інтернету, щоб швидко переглядати, редагувати та аналізувати дані про осіб з пільговими категоріями у зручному електронному вигляді.

1.3 Постановка завдання

Після проведення аналізу предметної області та актуальності, для вирішення проблеми, сформулюємо постановку задачі.

Основною метою розробки вебзастосунку є створення електронного кабінету для працівників соціальної служби та адміністративної панелі для отримання зведеної статистики. Система повинна відповідати певним вимогам.

До ключових функціональних вимог до системи будемо відносити наступні вимоги:

- автентифікація в системі;
- перегляд таблиці з особами пільгових категорій;
- виведення даних за певними критеріями.

У разі, якщо користувач не пройшов процедуру реєстрації та не авторизувався у системі, йому автоматично обмежується доступ до функціональних можливостей платформи, зокрема він не має права переглядати таблицю даних, а також не може отримати жодної інформації, що стосується осіб, які належать до пільгових категорій населення.

Авторизований користувач повинен мати наступні можливості:

- авторизація у системі;
- ведення електронного реєстру пільговиків у форматі зручної таблиці;
- можливість перегляду детальної інформації про пільговика при виборі відповідного запису;
- редагування та оновлення даних;
- пошук, фільтрація та сортування записів.

Вимоги до інтерфейсу вебзастосунку:

- простота та зрозумілість;
- кольори в одному стилі, без яскравих відтінків;
- гнучкість інтерфейсу.

1.4 Висновки за розділом 1

У першому розділі було розглянуто предметну область, визначені основні поняття щодо обліку пільговиків, а також обґрунтовано актуальність розробки вебзастосунку для соціальної служби.

Було зазначено постановку задачі, сформульовані функціональні вимоги та вимоги до інтерфейсу.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір мов програмування для розробки програмного забезпечення

Для розробки застосунку було обрано використовувати мову програмування TypeScript.

TypeScript - це строго типізована об'єктно-орієнтована мова програмування, яка є надмножиною JavaScript. Вона розширює можливості JavaScript за рахунок введення статичної типізації, інтерфейсів, дженериків, модифікаторів доступу та інших сучасних концепцій, які дозволяють писати більш надійний і масштабований код.

TypeScript є потужною надбудовою над JavaScript, яка може ефективно використовуватись для створення широкого спектра програмних рішень. Зокрема, його застосовують при розробці клієнтських (frontend) і серверних (backend) застосунків, повноцінних веб-додатків, мобільних застосунків, а також для написання скриптів і утиліт, які зазвичай реалізуються за допомогою JavaScript. Завдяки тому, що TypeScript компілюється у стандартний JavaScript, результуючий код може бути виконаний у будь-якому середовищі, яке підтримує JavaScript, включаючи сучасні веб-браузери, серверну платформу Node.js, а також інші платформи, що базуються на JavaScript-рушіях.

Основною перевагою є те, що завдяки статичній типізації, помилки в коді виявляються ще на етапі розробки, до запуску програми. Це значно полегшує відлагодження, прискорює розробку і робить код більш зрозумілим та надійним.

TypeScript має високу популярність завдяки широкій підтримці спільноти, великій кількості навчальних матеріалів та активному розвитку. Також він повністю сумісний з JavaScript, що дозволяє поступово впроваджувати TypeScript у вже існуючі JavaScript-проекти [7].

У табл. 2.1 буде наведено порівняльну характеристику мов програмування JavaScript та TypeScript

Таблиця 2.1 – Порівняльна таблиця мов програмування

Особливість	TypeScript	JavaScript
Статична типізація	+	-
Динамічна типізація	-	+
Інтерфейси	+	-
Виявлення помилок на етапі компіляції	+	-
Підтримка сучасних стандартів ECMAScript	+	+
Підтримка у великих проєктах	+	+
Абстрактні класи	+	-

2.2 Обґрунтування вибору програмної надбудови на стороні клієнта

Для розробки клієнтської частини застосунку розглядалися два варіанти: бібліотека React JS та фреймворк Vue.

Бібліотека React JS була обрана для створення користувацького інтерфейсу застосунку. Вона призначена для ефективного рендерингу візуальної частини програми та побудована на основі компонентного підходу [8].

React використовує розширення синтаксису JavaScript - JSX, це дозволяє поєднувати HTML-подібний синтаксис із JavaScript-кодом. Після компіляції JSX трансформується у звичайні виклики JavaScript-функцій і перетворюється на об'єкти JavaScript. Це дає можливість використовувати конструкції на кшталт if, for, а також передавати вирази як аргументи та повертати їх із функцій [8].

Основні концепції React [8]:

- компонентний підхід - інтерфейс розділюється на ізольовані, незалежні частини, які називаються компонентами;
- віртуальний DOM - оптимізована копія реального DOM, яка дозволяє оновлювати тільки ті елементи інтерфейсу, які змінилися, що підвищує продуктивність;

– життєвий цикл компонентів - включає етапи ініціалізації, оновлення стану та демонтажу компонента.

Використання віртуального DOM забезпечує ефективне повторне відображення компонентів, оскільки оновлюються лише ті частини DOM-дерева, які зазнали змін [8].

Далі розглянуто Vue, це JavaScript-фреймворк, який також призначений для створення користувацьких інтерфейсів. Він побудований на шаблоні MVVM (Model-View-ViewModel) і застосовує концепцію реактивного програмування. Тобто, що дані автоматично синхронізуються між моделлю та відображенням [9].

У табл. 2.2 наведено порівняльну характеристику надбудов.

Таблиця 2.2 – Порівняльна таблиця надбудови на стороні клієнта

Особливість	React	Vue
Компонентна архітектура	+	+
Віртуальний DOM	+	+
Підтримка TypeScript	++	+
Гнучкість налаштувань архітектури	+	-
Легка масштабованість	+	-

Основні особливості Vue:

- компонентна архітектура - код розділяється на незалежні модулі, які можна повторно використовувати, що сприяє зменшенню дублювання коду;
- реактивність - забезпечує автоматичне оновлення інтерфейсу при зміні стану даних;
- шаблонний синтаксис - Vue дозволяє створювати шаблони за допомогою JavaScript, HTML або JSX [9].

Таким чином, обидві технології орієнтовані на створення компонентних інтерфейсів, однак React був обраний через його гнучкість, широке поширення та активну підтримку спільноти.

2.3 Обґрунтування вибору програмної надбудови на стороні сервера

Supabase — це потужна платформа для розробки серверної частини, яка забезпечує різноманітні серверні функції, такі як аутентифікація, робота з базою даних та зберігання файлів, що дозволяє зосередитись на розробці без необхідності вручну налаштовувати сервер. Supabase використовує PostgreSQL як базу даних і автоматично генерує API для взаємодії з нею, що значно спрощує роботу розробників. Оскільки ця платформа надає можливість масштабувати проєкти без необхідності глибоких знань у серверних технологіях, вона ідеально підходить для швидкого старту [10].

Supabase також має інтеграцію з такими популярними технологіями, як JavaScript, що дозволяє розробникам ефективно працювати як з серверною, так і з клієнтською частиною. Крім того, платформу підтримують основні операційні системи, включаючи Windows, Linux та MacOS, і вона має активну спільноту користувачів та розробників. Supabase також використовує подійно-орієнтовану модель обробки запитів, що дозволяє зменшити навантаження на сервер та покращити ефективність роботи з даними [10].

Основні переваги використання Supabase:

- підтримка PostgreSQL та автоматична генерація API;
- простота інтеграції з фронтендом за допомогою JavaScript;
- гнучкість у налаштуваннях та можливість масштабування проєктів;
- хмарна інфраструктура, що полегшує налаштування та адміністрування серверної частини;
- підтримка масштабованості за рахунок використання розподіленої бази даних [10].

Для створення серверної частини з використанням Supabase не потрібно налаштовувати сервер вручну, на відміну від використання фреймворків, таких як Express.js чи NestJS. Supabase забезпечує всі необхідні функції для роботи з базами даних, аутентифікації та іншими серверними операціями, значно спрощуючи розробку та підтримку додатків [10].

У таблиці 2.3 наведено характеристику надбудов на стороні сервера

Таблиця 2.3 – Порівняння характеристик Supabase та Node.js.

Характеристика	Supabase	Node.js
Масштабованість	+ Легко масштабується через хмарну інфраструктуру	+ Висока, але потребує налаштування серверу
Простота використання	+ Легко інтегрується, мінімальна настройка	- Потрібно більше налаштувань та знань для налаштування серверу
Підтримка бази даних	+ Вбудована підтримка PostgreSQL з автоматичним API	- Потрібно налаштовувати базу даних вручну
Аутентифікація	+ Вбудована підтримка аутентифікації та керування користувачами	- Потрібно налаштовувати аутентифікацію вручну
Технології	+ Інтеграція з JavaScript, автоматичні API	- Потрібно налаштовувати сервер та фронтенд окремо

2.4 Вибір застосування рішень для виконання серверних запитів

Для використання запитів на клієнтській стороні можуть використовуватись два популярні рішення - Axios і Fetch. Їхні основні характеристики наведені у таблиці 2.4.

Fetch - це вбудований у браузер API, який дозволяє виконувати HTTP-запити до серверів і обробляти їхні відповіді. Метод `fetch()` повертає Promise, яка виконується після отриманого результату [11].

Основні переваги Fetch:

- вбудований у всі сучасні браузери, не потребує встановлення;
- більша гнучкість у налаштуванні запитів (можливість контролю заголовків, методів, тіла запиту тощо);
- зручність при роботі з потоками;
- компактний синтаксис [11].

Далі розглянемо Axios.

Axios - це бібліотека JS, призначена для взаємодії із сервером через HTTP-запити. Вона підтримує `Api Promise`, що дозволяє ефективно обробляти асинхронні запити, а також забезпечує можливість перехоплення HTTP-запитів і відповідей [12].

Основні переваги Axios:

- висока сумісність з браузерами;
- зручне налаштування таймауту запиту;
- автоматичне перетворення відповідей у формат JSON [12].

Таблиця 2.4 – Порівняльна характеристика застосувань для запитів

Особливість	Fetch	Axios
Потрібна установка бібліотеки	-	+
Підтримка Promise	+	+
Вбудованість у браузер	+	-
Автоматичне перетворення даних у JSON	-	+
Робота з потоками	+	-

2.5 Обґрунтування вибору бази даних

Система керування базами даних (СКБД) - це комп'ютеризована платформа для зберігання та обробки даних. Вона надає користувачам інструменти для створення, збереження, оновлення і пошуку інформації.

Для проєкту було обрано PostgreSQL - потужну об'єктно реляційну СКБД з відкритим вихідним кодом. Вона відома своєю надійністю, масштабованістю і високою розширюваністю. PostgreSQL підтримує складні SQL-запити, транзакції з ACID-властивостями, реплікацію даних, роботу з JSON, геопросторові функції та повнотекстовий пошук [13].

Ця СКБД легко масштабується - її можна використовувати як для невеликих вебзастосунків, так і для великих корпоративних систем із великими обсягами даних. PostgreSQL активно розвивається за підтримки широкої спільноти, що гарантує регулярні оновлення, виправлення помилок і впровадження нових можливостей. Завдяки підтримці Linux, Windows, та macOS, вона є універсальним рішенням для різних середовищ розробки. Гнучкі налаштування та підтримка сторонніх розширень дозволяють адаптувати систему до конкретних вимог проєкту [13].

У таблиці 2.5 наведено порівняльну характеристику баз даних.

Таблиця 2.5 – Порівняння PostgreSQL з його аналогом MySQL

Характеристика	PostgreSQL	MySQL
Відкритий вихідний код	+	+
Підтримка ACID транзакцій	+	+
Розширені SQL-запити	+	-
Повнотекстовий пошук	+	-
Робота з JSON	+	+

Основні переваги PostgreSQL:

- висока стабільність і надійність;
- підтримка ACID і розширений SQL;
- активна спільнота і відкритий вихідний код;
- крос-платформеність [13].

Хоча існує багато реляційних і нереляційних баз даних, PostgreSQL вирізняється стабільністю, масштабованістю, відповідністю міжнародним стандартам і можливістю розширення функціональності. Завдяки цим перевагам вона є відмінним вибором для створення складних і високонавантажених вебзастосунків, які потребують надійного та гнучкого управління даними.

2.6 Вибір середовища розробки

Сьогодні існує чимало сучасних редакторів коду, що задовольняють різні потреби розробників. Нижче наведено порівняння і короткий опис двох популярних середовищ - Visual Studio Code (VS Code) та PhpStorm.

У таблиці 2.6 наведено порівняльну характеристику редакторів коду VS Code та PhpStorm.

Таблиця 2.6 – Порівняльна характеристика редакторів коду

Функціональність	VS Code	PhpStorm
Відкритий вихідний код	+	-
Інтеграція з Git	+	+
Підтримка Node.js	+	+
Зовнішній вигляд	+	-
Легкість програми	+	-

Visual Studio Code (VS Code) — це сучасне, потужне та багатоцільове середовище розробки, яке широко використовується програмістами для написання, редагування та налагодження вихідного коду. Завдяки підтримці великої кількості мов програмування, великої кількості розширень, гнучке налаштування інтерфейсу, інтеграцію з системами контролю версій Git і підтримує багато мов програмування [14].

PhpStorm - це інтегроване середовище розробки, оптимізоване для PHP і JS. Воно підтримує запуск і налагодження застосунків на Node.js, редагування як серверного, так і клієнтського коду, автоматичне збереження файлів, а також сучасні можливості роботи з CSS і системами стилізації [15].

Для виконання завдань у цьому проєкті було обрано Visual Studio Code завдяки його універсальності, високій продуктивності та великій спільноті користувачів. VS Code забезпечує комфортне середовище для написання, налагодження та тестування коду, що дозволяє ефективно розробляти як фронтенд, так і бекенд частини застосунку.

2.7 Допоміжні засоби у розробці проєкту

У процесі розробки проєкту було використано GitHub - онлайн платформу для спільної роботи над програмним забезпеченням. Вона дозволяє зберігати, відстежувати зміни та керувати проєктами, а також легко співпрацювати з іншими розробниками. За допомогою GitHub можна завантажувати код, відновлювати попередні версії та повертатися до стабільних станів у разі помилок або збоїв [16].

2.8 Висновки за розділом 2

У другому розділі пояснювальної записки було обґрунтовано вибір технологій для реалізації проєкту. Основною мовою програмування обрано TypeScript.

Для клієнтської частини вирішено використовувати ReactJS завдяки її легкій масштабованості, активній спільноті, яка надає готові рішення, відеоуроки та навчальні матеріали. Також було розглянуто фреймворк Vue, однак перевагу надано саме ReactJS.

Для створення бази даних вебзастосунку було обрано PostgreSQL завдяки її можливості розширення, активній спільноті, гнучким параметрам налаштування та підтримці розподілених систем.

Серверна частина застосунку реалізована за допомогою Supabase, що дозволило спростити роботу з базою даних та аутентифікацією завдяки вбудованим сервісам, таким як автоматичне генерування API для роботи з PostgreSQL та вбудована підтримка аутентифікації користувачів.

Для взаємодії між клієнтом і сервером обрано вбудовану функцію Fetch API. Вона була вибрана завдяки своїй простоті, стандартній підтримці у сучасних браузерах, відсутності потреби встановлювати додаткові бібліотеки.

Середовищем розробки було обрано VS Code. Серед допоміжних інструментів у розробці застосовувалась платформа GitHub.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Алгоритм функціонування розробленої програми

3.1.1 Алгоритм роботи сторінки авторизації

Розглянемо алгоритм роботи сторінки авторизації. Після відкриття сторінки авторизації, користувач бачить форму для введення логіну та паролю. Користувач вводить логін та пароль. Якщо дані вірні - користувач переходить на наступну сторінку, якщо ні - то з'являється помилка.

Схема алгоритму роботи сторінки авторизації наведено на рисунку 3.1.



Рисунок 3.1 – Схема алгоритму роботи сторінки авторизації

3.1.2 Алгоритм роботи домашньої сторінки

Після авторизації користувача у системі, він переходить на домашню сторінку застосунку. На цій сторінці є таблиця з усіма особами пільгових категорій, які є у базі даних і деякою інформацією про них. У шапці таблиці є кнопка “Створити”. При натисканні на неї відкривається форма з додаванням нової особи пільгових категорій. Також у шапці таблиці є поля з фільтрами по районах проживання, по назві категорії та пошук за ПІБ. При натисканні на рядок у таблиці з даними особи пільгових категорій відкривається його форма. У шапці головної сторінки також є кнопка виходу. При натисканні на неї користувач виходить із системи та повертається на сторінку авторизації.

Схема алгоритму роботи домашньої сторінки наведено на рисунку 3.2.

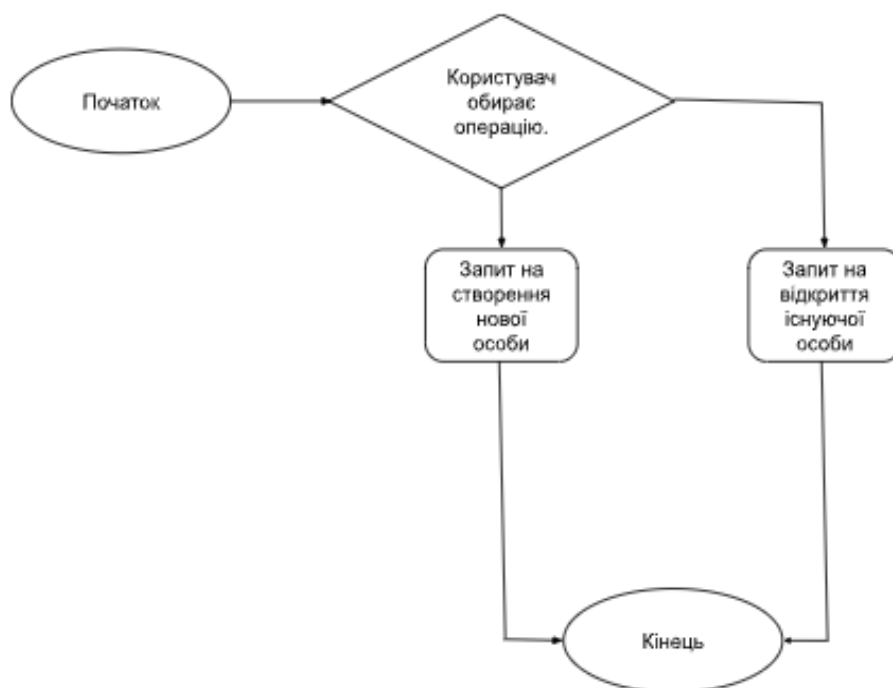


Рисунок 3.2 – Схема алгоритму роботи домашньої сторінки

3.1.3 Алгоритм роботи сторінки пільговика

Сторінка пільговика відкривається у двох випадках:

- при натисканні кнопки “Створити” на головній сторінці;
- при натисканні на рядок у таблиці з даними особи пільгових категорій.

При натисканні на кнопку “Створити” відкривається сторінка пільговика з пустою формою, яку можна заповнити даними та зберегти до бази даних. Також є можливість закрити форму та сторінку пільговика, натиснувши кнопку “Закрити”. Тоді користувач повертається на головну сторінку, а дані не зберігаються.

Схема алгоритму роботи сторінки пільговика при створенні нової особи пільгових категорій наведено на рисунку 3.3.

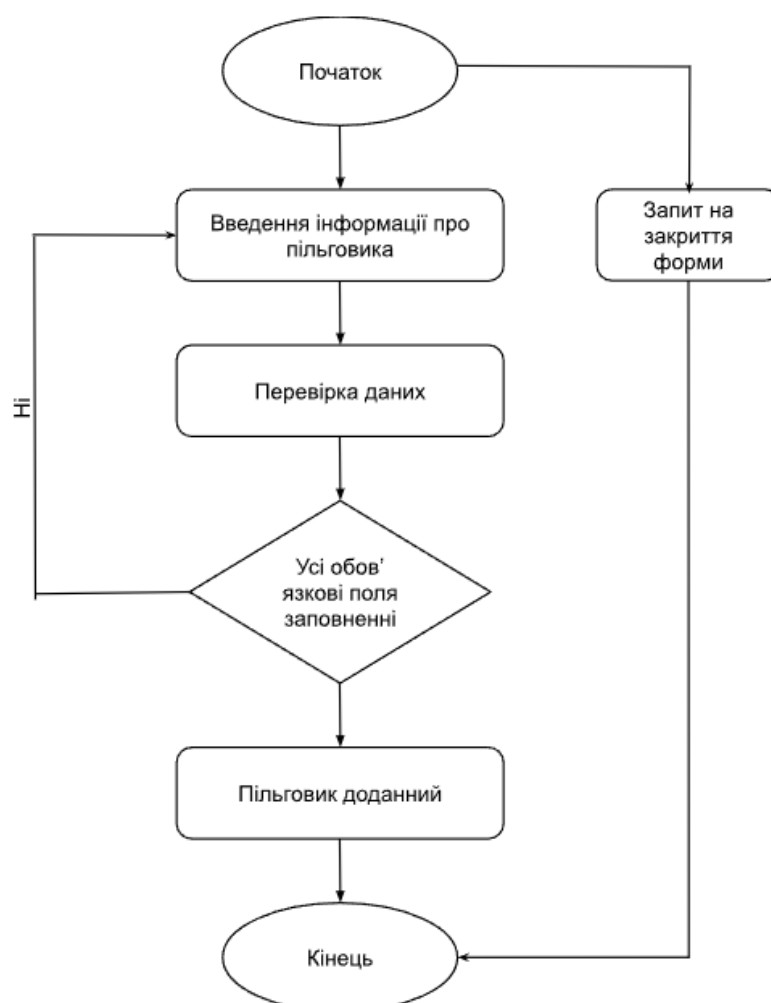


Рисунок 3.3 – Схема алгоритму роботи сторінки пільговика

При натисканні на рядок у таблиці з даними особи пільгових категорій відкривається сторінка пільговика. На цій сторінці є форма з усіма даними особи пільгових категорій. Цю інформацію можна редагувати та потім зберегти зміни, натиснувши на кнопку “Зберегти”. Є можливість видалити пільговика з бази даних, натиснувши на кнопку “Видалити”. Також є можливість закрити форму та сторінку пільговика, натиснувши кнопку “Закрити”. Тоді користувач повертається на головну сторінку, а змінені дані не зберігаються.

Схема алгоритму роботи сторінки пільговика при редагуванні існуючої особи пільгових категорій наведено на рисунку 3.4.

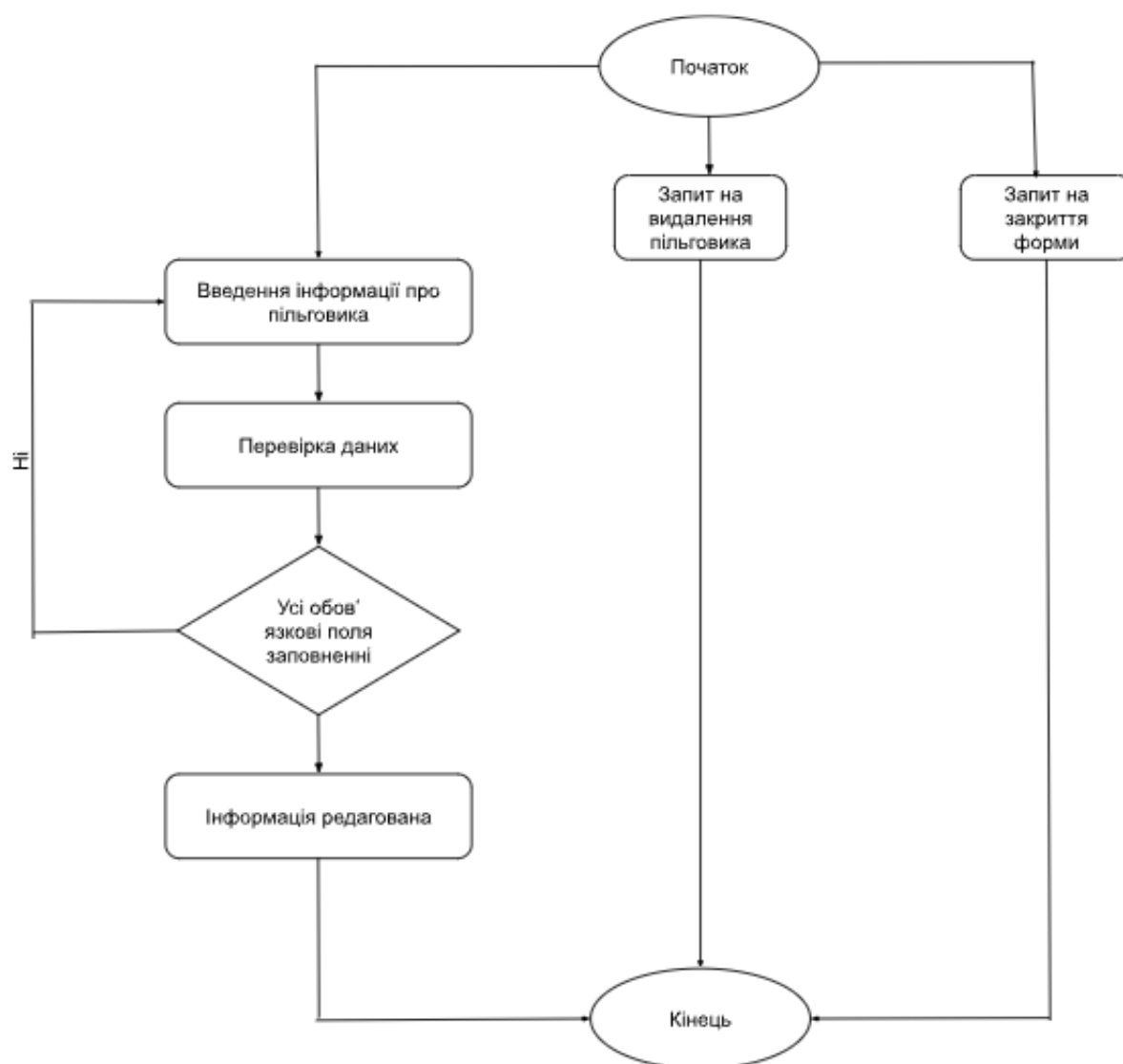


Рисунок 3.4 – Схема алгоритму роботи сторінки пільговика

3.2 Структура бази даних

На рисунку 3.5 зображена структура бази даних для обліку пільговиків (beneficiaries) та пов'язаних з ними даних. Нижче — пояснення по кожній таблиці.

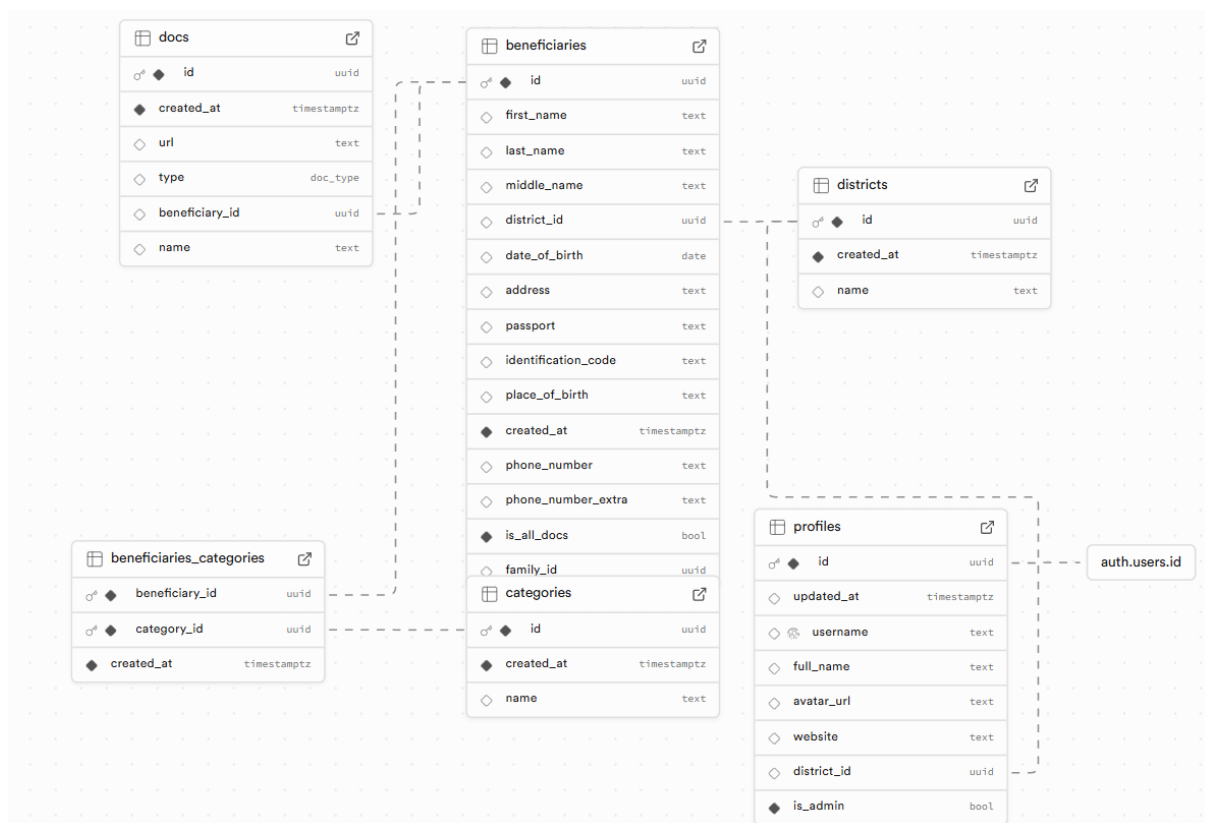


Рисунок 3.5 – Схема структури бази даних

Таблиця beneficiaries — головна таблиця з інформацією про пільговиків:

- id - унікальний ідентифікатор пільговика (UUID);
- first_name, last_name, middle_name - ПІБ;
- date_of_birth, place_of_birth - дата і місце народження;
- address - адреса;
- passport, identification_code - окремо номер паспорта та ІПН;
- district_id - посилання на район проживання (з таблиці districts);
- phone_number, phone_number_extra – телефони;
- is_all_docs - логічне поле (true/false), чи надано всі потрібні документи;

- `family_id` - використовується для групування членів сім'ї.

Таблиця `docs` - документи, пов'язані з пільговиками:

- `id` - ідентифікатор документа;
- `beneficiary_id` - кому належить документ (зв'язок з `beneficiaries`);
- `type` - тип документа (наприклад, `passport`, `identification_code`, або назва категорії);
- `url` - посилання на файл;
- `name` - ім'я файлу;
- `created_at` - дата додавання.

Таблиця `categories` - категорії пільговиків (наприклад: УБД, ВПО, багатодітна сім'я тощо):

- `id, name` - ідентифікатор та назва категорії;
- `created_at` - дата створення.

Таблиця `beneficiaries_categories` - зв'язок між пільговиками і категоріями:

- `beneficiary_id` - посилання на пільговика;
- `category_id` - посилання на категорію;
- `created_at` - дата додавання.

Таблиця `districts` – райони:

- `id, name` - ідентифікатор та назва району;
- `created_at` - дата створення.

Таблиця `profiles` - профілі користувачів:

- `id` - зв'язаний з `auth.users.id` (зовнішня автентифікація);
- `full_name, username, avatar_url, website` - інформація про користувача;
- `district_id` - з якого району користувач;
- `is_admin` - чи є користувач адміністратором.

Таблиця `auth.users` — зовнішня таблиця з Supabase.

Це система автентифікації, яка зберігає `uuid` користувача, використовується для зв'язку з `profiles`.

3.3 Структура застосунку

Вебзастосунок складається з двох частин: клієнтської та адміністраторської. Адміністратор має повний доступ до інформації — він може переглядати пільгові категорії осіб у всіх районах та додавати нових бенефіціарів незалежно від їхнього місця проживання.

Натомість звичайний працівник має обмеження: він може додавати та редагувати дані лише тих бенефіціарів, які зареєстровані в районі, що відповідає місцю розташування його соціальної установи.

Наприклад, якщо працівник працює в установі, розташованій у Шевченківському районі, то він зможе переглядати та вносити дані лише для мешканців цього району.

Структуру програмного забезпечення для соціальної служби у вигляді структурних діаграм в нотації UML, зокрема діаграми класів, пакетів, компонентів (рис. 3.6 - 3.8).

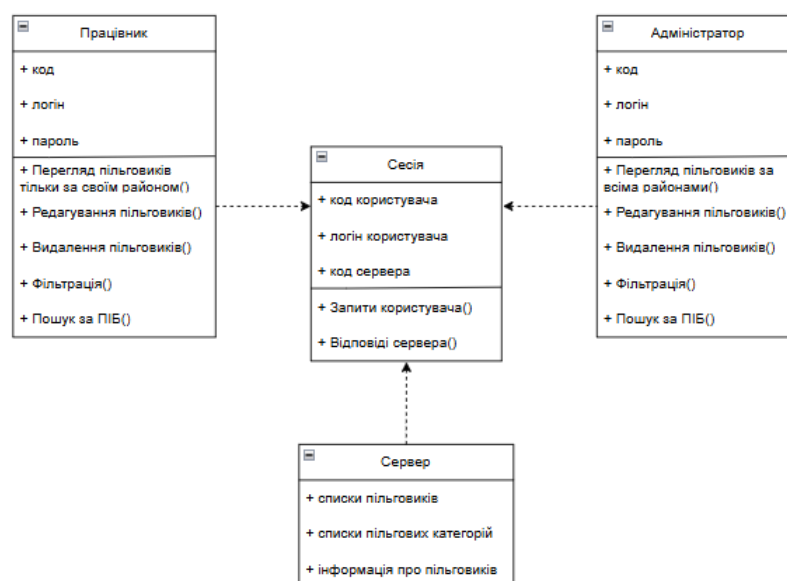


Рисунок 3.6 – Діаграма класів

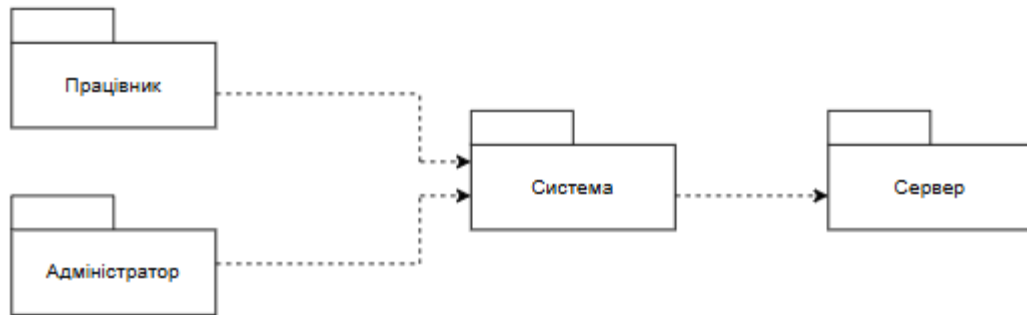


Рисунок 3.7 – Діаграма пакетів програмного забезпечення



Рисунок 3.8 – Діаграма компонентів програмного забезпечення

3.4 Проектування дизайну

Нижче на рисунках представлено приклади проектування дизайну вебзастосунку для соціальної служби.

На рисунку 3.9 зображено сторінку авторизації.

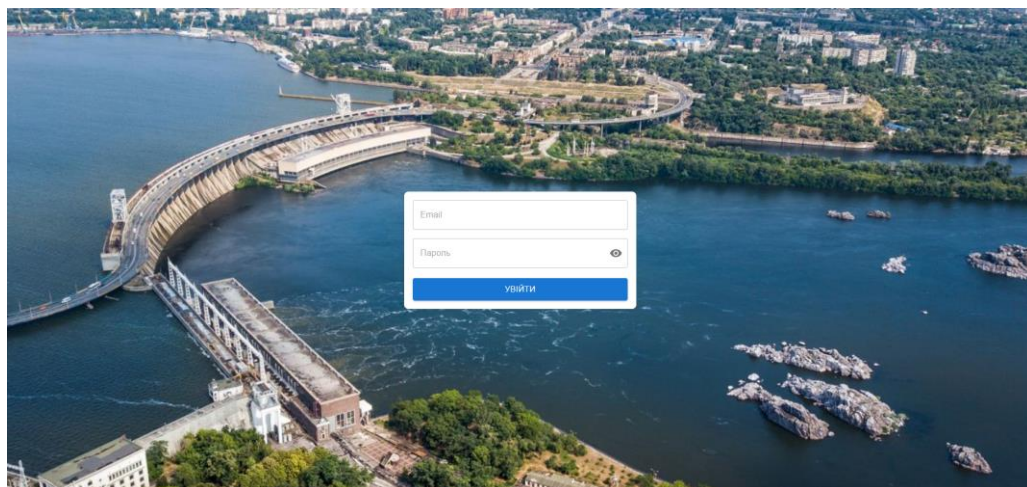
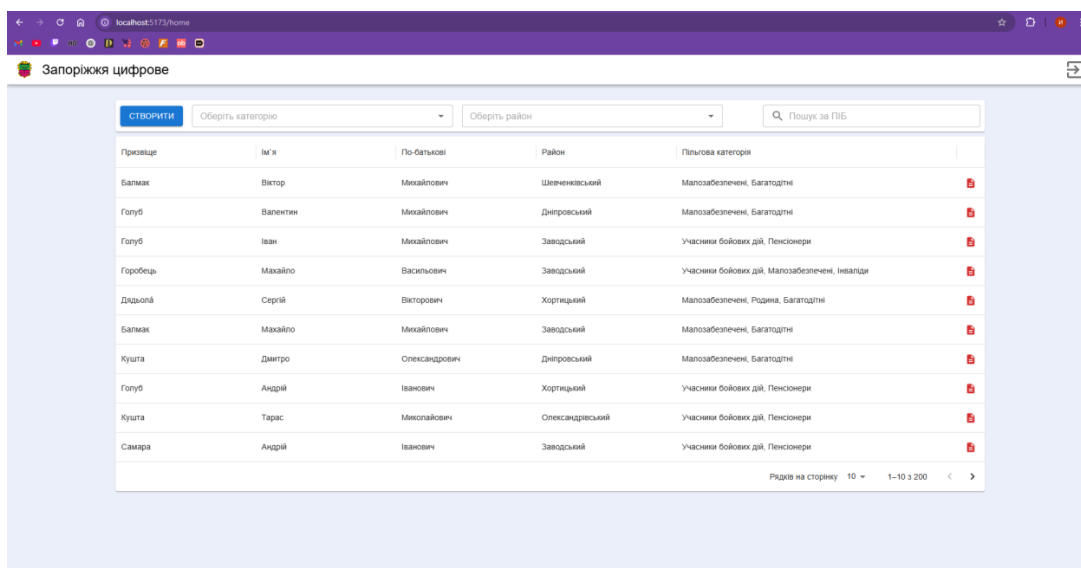


Рисунок 3.9 – Сторінка авторизації

На рисунку 3.10 зображено домашню сторінку.

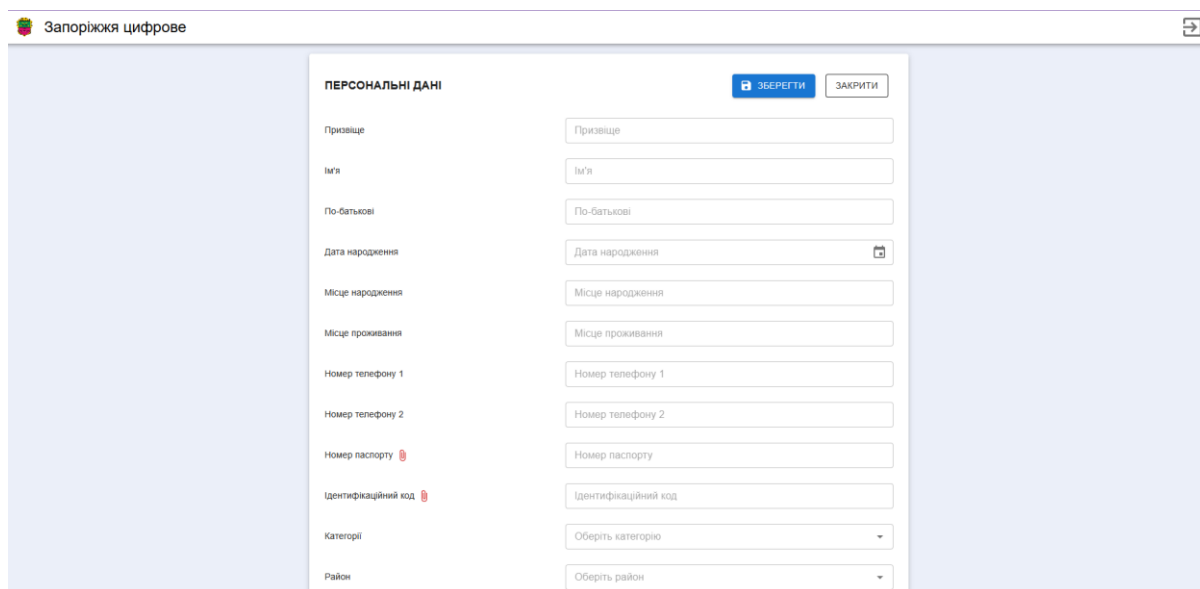


The screenshot shows a web application titled 'Запоріжжя цифрове'. At the top, there is a navigation bar with a 'СТВОРИТИ' button and two dropdown menus for 'Оберть категорію' and 'Оберть район'. A search bar labeled 'Пошук за ПІБ' is also present. Below this is a table with the following columns: Прізвище, Ім'я, По-батькові, Район, and Пільгова категорія. The table contains 10 rows of data. At the bottom right of the table, there is a pagination control showing 'Рядки на сторінку 10' and '1-10 з 200'.

Прізвище	Ім'я	По-батькові	Район	Пільгова категорія
Балмак	Віктор	Михайлович	Шевченківський	Малообезпечені, Багатодітні
Голуб	Валентин	Михайлович	Дніпровський	Малообезпечені, Багатодітні
Голуб	Іван	Михайлович	Заводський	Учасники бойових дій, Пенсіонери
Горобець	Михайло	Васильович	Заводський	Учасники бойових дій, Малообезпечені, Інваліди
Дядюла	Сергій	Вікторович	Хортицький	Малообезпечені, Родина, Багатодітні
Балмак	Михайло	Михайлович	Заводський	Малообезпечені, Багатодітні
Кушта	Дмитро	Олександрович	Дніпровський	Малообезпечені, Багатодітні
Голуб	Андрій	Іванович	Хортицький	Учасники бойових дій, Пенсіонери
Кушта	Тарас	Миколайович	Олександрівський	Учасники бойових дій, Пенсіонери
Самара	Андрій	Іванович	Заводський	Учасники бойових дій, Пенсіонери

Рисунок 3.10 – Домашня сторінка

На рисунку 3.11 зображено сторінку створення нового бенефіціара.



The screenshot shows a web application titled 'Запоріжжя цифрове'. The main content area is a form titled 'ПЕРСОНАЛЬНІ ДАНІ'. At the top right of the form are two buttons: 'ЗБЕРЕГТИ' and 'ЗАКРИТИ'. The form contains the following fields: Прізвище, Ім'я, По-батькові, Дата народження (with a calendar icon), Місце народження, Місце проживання, Номер телефону 1, Номер телефону 2, Номер паспорту, Ідентифікаційний код, Категорія (dropdown menu), and Район (dropdown menu).

Рисунок 3.11 – Сторінка створення нового бенефіціара

На рисунку 3.12 зображено сторінку редагування бенефіціара.

Запоріжжя цифрове

ПЕРСОНАЛЬНІ ДАНІ

ВІДАЛИТИ ЗБЕРЕЖТИ ЗАКРИТИ

Прізвище: Тронь

Ім'я: Тарас

По-батькові: Васильович

Дата народження: 09.11.1979

Місце народження: м. Запоріжжя

Місце проживання: вул. Шкільна, 23

Номер телефону 1: Номер телефону 1

Номер телефону 2: Номер телефону 2

Номер паспорту: KB123456

Ідентифікаційний код: 1234567890

Категорія: Учасники бойових дій ВПО

Район: Комунальний

Рисунок 3.12 – Сторінка редагування бенефіціара

3.5 Програмна реалізація

Файлова структура проекту зазначена на рисунку 3.13.

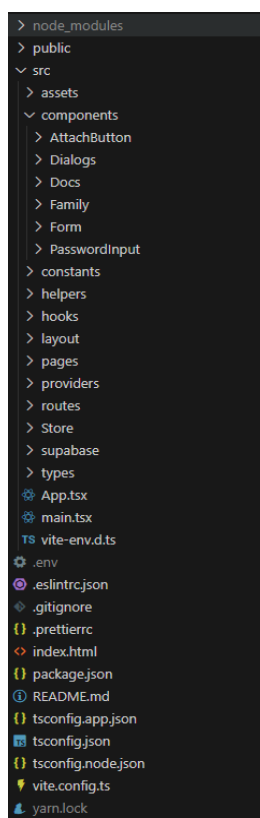


Рисунок 3.13 – Файлова структура проекту

Головним файлом проекту є `main.tsx`, оскільки саме в ньому відбувається:

- ініціалізація React-додатку через `createRoot(...)`;
- підключення основного компонента `App`;
- монтування React-дерева у DOM-елемент з `id="root"`.

Папка `node_modules` містить бібліотеки та залежності, необхідні для коректної роботи та запуску проекту.

Каталог `routes` відповідає за маршрутизацію, яка дозволяє серверу обробляти запити клієнта відповідно до вказаних шляхів.

Файл `package.json` містить основну інформацію про застосунок.

У папці `assets` зберігаються статичні ресурси, такі як зображення, що використовує вебзастосунок.

Каталог `components` (рис.3.14) містить багаторазові UI-компоненти, що повторно використовуються у застосунку.

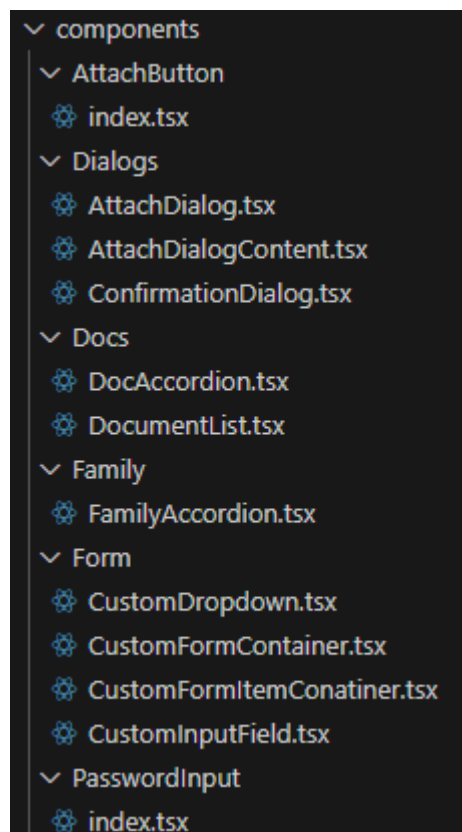


Рисунок 3.14 – Каталог `components`

Папка pages (рис.3.15) включає окремі сторінки застосунку, кожна з яких відповідає певному маршруту.

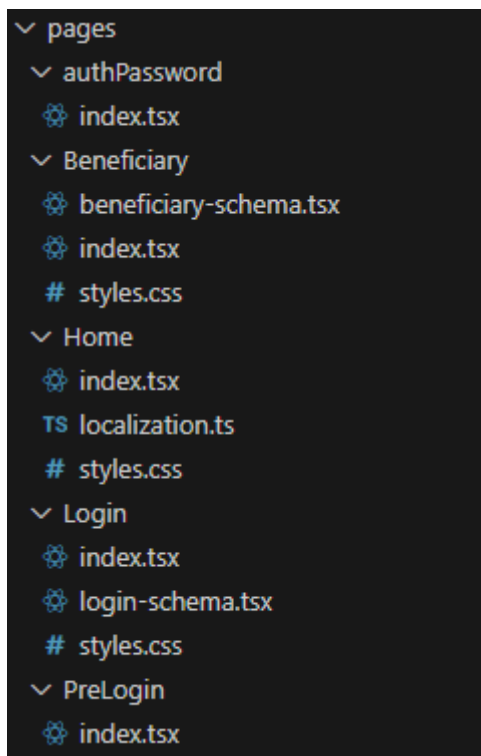


Рисунок 3.15 – Каталог pages

Після запуску проєкту користувач потрапляє на головну сторінку, якою є форма авторизації в системі (рис. 3.16). Щоб успішно авторизуватись, необхідно ввести дані, які зазначено в полях.

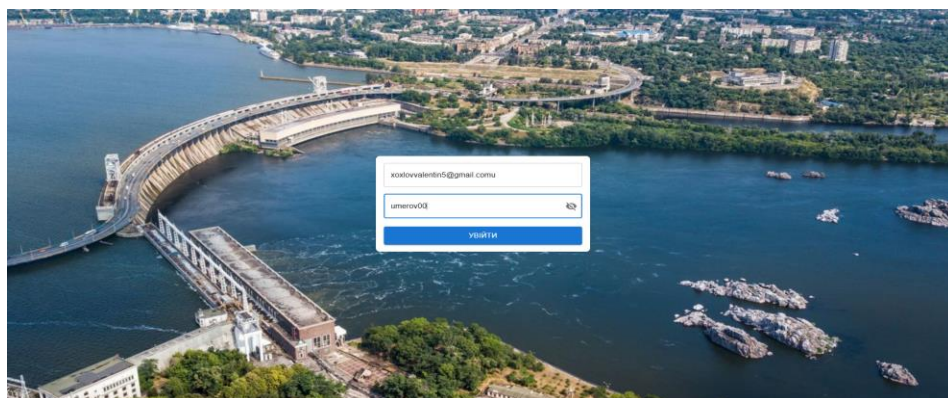


Рисунок 3.16 – Форма авторизації

Після успішної авторизації, користувач переходить на домашню сторінку (рис. 3.17). На цій сторінці користувач може подивитись коротку інформацію про осіб пільгових категорій:

- ПІБ;
- район проживання;
- пільгова категорія.

Також користувач може застосувати фільтри по районах та пільгових категоріям (рис. 3.18), а також виконати пошук за ПІБ (рис. 3.19). У шапці таблиці є кнопка “Створити”, при натисканні якої користувача переводить на сторінку створення нового бенефіціара (рис. 3.20). Якщо натиснути на рядок з вже існуючим бенефіціаром, відкривається сторінка редагування (рис. 3.23). У правій частині таблиці є значок документів, якщо не всі документи додані до пільговика , то значок горить червоним, якщо всі - то сірим.

У шапці сторінки є кнопка виходу. При натисканні на неї, користувач виходить з облікового запису та повертається на сторінку авторизації.

Прізвище	Ім'я	По-батькові	Район	Пільгова категорія
Балмак	Віктор	Михайлович	Шевченківський	Малозабезпечені, Багатодітні
Голуб	Валентин	Михайлович	Дніпровський	Малозабезпечені, Багатодітні
Голуб	Іван	Михайлович	Заводський	Учасники бойових дій, Пенсіонери
Горобець	Махайло	Васильович	Заводський	Учасники бойових дій, Малозабезпечені, Інваліди
Дядюла	Сергій	Вікторович	Хортицький	Малозабезпечені, Родина, Багатодітні
Балмак	Махайло	Михайлович	Заводський	Малозабезпечені, Багатодітні
Кушта	Дмитро	Олександрович	Дніпровський	Малозабезпечені, Багатодітні
Голуб	Андрій	Іванович	Хортицький	Учасники бойових дій, Пенсіонери
Кушта	Тарас	Миколайович	Олександрівський	Учасники бойових дій, Пенсіонери
Самара	Андрій	Іванович	Заводський	Учасники бойових дій, Пенсіонери

Рисунок 3.17 – Домашня сторінка

СТВОРИТИ

ВПО

Дніпровський

Пошук за ПІБ

Прізвище	Ім'я	По-батькові	Район	Пільгова категорія	
Матхей	Іван	Володимирович	Дніпровський	Учасники бойових дій, ВПО	
Щебетун	Махайло	Миколайович	Дніпровський	ВПО	
Жук	Іван	Володимирович	Дніпровський	ВПО, Багатодітні	
Балмак	Данило	Михайлович	Дніпровський	ВПО, Багатодітні	
Півень	Олександр	Вікторович	Дніпровський	ВПО, Багатодітні	
Горобець	Володимир	Вікторович	Дніпровський	ВПО	
Балмак	Олександр	Михайлович	Дніпровський	Учасники бойових дій, ВПО	
Щебетун	Василь	Вікторович	Дніпровський	Учасники бойових дій, ВПО	
Шунь	Данило	Володимирович	Дніпровський	Інваліди, ВПО	
Кушта	Валентин	Миколайович	Дніпровський	ВПО, Багатодітні	

Рядків на сторінку 10 1–10 з 12

Рисунок 3.18 –Застосування фільтрів

СТВОРИТИ

Оберіть категорію

Оберіть район

Голуб

Прізвище	Ім'я	По-батькові	Район	Пільгова категорія	
Голуб	Валентин	Михайлович	Дніпровський	Малозабезпечені, Багатодітні	
Голуб	Іван	Михайлович	Заводський	Учасники бойових дій, Пенсіонери	
Голуб	Андрій	Іванович	Хортицький	Учасники бойових дій, Пенсіонери	
Голуб	Махайло	Васильович	Хортицький	Учасники бойових дій, Пенсіонери	
Голуб	Василь	Вікторович	Заводський	Учасники бойових дій, ВПО	
Голуб	Дмитро	Миколайович	Комунарський	Учасники бойових дій, ВПО	
Голуб	Тарас	Васильович	Вознесенівський	ВПО, Багатодітні	
Голуб	Махайло	Сергійович	Дніпровський	Малозабезпечені, Багатодітні	
Голуб	Махайло	Вікторович	Дніпровський	Малозабезпечені	
Голуб	Валентин	Михайлович	Комунарський	Малозабезпечені	

Рядків на сторінку 10 1–10 з 20

Рисунок 3.19 – Пошук за ПІБ

Запоріжжя цифрове

ПЕРСОНАЛЬНІ ДАНІ ЗБЕРЕГТИ ЗАКРИТИ

Прізвище

Ім'я

По-батькові

Дата народження

Місце народження

Місце проживання

Номер телефону 1

Номер телефону 2

Номер паспорту

Ідентифікаційний код

Категорії

Район

Рисунок 3.20 – Сторінка створення нового бенефіціара

На сторінці створення нового бенефіціара є дві кнопки “Зберегти” та “Закрити”, а також поля для заповнення інформації. Якщо натиснути кнопку “Зберегти” не заповнивши всі обов’язкові поля, то з’являється помилка (рис. 3.21), якщо всі поля заповнені, то з’являється повідомлення про успіх (рис. 3.22), нова особа пільгових категорій зберігається у базу даних та відображається у таблиці.

Запоріжжя цифрове

ПЕРСОНАЛЬНІ ДАНІ ЗБЕРЕГТИ ЗАКРИТИ

Прізвище

Ім'я

По-батькові

Дата народження

Місце народження

Місце проживання

Номер телефону 1

Номер телефону 2

Номер паспорту

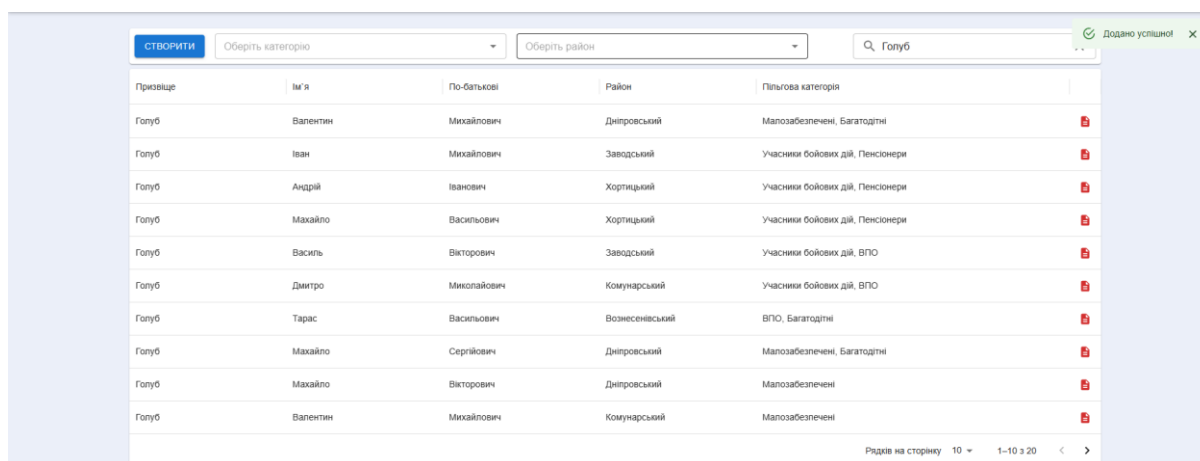
Ідентифікаційний код

Категорії

Район

Будь ласка, заповніть усі обов'язкові поля

Рисунок 3.21 – Помилка при збереженні

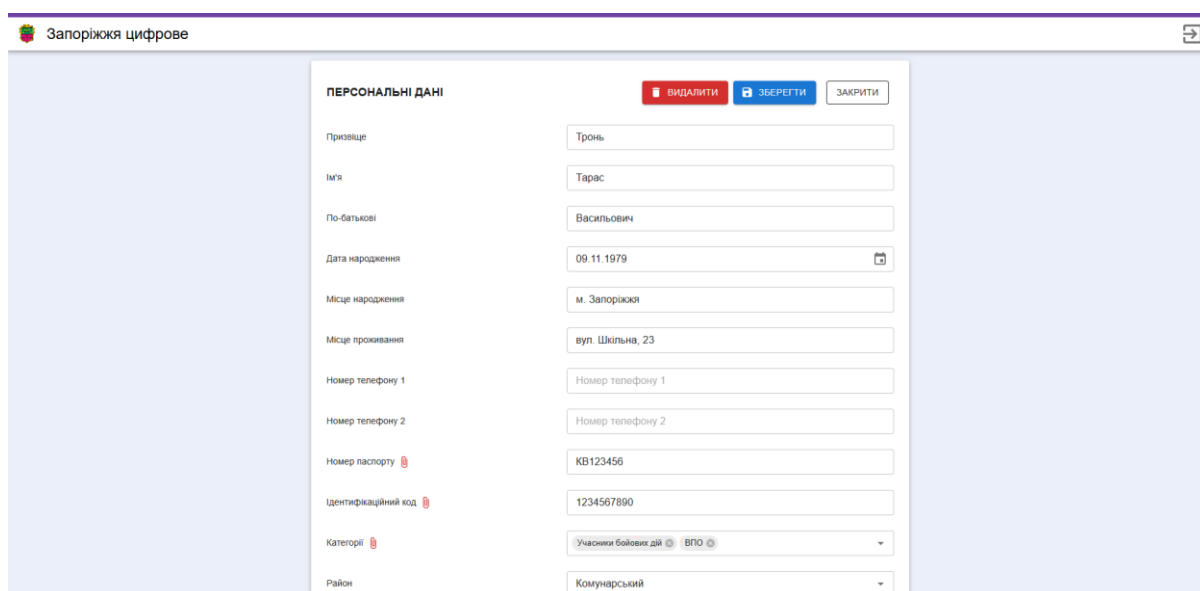


Додано успішно! ✕

Прізвище	Ім'я	По-батькові	Район	Пільгова категорія
Голуб	Валентин	Михайлович	Дніпровський	Малозабезпечені, Багатодітні
Голуб	Іван	Михайлович	Заводський	Учасники бойових дій, Пенсіонери
Голуб	Андрій	Іванович	Хортицький	Учасники бойових дій, Пенсіонери
Голуб	Махайло	Васильович	Хортицький	Учасники бойових дій, Пенсіонери
Голуб	Василь	Вікторович	Заводський	Учасники бойових дій, ВПО
Голуб	Дмитро	Миколайович	Комунарський	Учасники бойових дій, ВПО
Голуб	Тарас	Васильович	Вознесенський	ВПО, Багатодітні
Голуб	Махайло	Сергійович	Дніпровський	Малозабезпечені, Багатодітні
Голуб	Махайло	Вікторович	Дніпровський	Малозабезпечені
Голуб	Валентин	Михайлович	Комунарський	Малозабезпечені

Рядків на сторінку 10 1–10 з 20 < >

Рисунок 3.22 – Повідомлення про успіх



Запоріжжя цифрове

ПЕРСОНАЛЬНІ ДАНІ ВИДАЛИТИ ЗБЕРЕГТИ ЗАКРИТИ

Прізвище: Трощ

Ім'я: Тарас

По-батькові: Васильович

Дата народження: 09.11.1979

Місце народження: м. Запоріжжя

Місце проживання: вул. Шкільна, 23

Номер телефону 1: Номер телефону 1

Номер телефону 2: Номер телефону 2

Номер паспорту: KB123456

Ідентифікаційний код: 1234567890

Категорія: Учасники бойових дій ВПО

Район: Комунарський

Рисунок 3.23 – Сторінка редагування бенефіціара

На сторінку редагування додається кнопка “Видалити”, при натисканні якої з’являється меню вибору дії (рис. 3.24). Можна підтвердити видалення або скасувати.

На сторінці редагування є такі поля:

- прізвище;
- ім’я;
- по-батькові;
- дата народження;

- місце народження;
- місце проживання;
- номер телефону 1;
- номер телефону 2;
- номер паспорту (є можливість додати документ);
- ідентифікаційний код (є можливість додати документ);
- категорії (є можливість додати документ);
- район.

Також є акордеон з списком документів (рис. 3.25), які можна відкрити, завантажити, або видалити.

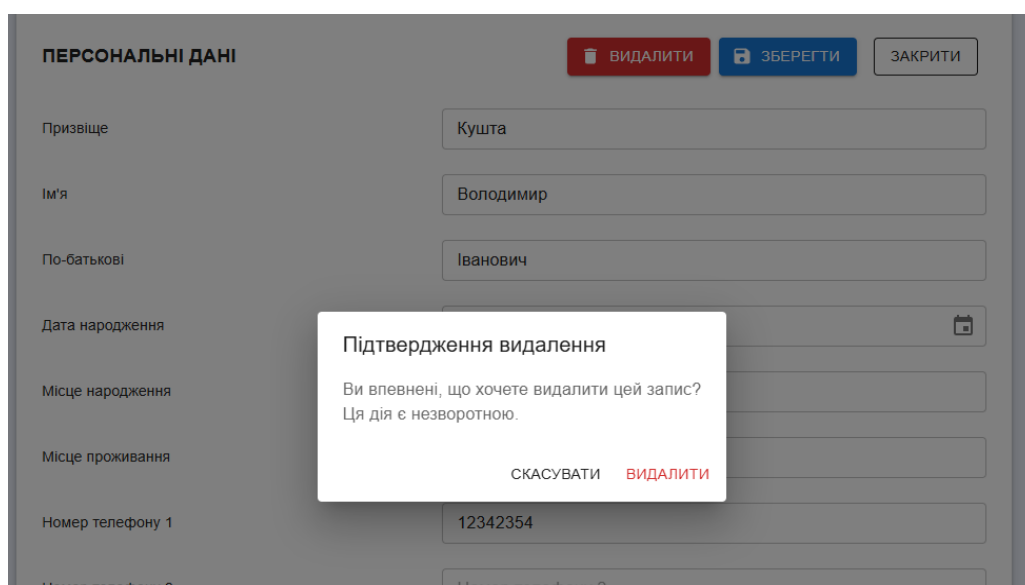


Рисунок 3.24 – Меню вибору дії

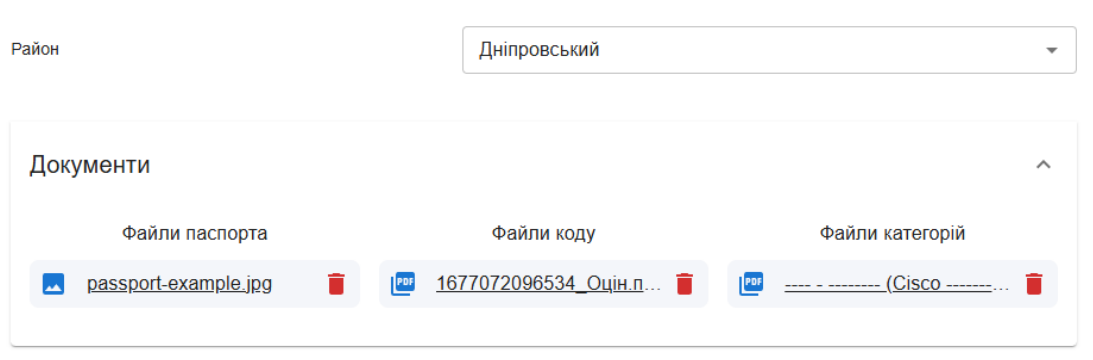


Рисунок 3.25 – Акордеон з документами

3.6 Висновки за розділом 3

Створено програмне забезпечення у формі вебзастосунку для обліку осіб пільгових категорій.

У третьому розділі представлено архітектуру програмного забезпечення, оформлену у вигляді структурних діаграм за нотацією UML, а також наведено діаграми класів, пакетів і компонентів.

Було завершено проєктування бази даних і розробка дизайну.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Опис застосування програми

Вебзастосунок призначений для оптимізації процесів надання соціальної допомоги та полегшення обліку осіб пільгових категорій.

Функції програми:

- автентифікація користувача;
- перегляд інформації про осіб пільгових категорій;
- фільтр по даним;
- створення запису про нового пільговика;
- редагування вже існуючої особи;
- видалення пільговика;
- пошук за ПІБ.

Мінімальні вимоги до клієнтської сторони:

- ОС: Windows 10 / macOS 12 / Ubuntu 20.04 і вище;
- процесор: 2-ядерний, 2,5 ГГц Intel Core i3 або AMD Ryzen 3;
- ОЗУ: 8 Гбайт;
- відеокарта: інтегрована Intel HD Graphics 4000 або AMD Vega 3 (512

Мбайт пам'яті);

- браузер: останні версії Chrome, Firefox, Edge, Safari.

Для запуску коду використовувати:

– редактор коду, який підтримує мови веб програмування, наприклад VS Code;

- систему керування базами даних PostgreSQL;
- supabase та yarn;
- браузер.

4.2 Характеристики програми

Програмним продуктом є вебзастосунок, створений для оптимізації роботи з обліком і управлінням даними користувачів. Він забезпечує зручне введення, зберігання та обробку інформації про бенефіціарів, їхніх документів та пов'язаних категорій. Система дозволяє швидко знаходити, оновлювати та контролювати дані, що сприяє ефективному управлінню та економії часу під час роботи з великими обсягами інформації.

4.3 Інструкція по експлуатації програми

Для запуску проєкту на локальному комп'ютері необхідно завантажити репозиторій з GitHub та встановити всі необхідні залежності. Для цього у терміналі слід виконати команду `yarn` у кореневій директорії проєкту.

Supabase проєкт вже створено, тому потрібно лише налаштувати підключення до нього. У кореневій директорії проєкту необхідно створити файл `.env` та додати до нього рядки:

- `VITE_SUPABASE_URL=https://sridjjmfsgijgpufhlcv.supabase.co`
- `VITE_SUPABASE_ANON_KEY=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6InNyaWRqam1mc2dpamdwdWZobGN2Iiwicm9sZSI6ImFub24iLCJpYXQiOiJlMjU2MTYwODYsImV4cCI6ImJA0MTE5MjA4Nn0.D65knOtfEB4Vtt8RPMxtcAxri-EMfmcMFSfLK22CzQ.`

Для запуску застосунку необхідно ввести команду `yarn run dev`. Після цього застосунок відкриється у браузері за посиланням `http://localhost:5173/`.

4.4 Тестування та результати роботи програмного забезпечення

На рисунку 4.1 демонструється відображення вебзастосунку у браузері Chrome.

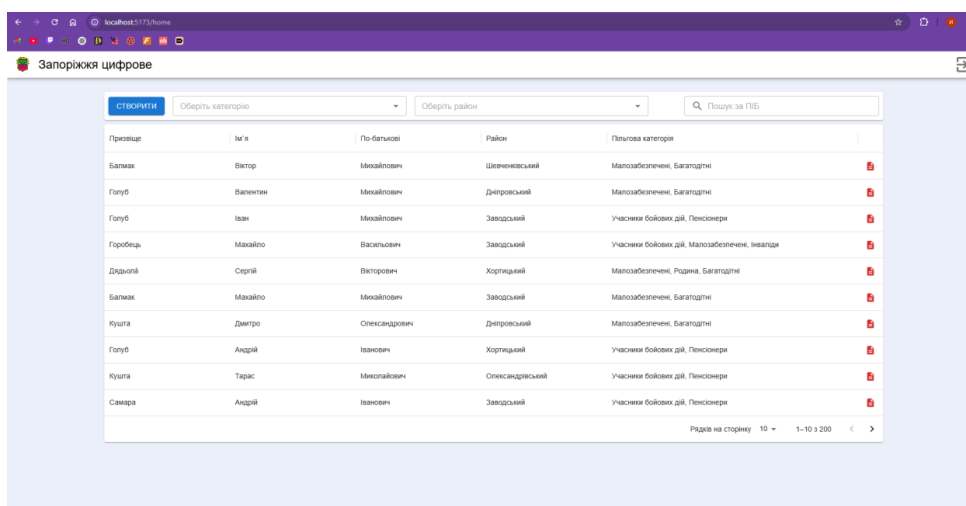


Рисунок 4.1 – Відображення у браузері Google Chrome

На рисунку 4.2 демонструється відображення вебзастосунку у браузері Microsoft Edge.

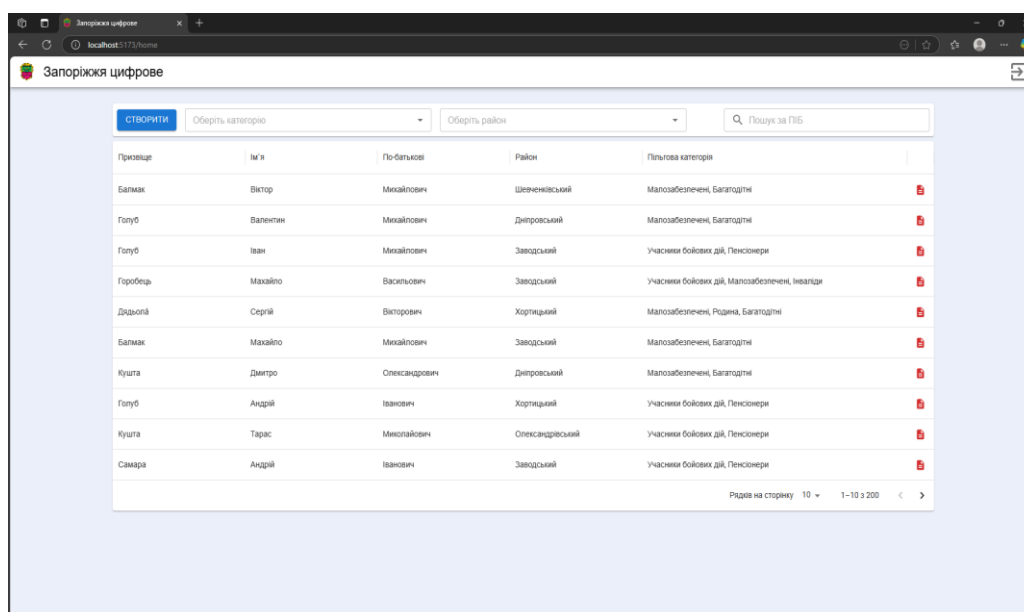


Рисунок 4.2 – Відображення у браузері Microsoft Edge

Для тестування системи написані тест-кейси. Тест-кейс – це професійна документація тестувальника, послідовність дій, спрямована на перевірку будь-якого функціоналу, що описує як прийти до очікуваного результату. Тест-кейси наведені у таблицях 4.1-4.3.

Таблиця 4.1 – Опис тест-кейсу функції авторизації

Мета	Перевірити роботу функції авторизації користувача		
Крок	Опис	Очікуваний результат	Фактичний результат
Крок 1	Відправлення форми без введених даних	Відбувається валідація полів та виводиться повідомлення про помилку	Спрацювала валідація полів і отримано повідомлення про помилку
Крок 2	Відправлення форми з невірно вказаними даними	Відбувається валідація полів та виводиться повідомлення про помилку	Спрацювала валідація полів та отримано повідомлення про помилку
Крок 3	Відправлення форми з коректними даними	Відбувається вхід до системи	Відбулася переадресація на сторінку

Таблиця 4.2 – Опис тест-кейсу процес створення нового запису пільговика

Мета	Перевірити процес створення нового запису пільговика		
Крок	Опис	Очікуваний результат	Фактичний результат
Крок 1	Дані не введені до полів	Відбувається валідація полів та виводиться повідомлення про помилку	Спрацювала валідація полів і отримано повідомлення про помилку
Крок 2	Не заповнені всі необхідні поля	Відбувається валідація полів та виводиться повідомлення про помилку	Спрацювала валідація полів та отримано повідомлення про помилку

Кінець таблиці 4.2

Крок	Опис	Очікуваний результат	Фактичний результат
Крок 3	Вірно заповнені всі необхідні поля	Відбувається валідація, виводиться повідомлення про успіх	Відбувається зберігання даних, виводиться повідомлення про успіх

Таблиця 4.3 – Опис тест-кейсу процес видалення запису пільговика

Мета	Перевірити процес створення видалення запису пільговика		
Крок	Опис	Очікуваний результат	Фактичний результат
Крок 1	Кнопка “Видалити” натиснута випадково	Виводиться модальне вікно для підтвердження дії - відхилення	Вивелося вікно для підтвердження дії, при натисканні на кнопку відхилення дані не змінюються
Крок 2	Кнопка “Видалити” натиснута спеціально	Виводиться модальне вікно для підтвердження дії - підтвердження	Вивелося вікно для підтвердження дії, при натисканні на кнопку підтвердження обраний запис видалився

4.5 Висновки за розділом 4

У розділі 4 представлено опис програмного забезпечення у формі вебзастосунку, призначеного для управління інформацією про пільговиків. Окреслено мінімальні системні вимоги, необхідні для стабільної та коректної роботи системи.

Надано детальну інструкцію щодо розгортання проєкту на персональному комп'ютері.

Також у вигляді таблиць наведено опис виконаних тест-кейсів із зазначенням мети тестування, кількості кроків, опису кожного кроку, очікуваного результату та фактичного результату тесту.

Тестування охопило такі функціональні можливості:

- авторизація користувача;
- створення нового запису пільговика;
- видалення існуючого запису.

ВИСНОВКИ

У процесі виконання дипломної роботи було проведено повний цикл розробки вебзастосунку для обліку осіб пільгових категорій. У першому розділі проаналізовано предметну область, визначено основні поняття та обґрунтовано доцільність створення системи, що забезпечує автоматизацію обліку пільговиків у соціальній службі. Сформульовано постановку задачі, а також визначено функціональні вимоги й вимоги до інтерфейсу користувача.

У другому розділі обґрунтовано вибір інструментів і технологій для реалізації проєкту. Основною мовою програмування обрано TypeScript, а для клієнтської частини – ReactJS, завдяки його гнучкості, широкому ком'юніті та великій кількості готових рішень. Для серверної частини застосовано платформу Supabase, що спростило взаємодію з базою даних PostgreSQL та реалізацію аутентифікації користувачів. Комунікація між клієнтом і сервером реалізована за допомогою Fetch API, а середовище розробки – VS Code із використанням GitHub для керування версіями.

У третьому розділі було представлено архітектуру вебзастосунку з використанням UML-нотації: діаграми класів, пакетів і компонентів. Завершено проєктування бази даних та створення дизайну інтерфейсу.

У четвертому розділі наведено опис реалізованого функціоналу вебзастосунку, визначено мінімальні системні вимоги для його запуску та надано детальну інструкцію з розгортання. Проведено тестування основних функцій системи – авторизації користувачів, створення та видалення записів пільговиків – із фіксацією результатів у вигляді тест-кейсів.

Таким чином, розроблено повнофункціональний вебзастосунок, який задовольняє поставлені вимоги та може бути використаний у роботі соціальних служб для ефективного обліку осіб пільгових категорій.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Про Національну соціальну сервісну службу України | Національна соціальна сервісна служба України [Електрон. ресурс]. – Режим доступу: <https://www.kmu.gov.ua/news/subsydiia-ta-pilha-shcho-spilnoho-ta-v-chomu-riznytsia>.
2. Субсидія та пільга: що спільного та в чому різниця? Кабінет Міністрів України [Електрон. ресурс]. – Режим доступу: <https://nssu.gov.ua/pro-nacsocsluzhbu>.
3. Пільгові категорії населення / Соціальний захист / iGov – Портал державних послуг [Електрон. ресурс]. – Режим доступу: <https://igov.org.ua/subcategory/1/3/situation/23#anchor1>.
4. 4. Урядовий контактний центр » Поняття внутрішньо переміщеної особи [Електрон. ресурс]. – Режим доступу: <https://ukc.gov.ua/knowledge/ponyattya-vnutrishno-peremishhenoyi-osoby/>.
5. Законодавче визначення поняття «інвалід», порядок та документальне оформлення визнання особи інвалідом, права та пільги при працевлаштуванні [Електрон. ресурс]. – Режим доступу: <https://mankrda.gov.ua/ohorona-praci/zakonodavche-viznachennya-ponyattya-invalid-poryadok-ta-dokumentalne-oformlennya-viznannya-osobi-invalidom-prava-ta-pilgi-pri-pracevlashtuvanni/#:~:text=2%20%D0%97%D0%B0%D0%BA%D0%BE%D0%BD%D1%83%20%E2%84%96%20875%2C%20%D0%86%D0%BD%D0%B2%D0%B0%D0%BB%D1%96%D0%B4%D0%BE%D0%BC,%D0%B2%20%D1%81%D0%BE%D1%86%D1%96%D0%B0%D0%BB%D1%8C%D0%BD%D1%96%D0%B9%20%D0%B4%D0%BE%D0%BF%D0%BE%D0%BC%D0%BE%D0%B7%D1%96%20%D1%96%20%D0%B7%D0%B0%D1%85%D0%B8%D1%81%D1%82%D1%96>.
6. Пенсіонерам – Пенсійний фонд України [Електрон. ресурс]. – Режим доступу: <https://www.pfu.gov.ua/poslugi/pensioneram/>.
7. TypeScript Tutorial | GeeksforGeeks [Electronic resource]. – Access mode: <https://www.geeksforgeeks.org/typescript/>.

8. React Introduction | GeeksforGeeks [Electronic resource]. – Access mode: <https://www.geeksforgeeks.org/reactjs-introduction/>.

9. What Is Vue JS? (Definition, Uses, Advantages) | Built In [Electronic resource]. – Access mode: <https://builtin.com/software-engineering-perspectives/vue-js>.

10. What is Supabase: A Review of Serverless Database Features - Bejamas [Electronic resource]. – Access mode: <https://bejamas.com/hub/serverless-database/supabase>.

11. Using the Fetch API - Web APIs | MDN [Electronic resource]. – Access mode: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch.

12. Axios in REACT JS. What is Axios? | by Awofesobi kolade Peace | Medium [Electronic resource]. – Access mode: <https://medium.com/@awofesobipeace/axios-in-react-js-f5a0cd821694>.

13. What Is PostgreSQL? How It Works, Use Cases, and Resources | DataCamp [Electronic resource]. – Access mode: [https://www.datacamp.com/blog/what-is-postgresql-](https://www.datacamp.com/blog/what-is-postgresql-introduction?utm_source=google&utm_medium=paid_search&utm_campaignid=19589720821&utm_adgroupid=152984011334&utm_device=c&utm_keyword=&utm_matchtype=&utm_network=g&utm_adposition=&utm_creative=733936221116&utm_targetid=dsa-2222697811358&utm_loc_interest_ms=&utm_loc_physical_ms=1012849&utm_content=ps-other~emea-en~dsa~tofu~blog-data-science&accountid=9624585688&utm_campaign=230119_1-ps-other~dsa~tofu_2-b2c_3-emea_4-prc_5-na_6-na_7-le_8-pdsh-go_9-nb-e_10-na_11-na&gad_source=1&gad_campaignid=19589720821&gbraid=0AAAAADQ9WsFAYfPWhfliVBUHOM-m4MKfM&gclid=CjwKCAjw_pDBBhBMEiwAmY02NprSukoTm65jB-cUILMaiVB1AMSRV2xNN55IWluBJb_jIRFsxw3M4RoCq18QAvD_BwE)

[introduction?utm_source=google&utm_medium=paid_search&utm_campaignid=19589720821&utm_adgroupid=152984011334&utm_device=c&utm_keyword=&utm_matchtype=&utm_network=g&utm_adposition=&utm_creative=733936221116&utm_targetid=dsa-](https://www.datacamp.com/blog/what-is-postgresql-introduction?utm_source=google&utm_medium=paid_search&utm_campaignid=19589720821&utm_adgroupid=152984011334&utm_device=c&utm_keyword=&utm_matchtype=&utm_network=g&utm_adposition=&utm_creative=733936221116&utm_targetid=dsa-2222697811358&utm_loc_interest_ms=&utm_loc_physical_ms=1012849&utm_content=ps-other~emea-en~dsa~tofu~blog-data-science&accountid=9624585688&utm_campaign=230119_1-ps-other~dsa~tofu_2-b2c_3-emea_4-prc_5-na_6-na_7-le_8-pdsh-go_9-nb-e_10-na_11-na&gad_source=1&gad_campaignid=19589720821&gbraid=0AAAAADQ9WsFAYfPWhfliVBUHOM-m4MKfM&gclid=CjwKCAjw_pDBBhBMEiwAmY02NprSukoTm65jB-cUILMaiVB1AMSRV2xNN55IWluBJb_jIRFsxw3M4RoCq18QAvD_BwE)

[2222697811358&utm_loc_interest_ms=&utm_loc_physical_ms=1012849&utm_content=ps-other~emea-en~dsa~tofu~blog-data-](https://www.datacamp.com/blog/what-is-postgresql-introduction?utm_source=google&utm_medium=paid_search&utm_campaignid=19589720821&utm_adgroupid=152984011334&utm_device=c&utm_keyword=&utm_matchtype=&utm_network=g&utm_adposition=&utm_creative=733936221116&utm_targetid=dsa-2222697811358&utm_loc_interest_ms=&utm_loc_physical_ms=1012849&utm_content=ps-other~emea-en~dsa~tofu~blog-data-science&accountid=9624585688&utm_campaign=230119_1-ps-other~dsa~tofu_2-b2c_3-emea_4-prc_5-na_6-na_7-le_8-pdsh-go_9-nb-e_10-na_11-na&gad_source=1&gad_campaignid=19589720821&gbraid=0AAAAADQ9WsFAYfPWhfliVBUHOM-m4MKfM&gclid=CjwKCAjw_pDBBhBMEiwAmY02NprSukoTm65jB-cUILMaiVB1AMSRV2xNN55IWluBJb_jIRFsxw3M4RoCq18QAvD_BwE)

[science&accountid=9624585688&utm_campaign=230119_1-ps-other~dsa~tofu_2-b2c_3-emea_4-prc_5-na_6-na_7-le_8-pdsh-go_9-nb-e_10-na_11-](https://www.datacamp.com/blog/what-is-postgresql-introduction?utm_source=google&utm_medium=paid_search&utm_campaignid=19589720821&utm_adgroupid=152984011334&utm_device=c&utm_keyword=&utm_matchtype=&utm_network=g&utm_adposition=&utm_creative=733936221116&utm_targetid=dsa-2222697811358&utm_loc_interest_ms=&utm_loc_physical_ms=1012849&utm_content=ps-other~emea-en~dsa~tofu~blog-data-science&accountid=9624585688&utm_campaign=230119_1-ps-other~dsa~tofu_2-b2c_3-emea_4-prc_5-na_6-na_7-le_8-pdsh-go_9-nb-e_10-na_11-na&gad_source=1&gad_campaignid=19589720821&gbraid=0AAAAADQ9WsFAYfPWhfliVBUHOM-m4MKfM&gclid=CjwKCAjw_pDBBhBMEiwAmY02NprSukoTm65jB-cUILMaiVB1AMSRV2xNN55IWluBJb_jIRFsxw3M4RoCq18QAvD_BwE)

[na&gad_source=1&gad_campaignid=19589720821&gbraid=0AAAAADQ9WsFAYfPWhfliVBUHOM-](https://www.datacamp.com/blog/what-is-postgresql-introduction?utm_source=google&utm_medium=paid_search&utm_campaignid=19589720821&utm_adgroupid=152984011334&utm_device=c&utm_keyword=&utm_matchtype=&utm_network=g&utm_adposition=&utm_creative=733936221116&utm_targetid=dsa-2222697811358&utm_loc_interest_ms=&utm_loc_physical_ms=1012849&utm_content=ps-other~emea-en~dsa~tofu~blog-data-science&accountid=9624585688&utm_campaign=230119_1-ps-other~dsa~tofu_2-b2c_3-emea_4-prc_5-na_6-na_7-le_8-pdsh-go_9-nb-e_10-na_11-na&gad_source=1&gad_campaignid=19589720821&gbraid=0AAAAADQ9WsFAYfPWhfliVBUHOM-m4MKfM&gclid=CjwKCAjw_pDBBhBMEiwAmY02NprSukoTm65jB-cUILMaiVB1AMSRV2xNN55IWluBJb_jIRFsxw3M4RoCq18QAvD_BwE)

[m4MKfM&gclid=CjwKCAjw_pDBBhBMEiwAmY02NprSukoTm65jB-](https://www.datacamp.com/blog/what-is-postgresql-introduction?utm_source=google&utm_medium=paid_search&utm_campaignid=19589720821&utm_adgroupid=152984011334&utm_device=c&utm_keyword=&utm_matchtype=&utm_network=g&utm_adposition=&utm_creative=733936221116&utm_targetid=dsa-2222697811358&utm_loc_interest_ms=&utm_loc_physical_ms=1012849&utm_content=ps-other~emea-en~dsa~tofu~blog-data-science&accountid=9624585688&utm_campaign=230119_1-ps-other~dsa~tofu_2-b2c_3-emea_4-prc_5-na_6-na_7-le_8-pdsh-go_9-nb-e_10-na_11-na&gad_source=1&gad_campaignid=19589720821&gbraid=0AAAAADQ9WsFAYfPWhfliVBUHOM-m4MKfM&gclid=CjwKCAjw_pDBBhBMEiwAmY02NprSukoTm65jB-cUILMaiVB1AMSRV2xNN55IWluBJb_jIRFsxw3M4RoCq18QAvD_BwE)

[cUILMaiVB1AMSRV2xNN55IWluBJb_jIRFsxw3M4RoCq18QAvD_BwE.](https://www.datacamp.com/blog/what-is-postgresql-introduction?utm_source=google&utm_medium=paid_search&utm_campaignid=19589720821&utm_adgroupid=152984011334&utm_device=c&utm_keyword=&utm_matchtype=&utm_network=g&utm_adposition=&utm_creative=733936221116&utm_targetid=dsa-2222697811358&utm_loc_interest_ms=&utm_loc_physical_ms=1012849&utm_content=ps-other~emea-en~dsa~tofu~blog-data-science&accountid=9624585688&utm_campaign=230119_1-ps-other~dsa~tofu_2-b2c_3-emea_4-prc_5-na_6-na_7-le_8-pdsh-go_9-nb-e_10-na_11-na&gad_source=1&gad_campaignid=19589720821&gbraid=0AAAAADQ9WsFAYfPWhfliVBUHOM-m4MKfM&gclid=CjwKCAjw_pDBBhBMEiwAmY02NprSukoTm65jB-cUILMaiVB1AMSRV2xNN55IWluBJb_jIRFsxw3M4RoCq18QAvD_BwE)

14. What is Visual Studio Code? Microsoft's extensible code editor | InfoWorld [Electronic resource]. – Access mode: <https://www.infoworld.com/article/2335960/what-is-visual-studio-code-microsofts-extensible-code-editor.html>.

15. How to use PhpStorm in 9 steps (plus its key features) | Indeed.com UK [Electronic resource]. – Access mode: <https://uk.indeed.com/career-advice/career-development/how-to-use-php-storm>.

16. What Is GitHub? A Beginner's Introduction to GitHub [Electronic resource]. – Access mode: <https://kinsta.com/knowledgebase/what-is-github/>.

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення у вигляді вебзастосунку для соціальної служби.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Розробка вебзастосунку для соціальної служби» затверджене наказом Національного університету «Запорізька політехніка» No 209 від 28 квітня 2025 р.

A.2 Призначення розробки

Проект призначений для працівників соціальних служб, який допомагає оперативно обробляти запити, вести облік наданих послуг і контролювати перебіг соціальних програм.

A.3 Основні вимоги до програми, що розробляється

A.3.1 Вимоги до функціональних характеристик

До ключових функціональних вимог відносяться:

- автентифікація в системі;
- перегляд таблиці з особами пільгових категорій;
- виведення даних за певними критеріями.

Авторизований користувач повинен мати наступні можливості:

- авторизація у системі;
- ведення електронного реєстру пільговиків у форматі зручної таблиці;
- можливість перегляду детальної інформації про пільговика при виборі відповідного запису;
- редагування та оновлення даних;

- пошук, фільтрація та сортування записів.

A.3.2 Вимоги до інтерфейсу програми

Основні вимоги до інтерфейсу вебзастосунку:

- інтерфейс має бути інтуїтивно зрозумілим і легким у використанні;
- дизайн виконаний в єдиній стилістиці з використанням стриманої кольорової гами;
- адаптивність і гнучкість інтерфейсу для зручної роботи на різних пристроях.

A.3.3 Вимоги до надійності

Програмне забезпечення має гарантувати стабільну та безперебійну роботу.

A.3.4 Умови експлуатації

Для використання програми необхідно мати доступ до її вихідного коду, розміщеного на GitHub, дотримуватись визначених апаратних вимог та виконати налаштування відповідно до інструкцій з експлуатаційної документації.

A.3.5 Вимоги до складу та параметрів технічний засобів

Мінімальні вимоги до клієнтської сторони:

- ОС: Windows 10 / macOS 12 / Ubuntu 20.04 і вище;
- процесор: 2-ядерний, 2,5 ГГц Intel Core i3 або AMD Ryzen 3
- ОЗУ: 8 Гбайт;
- відеокарта: інтегрована Intel HD Graphics 4000 або AMD Vega 3 (512 Мбайт пам'яті);

- браузер: останні версії Chrome, Firefox, Edge, Safari.

A.3.6 Вимоги до маркування і пакування

На пакуванні повинна бути назва програми – «Вебзастосунок для соціальної служби».

A.4 Вхідні дані до роботи

Вхідними даними до програмно забезпечення для соціальної служби є інформація про осіб пільгових категорій.

Вихідними даними до програмно забезпечення для соціальної служби є записи в базі даних про осіб пільгових категорій.

ДОДАТОК Б
Текст програми

```

import AttachButton from '../components/AttachButton'
import AttachDialog from '../components/Dialogs/AttachDialog'
import ConfirmationDialog from '../components/Dialogs/ConfirmationDialog'
import DocumentsAccordion from '../components/Docs/DocAccordion'
import FamilyAccordion from '../components/Family/FamilyAccordion'
import CustomDropdown from '../components/Form/CustomDropdown'
import
                                CustomFormContainer                                from
'../components/Form/CustomFormContainer'
import
                                CustomFormItemContainer                                from
'../components/Form/CustomFormItemConatiner'
import CustomInputField from '../components/Form/CustomInputField'
import { HOME } from '../constants/routes'
import { catalogToDocType } from '../helpers/catalogToDocType'
import { useBeneficiaries } from '../hooks/useBeneficiaries'
import Topbar from '../layout/Topbar/topbar'
import { useSupabaseSession } from '../providers/supabaseProvider'
import {
  BaseCatalog,
  BeneficiaryModel,
  DocModel,
  DocType,
} from '../types/beneficiaries'
import { beneficiaryValidationSchema } from './beneficiary-schema'
import { Alert, AlertColor, Snackbar } from '@mui/material'
import { AdapterDayjs } from '@mui/x-date-pickers/AdapterDayjs'
import { DatePicker } from '@mui/x-date-pickers/DatePicker'
import
    {
      LocalizationProvider
    }
    from
    '@mui/x-date-
pickers/LocalizationProvider'
import { ukUA } from '@mui/x-date-pickers/locales'

```

```

import dayjs from 'dayjs'
import 'dayjs/locale/uk'
import utc from 'dayjs/plugin/utc'
import { useFormik } from 'formik'
import { useEffect, useState } from 'react'
import { useLocation, useNavigate } from 'react-router-dom'

dayjs.extend(utc)

const Beneficiary = () => {
  const ukraineLocale = {
    ...ukUA.components.MuiLocalizationProvider.defaultProps.localeText,
    fieldYearPlaceholder: (params: { digitAmount: number }) =>
      'P'.repeat(params.digitAmount),
    fieldMonthPlaceholder: (params: { contentType: string }) =>
      params.contentType === 'letter' ? 'MMMM' : 'MM',
    fieldDayPlaceholder: () => 'ДД',
  }

  const location = useLocation()
  const { isAdmin } = useSupabaseSession()
  const { id } = location.state || {}
  const {
    fetchBeneficiaryById,
    createBeneficiary,
    updateBeneficiary,
    deleteBeneficiary,
    downloadFile,
  } = useBeneficiaries()

```

```

const [snackbar, setSnackbar] = useState<{
  message: string
  severity: AlertColor
  open: boolean
}>({
  message: "",
  severity: 'success',
  open: false,
})
const [docToDelete, setDocToDelete] = useState<DocModel | null>(null)

const [docs, setDocs] = useState<DocModel[]>([])
const [changedDocs, setChangedDocs] = useState<DocModel[]>([])
const [openDialog, setOpenDialog] = useState<{
  isOpen: boolean
  docType: DocType[]
}>({
  isOpen: false,
  docType: [],
})
const showSnackbar = (severity: AlertColor, message: string) => {
  setSnackbar({ message, severity, open: true })
}

const [dialogOpen, setDialogOpen] = useState(false)
const [deleteId, setDeleteId] = useState<string | null>(null)

const formik = useFormik<BeneficiaryModel>({
  initialValues: {

```

```

id: id || ",
last_name: ",
first_name: ",
middle_name: ",
place_of_birth: ",
address: ",
date_of_birth: ",
passport: ",
identification_code: ",
phone_number: ",
phone_number_extra: ",
district_id: null,
districts: { id: ", name: " },
categories: [],
is_all_docs: false,
},
validationSchema: beneficiaryValidationSchema,
onSubmit: async (values) => {
  try {
    const { id, districts, ...beneficiaryValues } = values
    if (districts?.id) {
      beneficiaryValues.district_id = districts.id
    }

    beneficiaryValues.docs = changedDocs
    if (id) {
      await updateBeneficiary(id, beneficiaryValues)
      navigate(HOME, {
        state: {

```

```

        message: 'Оновлено успішно!',
        severity: 'success',
        open: true,
      },
    })
  } else {
    await createBeneficiary(beneficiaryValues)
    navigate(HOME, {
      state: {
        message: 'Додано успішно!',
        severity: 'success',
        open: true,
      },
    })
  }
} catch (error) {
  console.error('Error saving beneficiary:', error)
  showSnackBar('error', 'Помилка при збереженні користувача')
}
},
})

```

```
const navigate = useNavigate()
```

```

useEffect(() => {
  const fetchBeneficiaryData = async () => {
    try {
      const data = await fetchBeneficiaryById(id)
      if (data) {

```

```

        data.date_of_birth = data.date_of_birth
        ? new Date(data.date_of_birth)
        : ""
    if (data.phone_number_extra == null) {
        data.phone_number_extra = ""
    }
    formik.setValues(data)

    setDocs(data.docs || [])
  }
} catch (error) {
  console.error('Error fetching beneficiary data:', error)
}
}

id && fetchBeneficiaryData()
}, [id])

const handleDeleteClick = (id?: string) => {
  if (id) {
    setDeleteId(id)
    setDialogOpen(true)
  } else {
    console.error('ID is undefined')
  }
}

const handleDialogClose = () => {
  setDialogOpen(false)
}

```

```

    setDeleteId(null)
    setDocToDelete(null)
  }

```

```

const handleConfirmDelete = async () => {
  if (docToDelete) {
    onDocumentChange([docToDelete!])
  }
  if (deleteId) {
    try {
      await deleteBeneficiary(deleteId)
      navigate(HOME, {
        state: {
          message: 'Видалено успішно!',
          severity: 'success',
          open: true,
        },
      })
    } catch (error) {
      console.error('Error while deleting:', error)
      showSnackbar('error', 'Помилка при видаленні')
    }
  }
  handleDialogClose()
}

```

```

const handleFormSubmit = async () => {
  const errors = await formik.validateForm()

```

```

if (Object.keys(errors).length > 0) {
  showSnackbar('error', "Будь ласка, заповніть усі обов'язкові поля")
  formik.setTouched(
    Object.keys(errors).reduce((acc, key) => ({ ...acc, [key]: true }), {})
  )
  return
}

```

```

  formik.handleSubmit()
}

```

```

const onDocumentChange = (changedDocs: DocModel[]) => {
  setChangedDocs((prevDocs) => [...prevDocs, ...changedDocs])
  setDocs((prevItems) =>
    changedDocs.reduce(
      (acc, doc) =>
        acc.includes(doc)
          ? acc.filter((item) => item !== doc)
          : [...acc, doc],
      prevItems
    )
  )
}

```

```

const handleAttachClick = (docType: DocType[]) => {
  setOpenDialog({ isOpen: true, docType })
}

```

```

const handleDelete = (doc: DocModel) => {

```

```

setDocToDelete(doc)
setDialogOpen(true)
}

```

```

const handleOpenFile = async (doc: DocModel) => {
  if (doc.url) window.open(await downloadFile(doc.url))
  else window.open(URL.createObjectURL(doc.file!), '_blank')
}

```

```

const hasAllDocuments = (values: BaseCatalog[] | BaseCatalog | null) => {
  return catalogToDocType(values).every((type) =>
    docs.some((doc) => doc.type === type)
  )
}

```

```

const isFamilyCategorySelected = () => {
  const catTypes = catalogToDocType(
    formik.values.categories ? formik.values.categories : null
  )
  return catTypes.some((category) => category === DocType.family)
}

```

```

const handleCustomDropdownValuesChange = (
  values: BaseCatalog[] | BaseCatalog | null
) => {
  const selectedCategories = Array.isArray(values)
    ? values.map((val) => val.name)
    : values?.name
    ? [values]

```

```

      : []
const categories = [
  ...selectedCategories,
  'Паспорт',
  'Ідентифікаційний код',
]

for (const doc of docs) {
  if (!categories.includes(doc.type)) {
    onDocumentChange([doc])
  }
}
formik.setFieldValue('categories', values)
}

return (
  <>
    <Topbar position="relative" />
    <CustomFormContainer
      handleSubmit={handleFormSubmit}
      formIsSubmitting={formik.isSubmitting}
      goBackRoute="/home"
      title="Персональні дані"
      handleDelete={
        id ? () => handleDeleteClick(formik.values.id) : undefined
      }
    >
    <CustomInputField
      inputType="text"

```

```

        title="Призвіще"
        fieldName="last_name"
        register={formik.getFieldProps}
        isError={!formik.errors.last_name && !formik.touched.last_name}
        isRequired
      />
    <CustomInputField
      inputType="text"
      title="Ім'я"
      fieldName="first_name"
      register={formik.getFieldProps}
      isError={!formik.errors.first_name && !formik.touched.first_name}
      isRequired
    />
    <CustomInputField
      inputType="text"
      title="По-батькові"
      fieldName="middle_name"
      register={formik.getFieldProps}
      isError={!formik.errors.middle_name
!!formik.touched.middle_name}
      isRequired
    />

    <LocalizationProvider
      dateAdapter={AdapterDayjs}
      adapterLocale="uk"
      localeText={ukraineLocale}
    >

```

```

<CustomFormItemContainer title="Дата народження">
  <DatePicker
    value={
      formik.values.date_of_birth
      ? dayjs.utc(formik.values.date_of_birth)
      : null
    }
    onChange={(value) => {
      const utcDateValue = value ? dayjs.utc(value).toISOString() : "
      formik.setFieldValue('date_of_birth', utcDateValue)
    }}
    slotProps={{
      textField: {
        fullWidth: true,
        error:
          !!formik.errors.date_of_birth &&
          !!formik.touched.date_of_birth,
        required: true,
        size: 'small',
        placeholder: 'Дата народження',
      },
    }}
  />
</CustomFormItemContainer>
</LocalizationProvider>

<CustomInputField
  inputType="text"
  title="Місце народження"

```

```

    fieldName="place_of_birth"
    register={formik.getFieldProps}
    isError={
      !!formik.errors.place_of_birth && !!formik.touched.place_of_birth
    }
    isRequired
  />
<CustomInputField
  inputType="text"
  title="Місце проживання"
  fieldName="address"
  register={formik.getFieldProps}
  isError={!!formik.errors.address && !!formik.touched.address}
  isRequired
/>
<CustomInputField
  inputType="text"
  title="Номер телефону 1"
  fieldName="phone_number"
  register={formik.getFieldProps}
  isError={
    !!formik.errors.phone_number && !!formik.touched.phone_number
  }
  isRequired
/>
<CustomInputField
  inputType="text"
  title="Номер телефону 2"
  fieldName="phone_number_extra"

```

```

    register={formik.getFieldProps}
  />
  <CustomInputField
    inputType="text"
    title="Номер паспорту"
    fieldName="passport"
    register={formik.getFieldProps}
    isError={!formik.errors.passport && !formik.touched.passport}
    isRequired
    actionButton={
      <AttachButton
        attached={docs.some((doc) => doc.type === DocType.passport)}
        onClick={() => handleAttachClick([DocType.passport])}
      />
    }
  />

  <CustomInputField
    inputType="text"
    title="Ідентифікаційний код"
    fieldName="identification_code"
    register={formik.getFieldProps}
    actionButton={
      <AttachButton
        attached={docs.some(
          (doc) => doc.type === DocType.identification_code
        )}
        onClick={() => handleAttachClick([DocType.identification_code])}
      />
    }
  />

```

```

    }
  />

  <CustomFormItemContainer
    title="Категорії"
    actionButton={
      formik.values.categories?.length ? (
        <AttachButton
          attached={hasAllDocuments(formik.values.categories)}
          onClick={() =>
            handleAttachClick(
              catalogToDocType(
                formik.values.categories ? formik.values.categories : null
              )
            )
          }
        />
      ) : undefined
    }
  >

  <CustomDropdown
    multiple
    placeholder="Оберіть категорію"
    catalogName="categories"
    values={formik.values.categories || []}
    onValuesChange={(values) =>
      handleCustomDropdownValuesChange(values)
    }
    error={!formik.errors.categories && !formik.touched.categories}
  />

```

```

/>
</CustomFormItemContainer>

<>
  {isAdmin && (
    <CustomFormItemContainer title="Район">
      <CustomDropdown
        catalogName="districts"
        placeHolder="Оберіть район"
        values={formik.values.districts || null}
        onValuesChange={(value) =>
          formik.setFieldValue('districts', value)
        }
        hidePlaceholderOnSelect={false}
        error={!formik.errors.districts && !formik.touched.districts}
      />
    </CustomFormItemContainer>
  )}

  <DocumentsAccordion
    docs={docs}
    handleDelete={handleDelete}
    handleOpenFile={handleOpenFile}
  />

  {isFamilyCategorySelected() && (
    <FamilyAccordion
      docs={docs}
      handleDelete={handleDelete}
      handleOpenFile={handleOpenFile}
    />
  )}

```

```

        />
    )}
</>
</CustomFormContainer>

<ConfirmationDialog
  open={dialogOpen}
  onClose={handleDialogClose}
  onConfirm={handleConfirmDelete}
/>

<AttachDialog
  docs={docs}
  onChange={onDocumentChange}
  onClose={() => setOpenDialog({ isOpen: false, docType: [] })}
  onDelete={handleDelete}
  open={openDialog.isOpen}
  types={openDialog.docType}
  openFile={handleOpenFile}
/>

<Snackbar
  anchorOrigin={{ vertical: 'top', horizontal: 'right' }}
  sx={{ margin: 5 }}
  open={snackbar.open}
  autoHideDuration={6000}
  onClose={() => setSnackbar({ ...snackbar, open: false })}
>

  <Alert
    onClose={() => setSnackbar({ ...snackbar, open: false })}

```

```
        severity={snackbar.severity}  
      >  
        {snackbar.message}  
      </Alert>  
    </Snackbar>  
  </>  
)  
}
```

```
export default Beneficiary
```

ДОДАТОК В
Слайди презентації

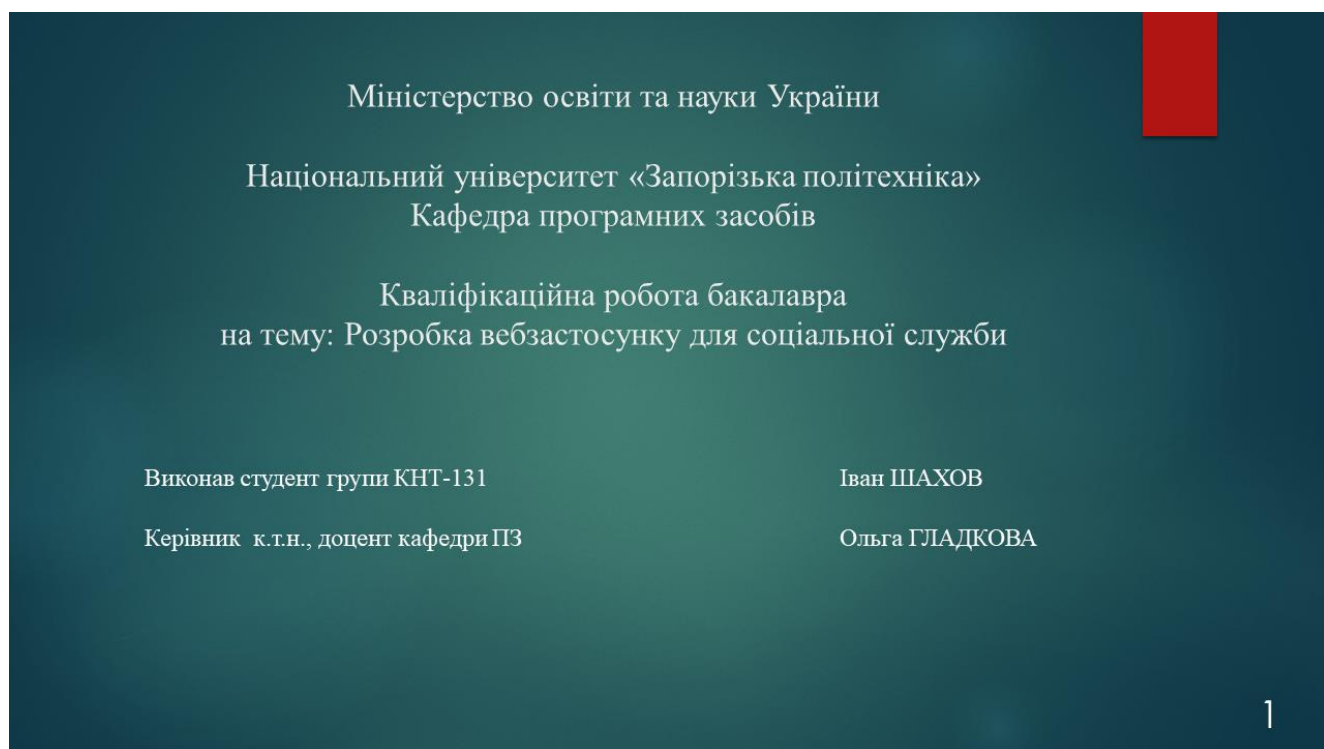


Рисунок В.1 – Слайд №1

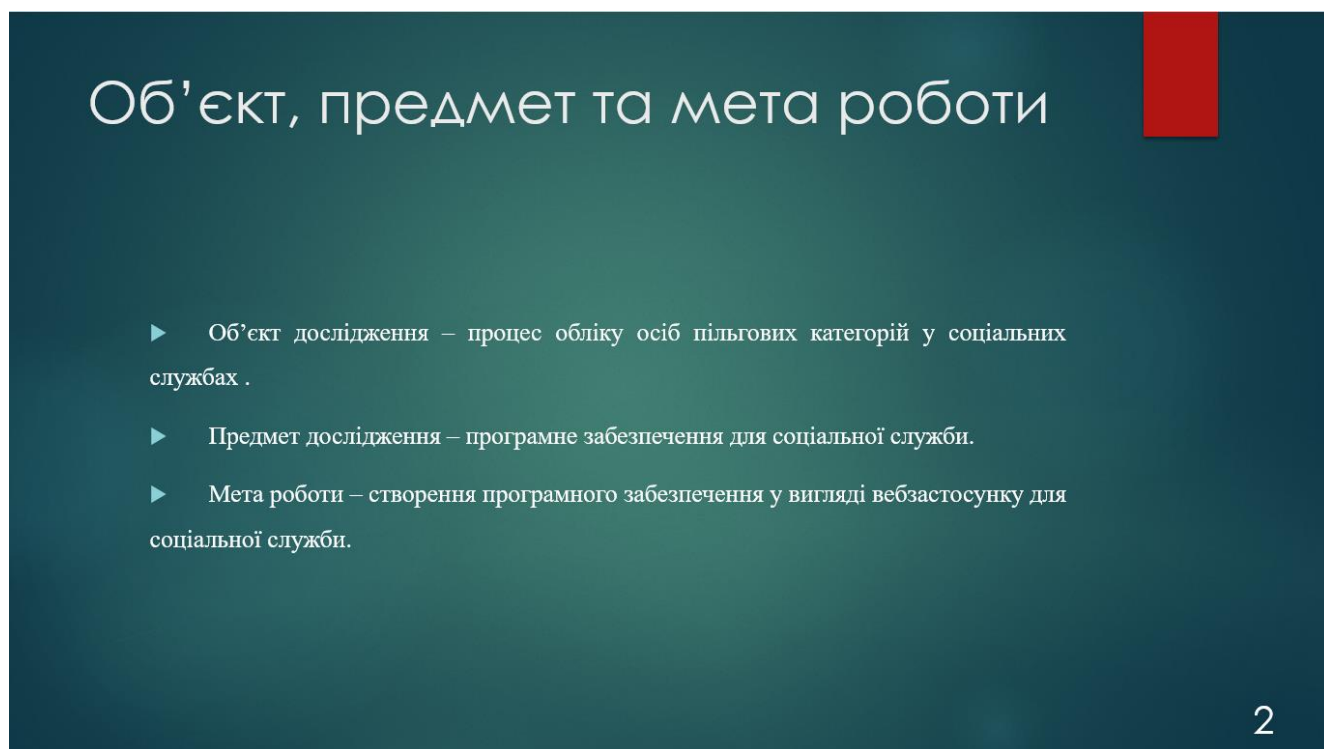


Рисунок В.2 – Слайд №2

Завдання

Завдання до роботи:

- ▶ проєктування програмного забезпечення у вигляді вебзастосунку;
- ▶ порівняння програмного забезпечення;
- ▶ розробка вебзастосунку для соціальної служби;
- ▶ тестування розробленого продукту.

3

Рисунок В.3 – Слайд №3

Вибір мови програмування

Особливість	TypeScript	JavaScript
Статична типізація	+	-
Динамічна типізація	-	+
Інтерфейси	+	-
Виявлення помилок на етапі компіляції	+	-
Підтримка сучасних стандартів ECMAScript	+	+
Підтримка у великих проєктах	+	+
Абстрактні класи	+	-

4

Рисунок В.4 – Слайд №4

Вибір технології для клієнтської частини

Особливість	React	Vue
Компонентна архітектура	+	+
Віртуальний DOM	+	+
Підтримка TypeScript	++	+
Гнучкість налаштувань архітектури	+	-
Легка масштабованість	+	-

5

Рисунок В.5 – Слайд №5

Вибір технології для серверної частини

Характеристика	Supabase	Node.js
Масштабованість	+ Легко масштабується через хмарну інфраструктуру	+ Висока, але потребує налаштування серверу
Простота використання	+ Легко інтегрується, мінімальна настройка	- Потрібно більше налаштувань та знань для налаштування серверу
Підтримка бази даних	+ Вбудована підтримка PostgreSQL з автоматичним API	- Потрібно налаштовувати базу даних вручну
Аутентифікація	+ Вбудована підтримка аутентифікації та керування користувачами	- Потрібно налаштовувати аутентифікацію вручну
Технології	+ Інтеграція з JavaScript, автоматичні API	- Потрібно налаштовувати сервер та фронтенд окремо

6

Рисунок В.6 – Слайд №6

Вибір технології бази даних

Характеристика	PostgreSQL	MySQL
Відкритий вихідний код	+	+
Підтримка ACID транзакцій	+	+
Розширені SQL-запити	+	-
Повнотекстовий пошук	+	-
Робота з JSON	+	+

7

Рисунок В.7 – Слайд №7

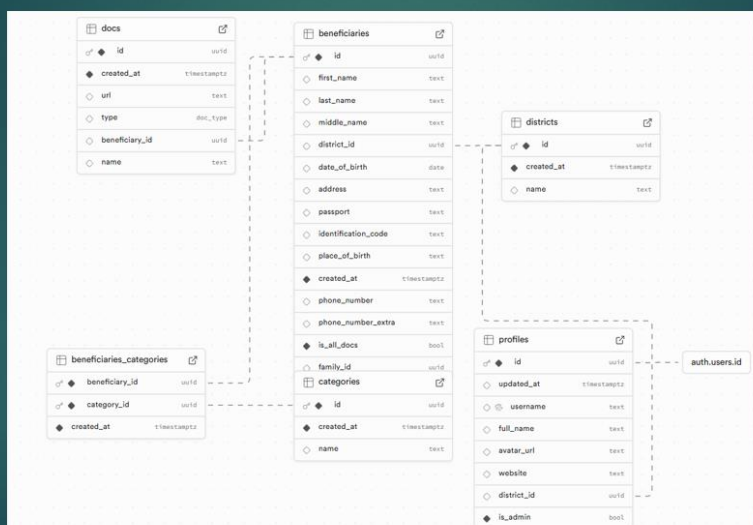
Вибір середовища розробки

Функціональність	VS Code	PhpStorm
Відкритий вихідний код	+	-
Інтеграція з Git	+	+
Підтримка Node.js	+	+
Зовнішній вигляд	+	-
Легкість програми	+	-

8

Рисунок В.8 – Слайд №8

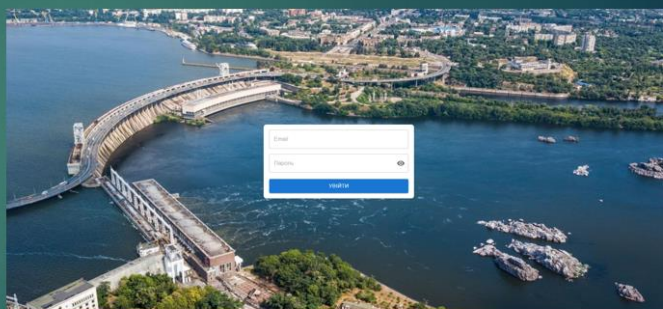
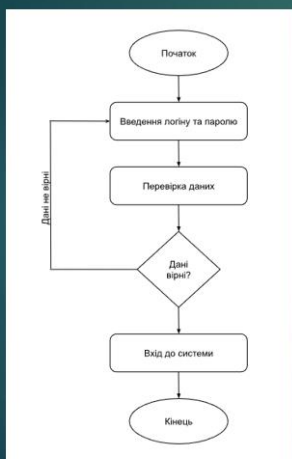
Схема бази даних



9

Рисунок В.9 – Слайд №9

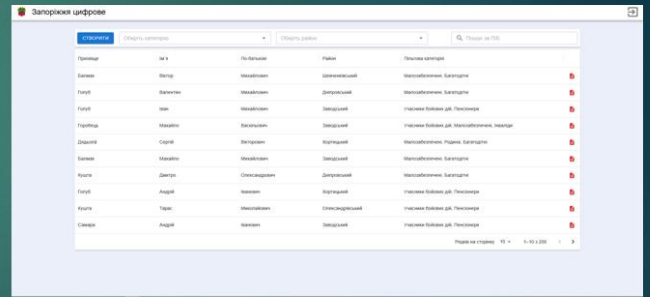
Сторінка авторизації



10

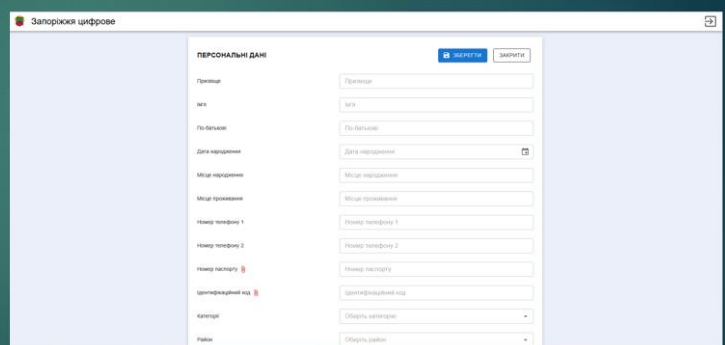
Рисунок В.10 – Слайд №10

Домашня сторінка



11

100



12

Сторінка редагування особи



Запоріжжя цифрове

ПЕРСОНАЛЬНІ ДАНІ

Ім'я:

Пі:

Пі:

Дата народження:

Місце народження:

Вір. Штатна:

Номер телефону 1:

Номер телефону 2:

Номер паспорта:

Ідентифікаційний код:

Категорія:

Район:

Кнопки:

13

Рисунок В.13 – Слайд №13

Висновок

В ході виконання дипломної роботи бакалавра було проаналізовано та досліджено процес розробки ПЗ для соціальної служби. Розроблене ПЗ у вигляді вебзастосунку для обліку осіб пільгових категорій забезпечує виконання таких функцій:

- автентифікація користувача;
- перегляд усіх осіб пільгових категорій;
- фільтрація осіб за категоріями та районами проживання;
- пошук за ПІБ;
- додавання нових та редагування вже існуючих бенефіціарів.

При тестуванні системи було виявлено, що система працює коректно та може використовуватись за призначенням.

14

Рисунок В.14 – Слайд №14