

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування назва факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавр

(ступінь вищої освіти)

на тему РОЗРОБКА ОНЛАЙН-СЕРВІСНОГО ЗАСТОСУНКУ ДЛЯ
СПІВВЛАСНИКІВ БАГАТОКВАРТИРНОГО БУДИНКУ
DEVELOPMENT OF AN ONLINE SERVICE APPLICATION FOR CO-OWNERS
OF A BLOCK OF FLATS

Виконав: студент 4 курсу, групи КНТ-129 сп

Спеціальності 121 Інженерія програмного

забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Цибульський К.Д.

(прізвище та ініціали)

Керівник Степаненко О.О.

(прізвище та ініціали)

Рецензент Казурова А.Є.

(прізвище та ініціали)

2022

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

Інститут, факультет ФКНТ
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.
С.О. Субботін
“ ” 20__ року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

Цибульського Костянтина Дмитровича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка онлайн-сервісного застосунку для співвласників багатоквартирного будинку. Development of an Online Service Application for Co-owners of a Block of Flats

керівник проєкту (роботи) Степаненко О.О., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “ ” 2022 року №__

2. Строк подання студентом проєкту (роботи) _____
3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Реферат. Перелік скорочень та умовних познач. Вступ. 1. Аналіз предметної області. 2. Вибір та обґрунтування структури системи, яка проєктується. 3. Розробка програми. 4. Керівництво користувача. Висновки. Перелік джерел посилання. Додатки.

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

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Степаненко О.О., доцент		
Нормоконтроль	Камінська Ж. К., асистент		

7. Дата видачі завдання “__” _____ року.

КАЛЕНДАРНИЙ ПЛАН

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

Студент

_____ Цибульський К.Д.
(підпис) (прізвище та ініціали)

Керівник проєкту (роботи)

_____ Степаненко О.О.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка: 106 сторінок, 25 рисунків, 3 таблиць, 22 посилань, 4 додатки.

ВЕБЗАСТОСУНОК, ОСББ, VUE.JS, PINIA, ДІАГРАМА, ГОЛОСУВАННЯ, ЗВІТ, NODE.JS, EXPRESS, JWT

Об'єкт дослідження – об'єднання співвласників багатоквартирного будинку.

Предмет дослідження – методи побудови онлайн-сервісного застосунку для співвласників багатоквартирного будинку .

Мета роботи – розробка програмного продукту, який дозволить. Матеріали, методи та технічні засоби: інтернет-браузер Google Chrome, сервіс Google Scholar, Adobe Reader, IDE Visual Studio Code, Vue.js, Express.js.

Актуальність розробки обумовлена сучасними тенденціями до цифровізації всіх форм господарювання, ускладненість комунікації між співвласниками та адміністрацією в умовах карантинних чи інших обмежень та вимогам прозорості обігу спільних коштів.

Вищеназвані проблеми вирішені за допомогою створення інструментів дистанційного комунікування, голосувань, різного роду обліків даних тощо.

В результаті роботи було розроблено онлайн-сервісний застосунок для співвласників багатоквартирного будинку, який вирішує більшість недоліків вже існуючих рішень, простий у використанні та адаптований до використання на портативних пристроях.

ABSTRACT

Explanatory note: 106 pages, 25 figures, 3 tables, 22 references, 4 appendixes.

WEB APPLICATION, ACMH, VUE.JS, PINIA, DIAGRAM, VOTING, REPORT, NODE.JS, EXPRESS, JWT

The object of research is the association of co-owners of an apartment building.

The subject of research - methods of building an online service application for co-owners of an apartment building.

The purpose of the work - to develop a software product on "Online service application for co-owners of an apartment building"

Materials, methods and technical means: Google Chrome Internet browser, Google Scholar service, Adobe Reader, Visual Studio Code IDE, Vue.js, Express.js.

The urgency of the development is due to current trends in the digitalization of all forms of management, the complexity of communication between co-owners and administration in quarantine or other restrictions and the requirements of transparency in the circulation of common funds.

The above problems are solved by creating tools for remote communication, voting, various types of data accounting and more.

As a result, an online service application was developed for apartment building co-owners, which solves most of the shortcomings of existing solutions, is easy to use and adapted to use on portable devices.

ЗМІСТ

Реферат.....	4
Перелік скорочень та умовних познач 8	8
Вступ	9
1 Аналіз предметної області	10
1.1 Об'єднання співвласників багатоквартирного будинку.....	10
1.2 Онлайн-сервісний застосунок для співвласників багатоквартирного будинку	11
1.3 Огляд існуючих рішень.....	12
1.4 Постановка завдання	15
1.5 Висновки за розділом 1	16
2 Вибір та обґрунтування структури проектованої системи.....	17
2.1 Вибір інструментарію розробки	17
2.2 Структурна схема програмного продукту	26
2.3 Висновки за розділом 2.....	29
3 Розробка програми.....	30
3.1 Структурна схема застосунку	30
3.2 Метод реалізації автентифікації користувачів	31
3.3 Способи представлення даних	35
3.4 Висновки за розділом 3.....	36
4 Керівництво користувача.....	37
4.1 Призначення та умови використання	37
4.2 Характеристики програмного продукту	37
4.3 Звернення до програмного застосунку.....	38
4.4 Вхідні та вихідні дані	39
4.5 Повідомлення.....	41
4.6 Призначення та умови виконання	41
4.7 Використання застосунку	42
4.8 Висновки за розділом 4.....	49

Висновки.....	50
Перелік джерел посилання	51
Додаток А Технічне завдання.....	53
Додаток Б Опис програмного продукту	58
Додаток В Лістинги файлів роботи	66
Додаток Г Слайди презентації.....	100

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API – Application Programming Interface

GIL – Global Interpreter Lock

IDE – Integrated Development Environment

REST – Representational state transfer

ВЗ – вебзастосунок

ЖКГ – житлово-комунальне господарство

ПЗ – програмний застосунок

ПК – персональний комп'ютер

ОСББ – об'єднання співвласників багатоквартирного будинку

ВСТУП

У сучасному світі існує тенденція переходу всіх сфер господарювання у цифрову площину. Не оминуло це й сферу житлово-комунального господарства (ЖКГ). Ми можемо бачити зараз, що у більшості надавачів комунальних послуг, управляючих компаній, є власні сайти з особистими кабінетами для користувачів.

Але об'єднання співвласників багатоквартирних будинків (ОСББ) – це відносно невеликі самостійні суб'єкти господарювання. Вони зазвичай не мають коштів на оплату готових масштабних сервісів з автоматизації господарювання. А якщо б й мали, як правило, у таких об'єднань немає потреби у більшості функцій, які пропонує така система.

Через специфічність даної форми господарювання, яка характеризується демократичністю та деякою децентралізацією, розроблюваний застосунок має просувати прозорість управління та спрощувати колективне прийняття рішень щодо керуванням будинком.

Також інформаційний сервіс, що проєктується має на меті запропонувати можливість користувачам зручно відслідковувати витрати газу, тепла, електроенергії, води тощо. Як у об'ємах, так і в грошовому еквіваленті. Відображення інформації у вигляді таблиць та графіків робить її наочною і зрозумілою для будь-якого користувача. А фільтрація за датою дозволить слідкувати за динамікою в часі.

Голові ж ОСББ застосунок має дозволити легко інформувати мешканців та вчасно й зрозуміло їм звітувати.

Тому вважаю за необхідне провести аналіз цієї предметної області й прийняти рішення щодо доцільності розробки програмного застосунку (ПЗ) для ОСББ.

Після ухвалення позитивного рішення буде розпочате проєктування вебзастосунку на основі сформульованих вимог. Вимоги мають бути виведені на основі дослідження існуючих рішень, їх переваг та недоліків.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Далі буде розглянуто основні визначення та особливості предметної області, проведено аналіз та порівняння вже реалізованих аналогічних програмних рішень, сформульовано переваги проєктованого ПЗ над вже існуючими, сформовані вимоги до нього.

1.1 Об'єднання співвласників багатоквартирного будинку

На сьогодні об'єднання співвласників багатоквартирного будинку – це сучасний та дуже гнучкий спосіб господарювання. До того ж він максимально відповідає державній політиці децентралізації [1]-[2].

Як перевагу так, і як недолік можна відзначити спосіб прийняття рішень у такій формі управління – всі важливі рішення ухвалюються шляхом загального колективного голосування [1]-[2].

Перевагою є те, що такий спосіб дозволяє порушувати максимально актуальні проблеми та приймати рішення, що задовільняють більшість, якщо не всіх, мешканців об'єднання [1]-[2].

Недоліком – те, що необхідно скликати збори, обирати певний час, який може влаштувати не всіх. І якщо на зборах не буде присутня більшість, згідно чинного законодавства, необхідно проводити поіменне письмове опитування думки всіх неprisутніх співвласників будинку з висунутої пропозиції [1]-[2].

Також при сучасному темпі життя багато співвласників не мають часу чи не хочуть його витратити на комунікацію з головою об'єднання або на загальні збори [1]-[2].

Для голови об'єднання або управляючого є проблемою налагодження комунікації з членами об'єднання. Інформування про події, звітність про фінансову діяльність, суспільні ініціативи тощо. Все це вимагає великої кількості часу та «паперової роботи» для донесення інформації до 100% мешканців [1]-[2].

Діджиталізація дозволяє створити єдиний інформаційний простір, де більшість (якщо не всі) проблем буде дуже просто вирішити [1]-[2].

Актуальність цього напрямку доводить й те, що на державному рівні планують регламентувати подібні ініціативи або створити єдину систему електронної взаємодії та прийняття рішень в об'єднанні співвласників багатоквартирного будинку. Це пропонує законопроект від 16.12.2019 №2606 «Про внесення змін до деяких законодавчих актів щодо підвищення якості управління багатоквартирними будинками та вдосконалення діяльності об'єднань співвласників багатоквартирного будинку» [1]-[2].

1.2 Онлайн-сервісний застосунок для ОСББ

На основі того, що було зазначено вище можемо виділити основні очікування від подібних додатків. Найбільш необхідними можливостями є наявність особистого кабінету, розділу для голосування та пропозицій, оголошень, повідомлень чи звітів голови, керуючого, бухгалтера та іншої можливої адміністрації. Розглянемо детальніше [3]-[4].

Можливість користувачу мати власний кабінет, в якому він може вести облік використаних енергоресурсів, бачити статистику по них у одиницях чи грошовому еквіваленті та у різних формах представлення інформації, таких як графіки, таблиці тощо, дозволить користувачу планувати свої витрати та, можливо, задуматись про більш економне їх використання. Що в свою чергу допоможе людині заощадити кошти. Також така «економія» виглядає цікаво у розрізі світової тенденції з прийняття заходів з метою покращення загальносвітової екологічної ситуації. Якщо ж міркувати менш глобально, то історія показів лічильників та обсягів споживання допоможе користувачам виявити розбіжності в квитанціях, а орендодавцям бути більш спокійними за стан справ у здаваному помешканні [3]-[4].

Розділ для голосування та внесення пропозицій дозволяє співвласникам приймати безпосередню активну участь у визначенні подальшого шляху

розвинення та покращення будинку. Також це дає змогу адміністрації точніше розуміти потреби мешканців й своєчасно приймати необхідні рішення, планувати бюджет тощо [3]-[4].

В свою чергу, розділ зі звітами, оголошеннями та повідомленнями дає співвласникам певний зворотній зв'язок. У ньому мешканці можуть відслідковувати витрати та прибутки, що дозволяє їм краще розуміти стан справ у будинку, співвласниками якого вони є. Для адміністрації цей інструмент теж полегшує роботу, бо знімає багато питань до ний, полегшує інформування спільноти та проведення загальних голосувань з питань, що потребують колегіального рішення [3]-[4].

Також корисним може бути реалізація своєрідної «довідкової» для адміністрації. В цьому розділі буде зібрана інформація про мешканців будинку. Повне ім'я, контактна інформація, чи є він або вона власником помешкання або орендарем. Це все спростить комунікацію адміністрації з населенням.

Ще більшої актуальності перехід усіх сфер господарювання в цифрову площину набув в останні роки. Спочатку карантинні обмеження, за яких загальні зібрання були унеможливлені, потім військові дії. Все це сприяло переходу на дистанційну модель. Але це не можливо без гарно налагодженої, стабільної інфраструктури [3]-[4].

Саме через вище згадані причини, а також для нівелювання недоліків існуючих рішень та вдосконалення їх переваг і буде спроектовано новий застосунок.

1.3 Огляд існуючих рішень

Тенденція до діджиталізації не оминула багато ОСББ, і вони завели власний сайт. Але в результаті пошуків аналогів я дійшов невтішних висновків – більшість цих сайтів розроблено непрофесійно, вони мають обмежений функціонал і не відповідають вимогам сучасності, зокрема зручності використання на мобільних пристроях [5]-[6].

Далі наведено декілька прикладів, щоб проілюструвати деякі з недоліків.

Ось наприклад сайт об'єднання співвласників багатоквартирного будинку «Бажана 12» в м. Київ. Він має досить обмежений функціонал. Фактично він виконує лише інформаційну роль. Також суттєвим недоліком є досить обмежена адаптивність для мобільних пристроїв. Нижче на рис. 1.1 наведено скріншот головної сторінки цього сайту [5]-[6].

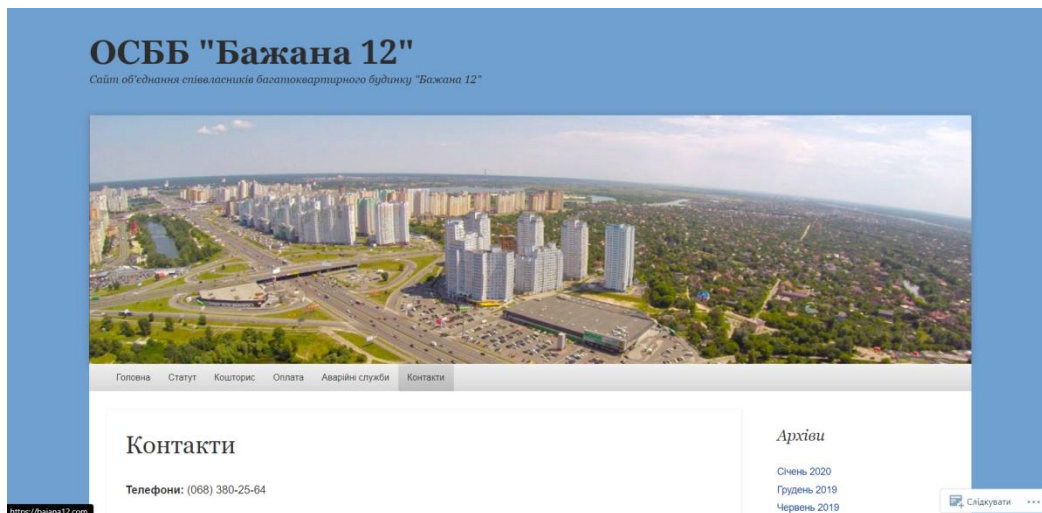


Рисунок 1.1 – Головна сторінка сайту ОСББ «Бажана 12» [7]

Всі ті ж самі зауваження можна віднести до сайту ОСББ «Зоряне» в м. Київ. Інформативність на цьому сайті значно ширша за попередній, але інший функціонал майже відсутній. Хоча головною проблемою є повна відсутність адаптивності до мобільних пристроїв. Також можна відзначити застарілий дизайн оформлення. Ілюстрацію наведено на рис. 1.2.

Схожі недоліки має й інформаційний портал ОСББ «Фундуклеєвський». Він має не дуже широкий функціонал, лише інформування про події, деяку звітність та можливість обговорення питань на внутрішньому форумі. Справа з адаптивністю на цьому сайті значно краща, хоча й не ідеальна. Головним зауваженням є дуже архаїчний дизайн оформлення сайту. В якості прикладу на рис. 1.3 наведено скріншот головної сторінки сайту [5]-[6].

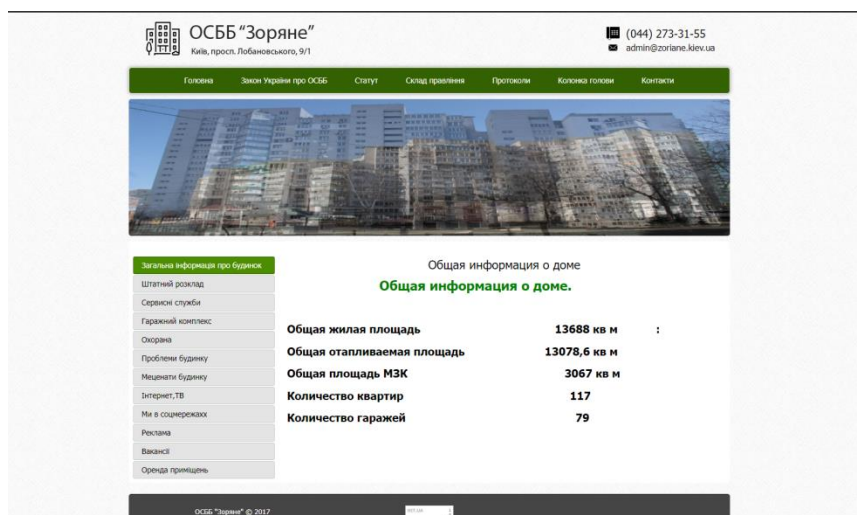


Рисунок 1.2 – Головна сторінка сайту ОСББ «Зоряне» [8]

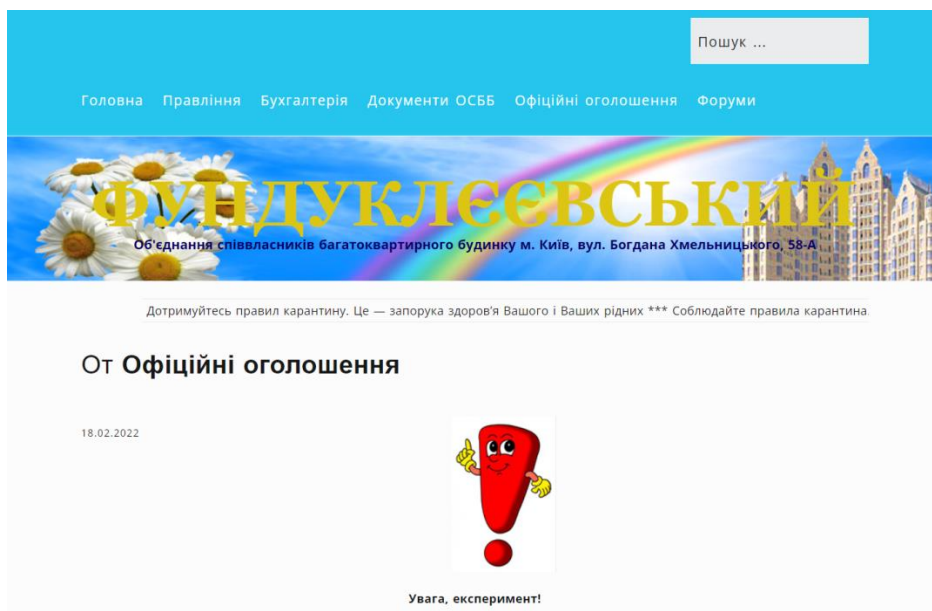


Рисунок 1.3 – Головна сторінка сайту ОСББ «Фундуклєєвський» [9]

В результаті дослідження можна зробити висновок, що більшість подібних сайтів виконують лише інформаційну функцію, є застарілими за технологіями та дизайном [5]-[6].

Окремою справою є сервіси для керування ОСББ. Вони надають значно ширший, часто надлишковий функціонал. Хоча також часто мають застарілий дизайн та технології. Але головно перешкодою для поширеного використання є

досить висока ціна таких систем. Здебільшого непомірна для невеликих об'єднань [5]-[6].

Виходячи з цього є підстава для розробки системи яка має розширений функціонал, в порівнянні зі звичайними інформаційними сайтами, але без зайвої надлишковості, яка притаманна великими системам управління.

1.4 Постанова завдання до роботи

Розроблюваний онлайн-сервісний застосунок для співвласників багатоквартирного будинку має на меті забезпечити спрощену комунікацію між співвласниками та адміністрацією, забезпечити можливість прийняття колегіальних рішень у віддаленому форматі, покращити поінформованість мешканців будинку про життя громади та дати можливість користувачам вести облік комунальних витрат помешкання.

1.4.1 Функціональні вимоги

Функціональні вимоги були визначені такі:

- реєстрація мешканців;
- додавання адміністратором інформації щодо тарифів на комунальні послуги;
- додавання адміністратором звітів;
- додавання адміністратором оголошень чи повідомлень;
- додавання користувачем даних про спожиті комунальні послуги;
- сортування даних про спожиті раніше послуги та датою чи типом;
- постановка адміністратором питання на голосування;
- додавання користувачем пропозиції на голосування;
- можливість користувача віддати свій голос за пропозицію.

1.4.2 Вимоги до інтерфейсу

Інтерфейс має відповідати сучасним вимогам дизайну, бути зручним та інтуїтивно зрозумілим для користувача. Також через повальне збільшення обсягу мобільного трафіку, інтерфейс повинен легко трансформуватися для використання на мобільних пристроях.

1.4.3 Вимоги до безпеки

Незареєстровані користувачі мають змогу переглядати лише головну інформаційну сторінку ресурсу. Для доступу до звітів, повідомлень та особистого кабінету користувач має зареєструватись.

1.5 Висновки за розділом 1

У даному розділі було досліджено предметну область «Онлайн-сервісний застосунок для співвласників багатоквартирного будинку». Аргументовано доведено необхідність проєктування та розробки нового ПЗ.

Також були розглянуті переваги та недоліки існуючих систем. На їх основі було визначено основну вимоги до розроблювального застосунку.

2 ВИБІР ТА ОБҐРУНТУВАННЯ СТРУКТУРИ ПРОЄКТОВАНОЇ СИСТЕМИ

2.1 Вибір інструментарію розробки

На самому початку проєктування кожному розробнику необхідно визначитись з технологіями, інструментами та архітектурою, що будуть використовуватись при розробці майбутнього додатку.

Виходячи з призначення застосунку, що розроблюється можна сформулювати основні можливості, які він обов'язково має мати.

Перерахуємо їх нижче:

- обробка інформації, що надходить;
- зберігання інформації в базі даних;
- введення та виведення інформації;
- керування станом застосунка.

Після того, як стало зрозуміло які саме можливості має реалізовувати система необхідно обрати інструменти та засоби розробки, такі як:

- середовище розробки (IDE);
- архітектуру застосунка;
- мову програмування;
- базу даних.

Далі по черзі детально буде розглянуто кожен з вище наведених пунктів.

2.1.1 Вибір середовища розробки

Вдало вибране середовище розробки здатне значно підвищити ефективність та зручність роботи розробника.

Існує безліч різноманітних IDE, але насправді потужних, зручних та популярних не так вже й багато. Можна виділити лише три дійсно

багатофункціональних, зручних та підтримуваних середових – це IntelliJ IDEA WebStorm, Visual Studio Code та Nova.

Почнемо з останньої. Nova – це справді дуже потужний інструмент для веброзробника. Це середовище поєднує в собі всі аспекти роботи розробника вебзастосунків. Воно дає доступ до таких інструментів як: редактор коду, FTP клієнт для завантаження файлів з та на сервер, вбудований браузер, SSH термінал та інструменти для прямої роботи з базою даних [10].

Все вищезгадане можна віднести до переваг цього інструменту. До недоліків же те, що через настільки широкий функціонал, та й ще з можливістю додаткового розширення, поріг входження для розробників-початківців чи просто нових користувачів є досить високим. Також потенціальних користувачів може відлякнути висока ціна цього програмного забезпечення та обмеження у операційній системі – це може бути лише Mac OS [10].

Ще одними представником сучасних мультифункціональних IDE є WebStorm. Його функціонал у багато чому подібний до вищезгаданого Nova. Він також вільно працює з мовами JavaScript, HTML, CSS, полегшує розгортання більшості сучасних JavaScript фреймворків, таких як Node.js, ReactJS, Angular, Vue.js тощо. Так само як і Nova він має вбудований браузер, засоби для роботи з БД напряду, підтримує інтеграцію з репозиторіями Git, має вбудовану консоль. WebStorm розповсюджується лише за платною моделлю, а безоплатний випробувальний термін обмежений 30 днями [11].

До недоліків цього середовища розробки можна віднести необхідність оформлення підписки, щоб мати можливість використовувати цей програмний продукт, а також високий поріг входження через дуже широкий функціонал [11].

До переваг у порівнянні з Nova можна віднести сумісність WebStorm з основними операційними системами – Windows, Linux, MacOS [11].

Visual Studio Code – це легкий оптимізований редактор коду, що дозволяє розробнику працювати з більшістю мов та форматів програмування

таких як C++, C#, CSS, Dart, Dockerfile, F#, Go, HTML, Java, JavaScript, JSON, PHP, Python, R, Rust, SCSS, T-SQL, TypeScript [12].

Visual Studio Code на відміну від Nova та WebStorm не має такого широкого передвстановленого функціоналу, але це середовище розробки має відкритий вихідний код та велику бібліотеку додаткових модулів (плагінів). Це дозволяє мати на старті легку та швидку систему, яку кожен користувач може легко конфігурувати під себе за необхідності [12].

Для задоволення базових потреб ця IDE має інструменти для відлагодження коду, автодоповнення, інтеграцію з системами контролю версій тощо [12].

Головними перевагами Visual Studio Code є те, що вона розповсюджується на безоплатній основі та має широке коло користувачів, а отже і потужну підтримку. Також можна відзначити її кроссплатформенність [12].

До недоліків можна віднести важкість першого налаштування.

На основі досліджень було сформовано таблицю 2.1 з порівняльними характеристиками середовищ розробки, що досліджувались.

Таблиця 2.1 – Порівняння середовищ розробки

Критерії порівняння	Visual Studio Code [12]	WebStorm [11]	Nova [10]
Кроссплатформенність	+	+	—
Оплатність	—	+	+
Автодоповнення	+	+	+
Інтеграція з Git	+	+	+
Підтримка сторонніх плагінів	+	—	+
Аналіз коду	+	+	+
Рефакторинг коду	+	+	+
Вбудована консоль	+	+	+
UX/UI	+	±	+
Налаштовуваність	+	±	+

У результаті порівняння вищеназваних IDE для подальшого використання при розробці було обрано Visual Studio Code. Вибір на користь саме нього обумовлено його доступністю, легкістю та варіативністю.

2.1.2 Вибір мови програмування та архітектури проєкту

Ще одним з визначальних моментів проєктування є обрання власне мови, на якій буде вестися розробка. Так як проєктується онлайн-сервісний застосунок вибір мов скорочується, але все ж залишається досить широким. Оберемо декілька з найбільш популярних, визначимо переваги та недоліки кожної, порівняємо їх критичність та зробимо остаточний вибір.

Почнемо з Python. Це дуже гнучка, динамічна та потужна мова програмування й дуже багато розробників надають перевагу саме йому. Багато відомих вебзастосунків було реалізованою мовою Python. Зокрема Instagram, Spotify та чимало інших. Хоча слід зазначити, що Python все ж таки найкраще підходить для серверних рішень. Його використання на «клієнті» менш популярне [13].

Його відносна популярність зумовлюється легкістю використання при розробці. Ця легкість пояснюється дуже простим синтаксисом мови, широкими можливостями з візуалізації даних, широкою сумісністю з іншими мовами програмування та підтримкою асинхронного програмування [13].

Але у всього є недоліки і Python не виняток. До них можна віднести проблеми з потоками, саме Global Interpreter Lock (GIL) дозволяє виконання лише одного потоку в окремий момент часу. Через те, що Python мова, що інтерпретується, а не компілюється існують деякі обмеження в швидкості його виконання, що також є досить суттєвим недоліком [13].

Отже підсумуємо плюси та мінуси використання мови Python.

Плюси:

- простота використання;
- легкість вивчення;

- простота візуалізації даних;
- зрозумілість;
- можливість асинхронного програмування;
- сумісність з іншими мовами.

Мінуси:

- обмеження швидкості роботи;
- проблеми з потоками.

Наступний на черзі JavaScript, - досить стара мова програмування, яка на початку свого існування була дуже обмеженою, фактично виконуючи роль мови сценаріїв для вебсторінок. Але з плином часу вона розвивалась і перетворилась на дуже потужний і, що не менш важливо, сучасний інструмент. Особливістю цієї мови є те, що вона виконується на стороні «клієнта» і продуктивність жить саме від нього. Та на сьогоднішній день більшість сучасних девайсів достатньо потужні для того щоб не виникало якихось проблем з швидкодією. Ще більше підвищується швидкість роботи за рахунок використання можливостей постійного підключення до сервера, функцій зворотного виклику та двигуна V8 [14].

Ще однією безперечною перевагою JavaScript є відсутність необхідності додаткового налаштування, бо більшість сучасних браузерів вже мають її підтримку [14].

Але найбільш важливою перевагою JavaScript можна назвати універсальність цієї мови. Вона однаково добре підходить для розробки як клієнтських, так і серверних застосунків. А допоможуть в цьому багато існуючих фреймворків на кшталт Angular, React, Vue, Express тощо [14].

Якщо стисло, то переваги JavaScript такі:

- відсутність необхідності окремого встановлення, через підтримку більшістю сучасних браузерів;
- універсальність;
- швидкість роботи;

- широка популярність та офіційна й неофіційна підтримка;
- велика кількість різноманітних фреймворків, що значно спрощує розробку.

З недоліків можна відзначити:

- складність відлагодження коду;
- залежність швидкодії від клієнтського пристрою.

І насамкінець розглянемо добре всім відому мову PHP. Це мова програмування, що виконується на стороні сервера. Вона слугує для створення динамічних та інтерактивних вебзастосунків [15].

Перелічимо переваги та недоліки цієї мови.

Переваги:

- відкритий вихідний код;
- кросплатформеність;
- широке поширення та підтримка;
- підтримка багатьох баз даних;
- підтримується багатьма серверами;
- велика кількість розширень та бібліотек.

Недоліки:

- слабкі засоби для роботи з виключеннями;
- глобальні параметри конфігурації впливають на базовий синтаксис мови;
- можливі проблеми з безпекою.

Доповнюючи все вищесказане, можна зазначити, що всі мови, які розглядалися, з часом продовжують розвиватися та йти у ногу з часом, покращуючись та виправляючи власні недоліки.

Для наочного порівняння та прийняття зваженого рішення було створено порівняльну таблицю 2.2.

Таблиця 2.2 – Порівняння мов програмування

Характеристики	Python [13]	JavaScript [14]	PHP [15]
Призначена для веброзробки	-	+	+
Швидкість виконання	-	+	+
Документованість	+	+	+
Простота	+	+	+
Розширюваність	+	+	—
Різноманіття вбудованих методів	+	+	+
Асинхронність	+	+	+
Можливість інтеграції та взаємодії з іншими мовами	++	+	+
Кросплатформеність	+	+	+

Враховуючи результати порівняння та вимоги проєктованого програмного продукту було вирішено обрати мову JavaScript. Це обумовлено можливістю розробки як серверної, так і клієнтської частини одною мовою, пристосованістю до роботи зі зручним форматом даних JSON, простою розширюваністю та наявністю великої кількості бібліотек для реалізації необхідного функціоналу.

Виходячи з простої реалізації на такому поєднанні клієнта та сервера для застосунку було вирішено обрати архітектуру REST.

Але для впевненості у прийнятому рішенні проведемо порівняння даної архітектури з тим, з чим порівнюють її завжди – SOAP.

Спершу наведу деякі основні принципи REST.

REST архітектура – це завжди клієнт-серверний застосунок працюючий по протоколу HTTP, реалізації і процеси максимально відокремлені один від одного. З цього логічно витікає ще один принцип даної архітектури – кожен запит до сервера повинен представляти свій стан. Сервер повинен мати змогу вільно розуміти і працювати з запитом без жодного використання стану сервера чи сесії. Всі стани повинні зберігатися на клієнті [16].

Головною же перевагою архітектури REST полягає в тому, що всі взаємодії клієнта з сервером проходить через єдиний уніфікований інтерфейс.

Це дозволяє легко модифікувати серверну чи клієнтську частину незалежно одна від одної чи, навіть, працювати серверу одразу з декількома клієнтами одночасно через єдиний інтерфейс [16].

В свою чергу SOAP складніше реалізувати, він потребує більше пропускну здатності та ресурсів. Запит SOAP довше оброблюється і не використовує кешування, на відміну від REST. Натомість SOAP має підвищений рівень безпеки, бо має свій власний безпековий протокол. Також варто вибрати її у разі необхідності реалізації у застосунку транзакцій [17].

Отже резюмуючи наявну інформацію можна обґрунтувати вибір на користь REST швидкістю виконання, використання зрозумілого протоколу HTTP та легку масштабованість обох компонентів системи.

2.1.3 Вибір бази даних

Одним із важливих етапів проєктування є вибір бази даних. На сьогоднішній день існує два найбільш розповсюджених типи баз даних: реляційні та нереляційні, вони ж документоорієнтовані. Ці два типи реалізують кардинально протилежні підходи та принципи по побудові БД [18].

Реляційні БД зберігають структуровані дані, що представляють собою об'єкти реального світу, об'єднані в таблиці, формат яких заданий на етапі створення [18].

Характерними особливостями реляційних баз даних наприклад є використання мови запитів SQL, орієнтованість на збереження цілісності даних, легка реалізація транзакцій, відповідність вимогам ACID.

SQL (Structured Query Language) – це основний спосіб роботи з реляційними базами даних. Ця мова є стандартом Національного інститута стандартів США і підтримується більшістю ядер сучасних реляційних БД. SQL використовується для роботи з усіма аспектами таких БД.

Під цілісністю даних розуміють повноту, точність та однорідність даних. Для її збереження реляційні БД використовують різноманітні

інструменти, такі як первинні ключі, зовнішні ключі, обмеження унікальності. Такі заходи дозволяють гарантувати надійність даних.

Транзакції – це послідовне виконання декількох команд SQL, об'єднаних в одну логічну одиницю. Тобто вони мають або всі завершитися успішно, або при провалі хоча б однієї «відкатитися» до первинного стану.

Принципи ACID (Atomicity, Consistency, Isolation, Durability) є запорукою збереження цілісності даних при транзакціях.

Документоорієнтовані ж бази даних зберігають дані у вигляді деякої ієрархічної структури даних. Переважно дані зберігаються у JSON подібних документах для кожної окремої цілої сутності [18].

Використання нереляційної бази даних є виправданим якщо від БД вимагається гнучкість, масштабованість, висока продуктивність, широкі функціональні можливості [18]-[19].

Зазвичай такі БД пропонують гнучкі схеми даних, які підходять для частково структурованих або неструктурованих даних, та надають можливості для поетапної реалізації [18]-[19].

Для реалізації онлайн-сервісного застосунка, що проєктується було вибрано нереляційну базу даних. Це було обумовлено використанням в проєкті невеликої кількості даних з динамічними форматами, швидкістю роботи БД такого типу та зручністю роботи з документами формату JSON.

Як представника цього типу баз даних було обрано MongoDB.

Основними можливостями СКБД MongoDB є:

- JSON подібна схема даних;
- підтримка індексів;
- ефективне зберігання бінарних даних;
- легка масштабованість;
- стійкість до відмов;
- журналювання операцій [19].

Поміж усіх переваг які має ця СКБД, вона пропонує ще й цікаве хмарне рішення [19]. Тому, опираючись на всі вище названі факти, було прийнято рішення використовувати саме MongoDB.

2.2 Структурна схема програмного продукту

Проектований застосунок складається з клієнтської частини, що забезпечує введення, первинну валідацію та відображення даних, відповідає за всю взаємодію користувача із програмним продуктом. Та серверної частини, яка забезпечує повторну валідацію, обробку та збереження даних у БД.

Застосунок побудований за архітектурою REST, що говорить проте, що взаємодія між клієнтською та серверною частинами відбувається по протоколу HTTP.

Дані між «клієнтом» та «сервером» передаються у фалах типу JSON. Вибір саме цього типу файлів був зумовлений легкістю маніпуляцій з ними за допомогою обраної мови програмування JavaScript та використання саме цього формату для збереження документів у нереляційній базі даних MongoDB.

Як клієнтська, так і серверна частини були розроблені на мові програмування JavaScript.

Для реалізації клієнтської частини застосунка було використано фреймворк Vuejs. Це сучасний фреймворк, розроблений для створення інтерфейсів користувача. Його перевагою є можливість використання однофайлових компонентів Vue – це формат файлів, що дозволяю зібрати шаблон, логіку та стилі компонента. Це значно спрощує розробку та повторене використання створених компонентів, оскільки в своїй більшості вони виходять повністю незалежними один від одного [20].

Серверна частина була реалізована за допомогою фреймворка Express для застосунків Node.js.

Node.js – це деяке подієве оточення для виконання JavaScript, створене для розробки масштабованих вебзастосунків [21].

Фреймворк Express є надбудовою над класичним Node.js та мінімалістичним та гнучким інструментом для швидкої побудови серверного API [22].

Для зберігання численних даних, необхідних для коректного функціонування застосунку було обрано документоорієнтовану базу даних MongoDB. Її перевагою можливість хмарного розміщення БД та зберігання даних у JSON подібному форматі. Отже кожен запис у такій БД є окремим документом формату, подібного до JSON, що дозволяє зберігати дані про об'єкт як окрему повну сутність, не розбиваючи її на декілька окремих. Це та індексний доступ до екземплярів сутностей дозволяє значно зменшити кількість операцій та часу для отримання необхідних даних.

Ще однією значною перевагою БД цього типу є можливість динамічного визначення параметрів екземплярів сутностей. Тобто різні екземпляри однієї сутності можуть містити різний набір параметрів.

При використанні проєктованого онлайн-сервісного застосунку незареєстрований користувач має доступ лише до базової інформації, що розміщена на головній сторінці вебзастосунку. Для використання всіх можливостей системи користувачу необхідно зареєструватися.

Після проходження реєстрації користувач може отримати доступ до особистого кабінету, де зосереджено основний функціонал системи.

Окремим пунктом йде адміністратор, який має розширені можливості у порівнянні зі звичайним користувачем. Наприклад видаляти звичайних користувачів, створювати оголошення, звіти тощо.

Нижче на рисунку 2.1 наведена загальна схема взаємодії користувачів різних ролей проєктованою системою.

Щоб проілюструвати клієнт-серверну взаємодію на рисунку 2.2 приведено спрощену схему роботи архітектури «клієнт - сервер».

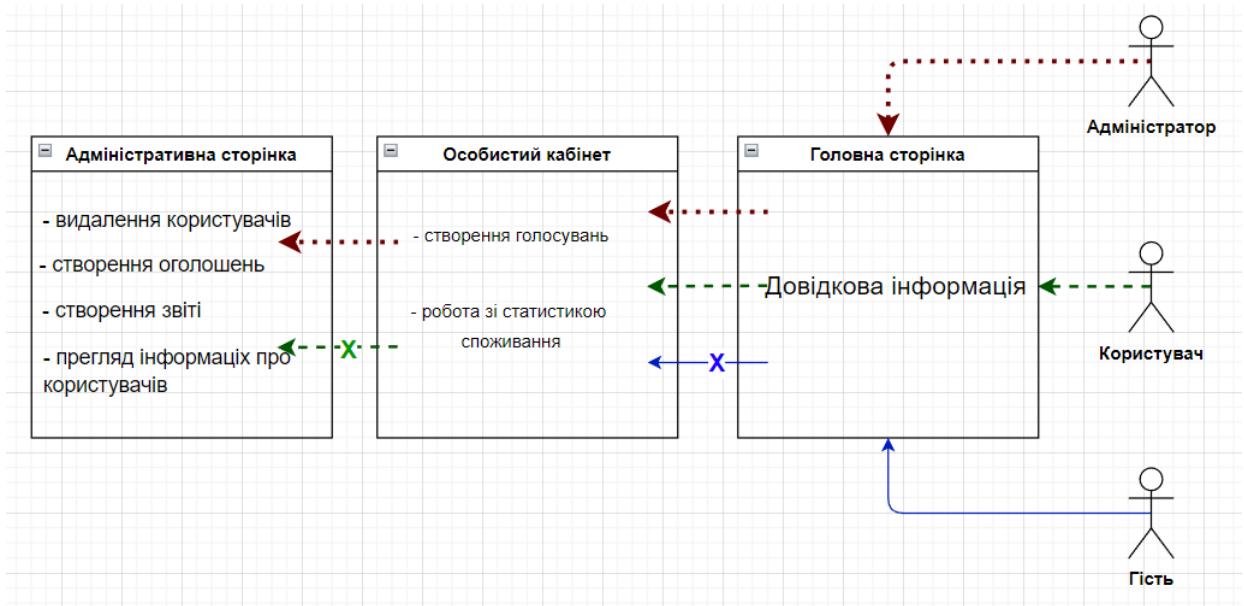


Рисунок 2.1 – Схема взаємодії користувачів з системою

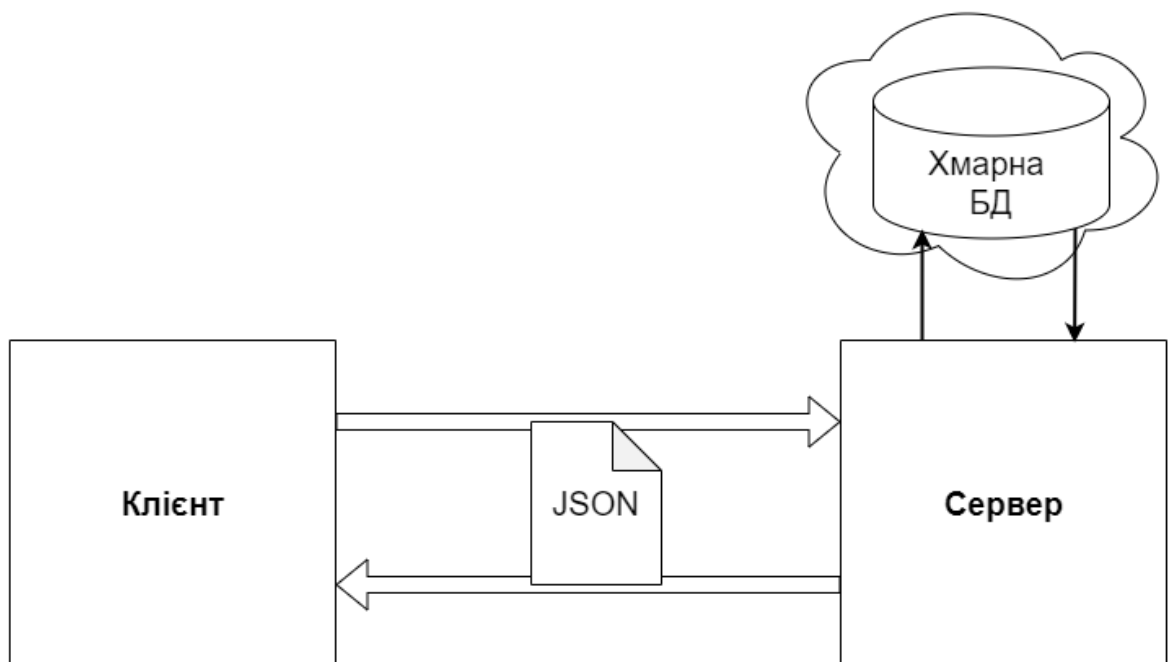


Рисунок 2.2 – Схема клієнт-серверної взаємодії

2.3 Висновки за розділом 2

У другому розділі були проаналізовані основні можливі концепції реалізації проєктованого онлайн-сервісного застосунку й були прийняті рішення щодо них.

А саме було порівняно з аналогами та обрано архітектуру застосунку, мову програмування, якою він буде написаний та спосіб зберігання даних.

Також було сформульовані положення щодо структурної компоновки майбутнього застосунку.

3 РОЗРОБКА ПРОГРАМИ

3.1 Структурна схема застосунку

Метою роботи є створення онлайн-сервісного застосунку для співвласників багатоквартирного будинку. Тобто проєктований застосунок має на меті забезпечення деяких потреб цієї категорії людей. Також, зазвичай, в об'єднаннях співвласників багатоквартирних будинків є певна особа, або група осіб, що виконують роль адміністрації, виконуючи функції з керування цим об'єднанням. Тож проєктований застосунок має включати в себе певний виділений функціонал для керуючих ОСББ. Також гості сайту не мають отримувати доступ до функціоналу, що призначений виключно для учасників об'єднання.

Виходячи з перерахованих загальних вимог було визначено структурну схему клієнтської частини застосунка. Вона представлена на рисунку 3.1.



Рисунок 3.1 - Структурна схема клієнтської частини застосунка

3.2 Метод реалізації автентифікації користувачів

Як видно з вищенаведеної схеми доступ до головних функції системи дозволено лише зареєстрованим та авторизованим користувачам. Для надання та відслідковування доступу в застосунку використано принцип автентифікації користувачів на основі двох так званих «токенів».

Під «токеном» розуміють певний зашифрований ключ, який додатково містить в собі певну інформацію. При використанні такого типу авторизації застосовується пара токенів – access та refresh токени.

Перед поясненням що таке власне access та refresh токени треба зробити примітку, що для токенів можна налаштовувати час їх життя. Після проходження того періоду часу, який було зазначено при створенні, токен «псується» й перестає проходити перевірку на валідність. Власне на цьому принципі й побудовано двотокенну автентифікацію.

Access токен має короткий термін життя та призначений для багаторазового використання. Він може містити в собі різноманітну інформацію в зашифрованому вигляді, і отримати її можна лише знаючи «секретний ключ» -деяку довільну слово, фразу чи набір символів, що було використано при створенні токена. У проєктованому додатку цей токен має в собі інформацію про користувача – це дозволяє зазначати менше параметрів при запиті на сервер та робити менше запитів до БД.

Refresh токен має, навпаки, довгий термін життя та є одноразовим. Він зазвичай зберігається в БД для кожного авторизованого користувача, і при використанні має бути перезаписаний на новий.

Ця схема автентифікації користувача працює наступним чином – спочатку користувач авторизується в застосунку вводячи логін та пароль, якщо авторизація успішна, то у відповідь він отримує пару токенів. Тепер при кожному запиті на сервер користувач відправляє разом із запитом access токен і він перевіряється на валідність на сервері. Якщо токен валідний – користувач автентифікований і запит буде виконано. Якщо – ні, то сервер повертає

помилку. Отримавши помилку автентифікації, «клієнт» відправляє на сервер автентифікації запит на оновлення tokenів, прикріпивши до нього другий token – refresh. На сервері він порівнюється з refresh tokenом цього користувача в базі та перевіряється на валідність. І, якщо всі перевірки пройшли успішно, сервер повертає нову пару tokenів «клієнту», а refresh token в базі оновлює. У разі виникнення помилок на етапі перевірок користувач буде вважатись неавторизованим і йому необхідно буде знову вводити логін та пароль. Як працює ця система проілюстровано на рисунку 3.2.

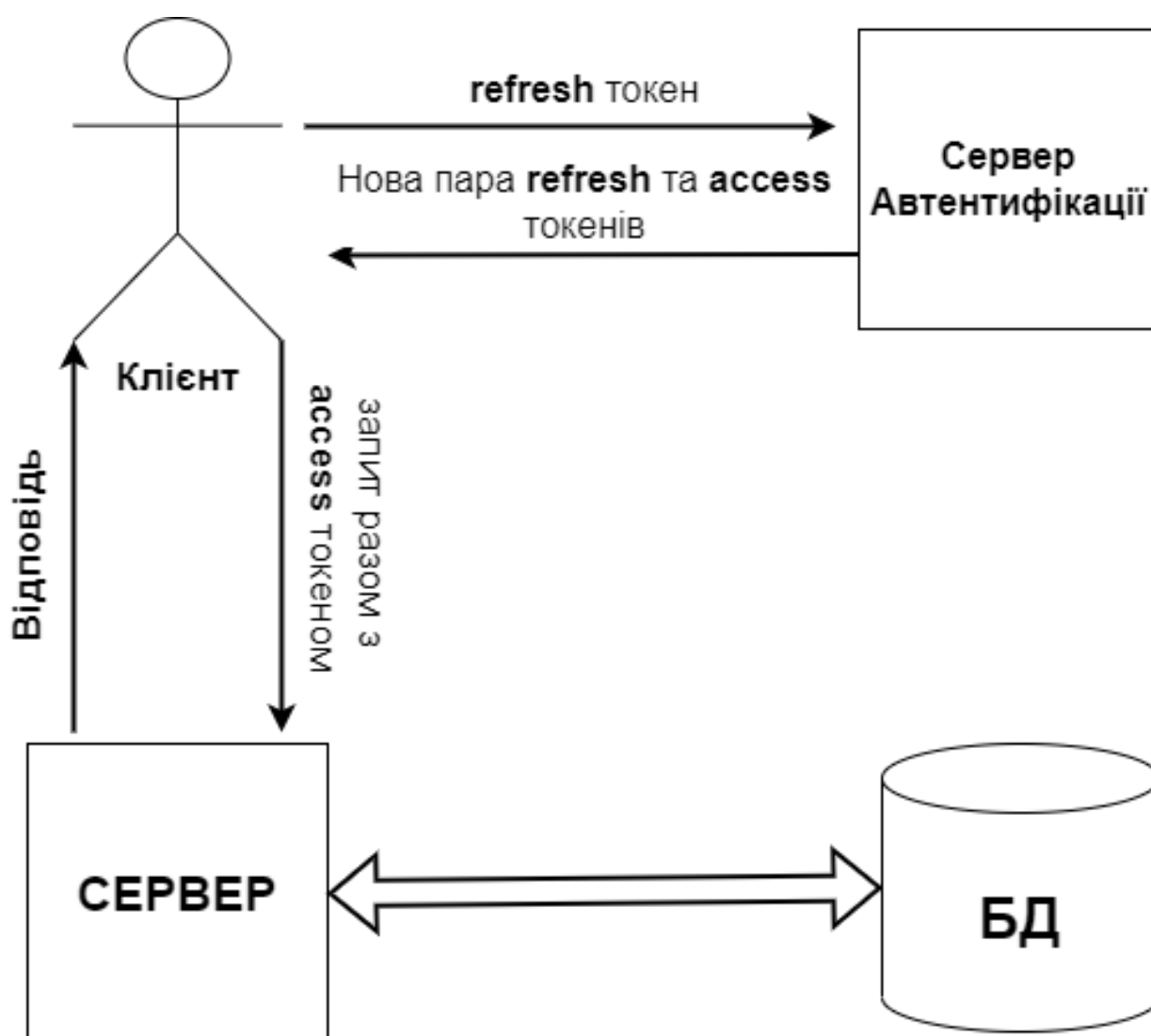


Рисунок 3.2 – Схема роботи двотокенної автентифікації

Що повністю завершити тему реєстрації та авторизації складемо блок-схеми цих процесів. Вони будуть зображені на рисунках 3.3 та 3.4 відповідно.



Рисунок 3.3 – Блок-схема процесу реєстрації



Рисунок 3.4 –Блок-схема процесу авторизації

3.3 Способи представлення даних

Після вдалого проходження процесу реєстрації та\або авторизації користувач отримує доступ до особистого кабінету, де має наступні можливості:

- перегляд активних оголошень від адміністрації;
- перегляд статистики споживань енергоресурсів;
- додавання даних для формування статистики;
- створення та перегляд активних голосувань.

На основі введених даних формується певна статистика, що в свою чергу трансформується в зручні для перегляду та аналізу діаграми та графіки. Для детального ознайомлення можна використати також й табличне представлення даних.

Для представників адміністрації додатково доступні наступні функціональні можливості:

- створення звітів;
- перегляд списку мешканців;
- додавання мешканців у список;
- видалення мешканців зі списку;
- створення оголошень.

Більшість елементів користувацького інтерфейсу було створено за допомогою бібліотеки компонентів користувацького інтерфейсу ElementPlus. Ця бібліотека надає доступ до гнучких, адаптивних та реактивних компонентів і добре сумісна з обраним фреймворком Vue.js найновішої, третьої версії.

Для реалізації компонентів графіків та діаграм було використано бібліотеку Vue-Chart.js. Це адаптована до використання з фреймворком Vue.js бібліотека Chart.js. Вона дозволяє створювати різноманітні відображення даних за допомогою численних видів налаштовуваних графіків та діаграм.

3.4 Висновок за розділом 3

В даному розділі були розглянуті основні можливості та шляхи їх реалізації, способи представлення даних. Також було розглянуто заходи щодо забезпечення конфіденційності користувачів, процеси реєстрації, авторизації та автентифікації.

4 КЕРІВНИЦТВО КОРИСТУВАЧА

4.1 Призначення та умови використання

Спроектований програмний продукт являє собою онлайн-сервісний застосунок призначений для інформування та полегшення взаємодії співвласників багатоквартирного будинку.

Основними мінімальними вимогами для функціонування вищезгаданого застосунку є:

- операційна система Microsoft Windows 7 і вище, або Linux з ядром системи версії Kernel 11 та вище;
- центральний процесор з тактовою чистотою 2.0 ГГц чи вище та кількістю ядер не менше двох;
- об'єм оперативної пам'яті вище за 3 Гб;
- об'єм постійної пам'яті вище за 1 Гб;
- доступ до мережі інтернет, з пропускнуою здатністю вище за 20 Мб\с.

4.2 Характеристики програмного продукту

Онлайн-сервісний застосунок складається з клієнтської та серверної частин, а також з БД.

Клієнтська частина відповідає за реалізацію графічного інтерфейсу, взаємодію з користувачем, введення, виведення та вторинну обробку даних. Ця частина програмного продукту виконана із застосуванням фреймворка Vue.js мови JavaScript. Він дозволяє використовувати багатокomпонентну структуру додатка, де окремий файл з розширенням .vue містить в собі все необхідне для функціонування компонента.

Серверна частина відповідає за роботу з даними – фільтрацію, валідацію, запити в БД, надсилання даних на клієнт та прийом їх з клієнта. Дана частина реалізована з використанням оточення Node.js для мови

програмування JavaScript та його фреймворка Express. Така реалізація дозволяє відносно легко налаштувати клієнт-серверну взаємодію за протоколом HTTP та підключення до сторонньої бази даних.

У якості бази даних було обрано документоорієнтовану СКБД MongoDB. Цей тип БД характеризується масштабованістю, динамічними параметрами екземплярів сутностей та зручним форматом документів, в яких ці сутності зберігаються.

4.3 Звернення до програмного застосунку

Щоб отримати доступ до стабільної загальнодоступної версії сайту необхідно перейти за посиланням у браузері.

Для отримання доступу до вихідного коду застосунку, що розроблюється необхідно за допомогою утиліти системи контролю версій Git підключитись до віддаленого репозиторію та клонувати проєкт на свою локальну станцію. Після чого, викликавши системну консоль в директорії проєкту на локальній станції, встановити пакети, яких бракує за допомогою команди «npm i». Дочекавшись завершення встановлювання всіх необхідних пакетів, можна почати розробку запустивши виконання проєкту на локальній машині командою «npm run dev». Зробити це необхідно як для клієнтської частини, так і для серверної. Це дозволить системі виконувати автоматичну перекомпіляцію коду після кожної збереженої зміни в файлах застосунку. Отримати доступ до результатів виконання клієнтської частини програмного застосунку на локальній станції можна за адресою «http://localhost:3000» в браузері. Запити до сервера будуть доступні за локальною адресою «http://localhost:5000/api».

4.4 Вхідні та вихідні дані

Як вхідні дані онлайн-сервісного застосунку для співвласників багатоквартирного будинку можна розглядати текстові дані, які вводяться у різноманітні форми, на кшталт форми реєстрації нового користувача, що зображена на рисунку 4.1. Також вхідними даними можна вважати будь-які файли, які користувачі завантажують у певних формах.

The image shows a registration form titled "Реєстрація" (Registration) with a close button in the top right corner. The form contains several input fields and a role selection button:

- * Я** (I): A button with two options, "Власник" (Owner) and "Орендар" (Tenant).
- * Квартира №** (Apartment No.): A numeric input field with a minus sign, the value "0", and a plus sign.
- * Ім'я** (Name): A text input field with the placeholder "Ім'я".
- По-батькові** (Patronymic): A text input field with the placeholder "По-батькові".
- * Прізвище** (Surname): A text input field with the placeholder "Прізвище".
- * Email**: A text input field with the placeholder "example@email.com".
- Телефон** (Phone): A text input field with the placeholder "+38(000)000-0000".
- * Пароль** (Password): A text input field with the placeholder "Пароль".
- * Підтвердження** (Confirmation): A text input field with the placeholder "Підтвердження паролю".

At the bottom right of the form are two buttons: "Відправити" (Send) in blue and "Очистити" (Clear) in white.

Рисунок 4.1 – Форма реєстрації користувача

Вихідні дані представлені текстовою інформацією на вебсторінках, графіками, діаграмами, таблицями тощо. Приклади деяких з них наведено на рисунках 4.2, 4.3, 4.4.

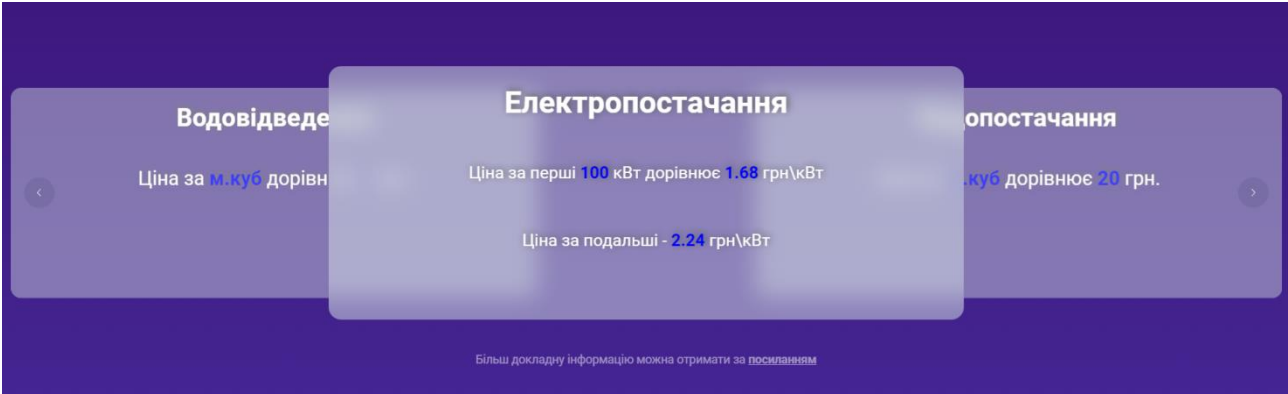


Рисунок 4.2 – Відомості щодо тарифів на енергоносії

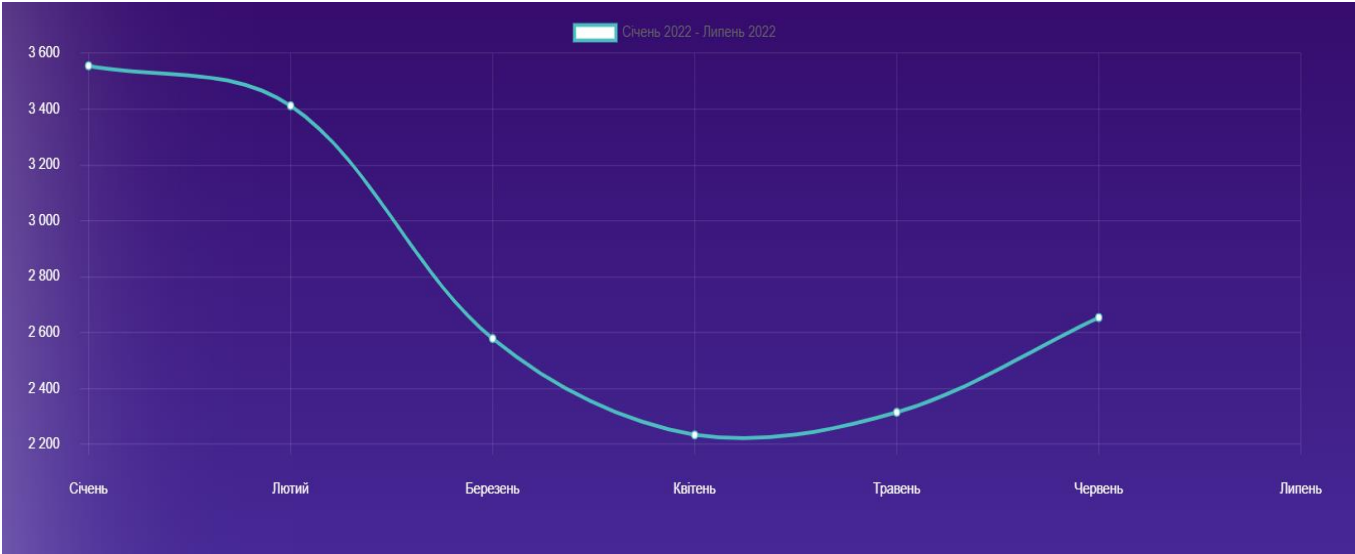


Рисунок 4.3 – Графік витрачених коштів

Кв.	Ім'я	Email	Статус	Телефон	Дії
1	Карамболь Олена Василівна	ex@ecx.com	Власник	+380971111111	Редагувати Видалити
2	Щериця Анатолій Андрійович	owner@ecx.com	Власник	+380972222222	Редагувати Видалити
3	Живченко Світлана Миколаївна	ex9988@ecx.com	Власник	+380973333333	Редагувати Видалити
4	Андерсон Реджинальд Томас	асрох@ecx.com	Власник	+380974444444	Редагувати Видалити
5	Занечуй Віктор Романович	gist_body@ecx.com	Орендар	+380975555555	Редагувати Видалити

Рисунок 4.4 – Таблиця «Будинкової книги»

4.5 Повідомлення

Для забезпечення реактивності системи на дії користувача, а також для реакцій на виникаючі помилки застосунок може генерувати деякі повідомлення. Список таких повідомлень наведено в таблиці 4.1.

Таблиця 4.1 – Повідомлення застосунка

Повідомлення	Значення
<code>{назва_поля}</code> не може бути порожнім	При заповненні деяких форм користувач може зіткнутися з полями, що є обов'язковими для заповнення. Така помилка виникає, якщо обов'язкове поле залишилось порожнім.
Користувача не знайдено	Виникає якщо користувач з такою поштою незареєстрований.
Неправильний пароль	Виникає якщо при спробі входу введено неправильний пароль
Користувач вже існує	Виникає при спробі реєстрації користувача з електронною поштою, яка вже кимось використовується.
Успіх!	Використовується при успішній реєстрації нового користувача.
Користувач не авторизований	Виникає у разі спроби доступу до ресурсів, що вимагають авторизації.

4.6 Призначення та умови виконання

Спроектований програмний продукт являє собою онлайн-сервісний застосунок призначений для інформування та полегшення взаємодії співвласників багатоквартирного будинку.

Для використання програмний застосунок вимагає наявності інтернет-браузера та стабільного з'єднання з мережею Інтернет.

4.7 Використання застосунку

Для початку роботи необхідно ввести адресу застосунка в браузері та дочекатися завантаження головної сторінки. Зовнішній вигляд першого розділу головної сторінки сайту проілюстровано на рисунку 4.5.

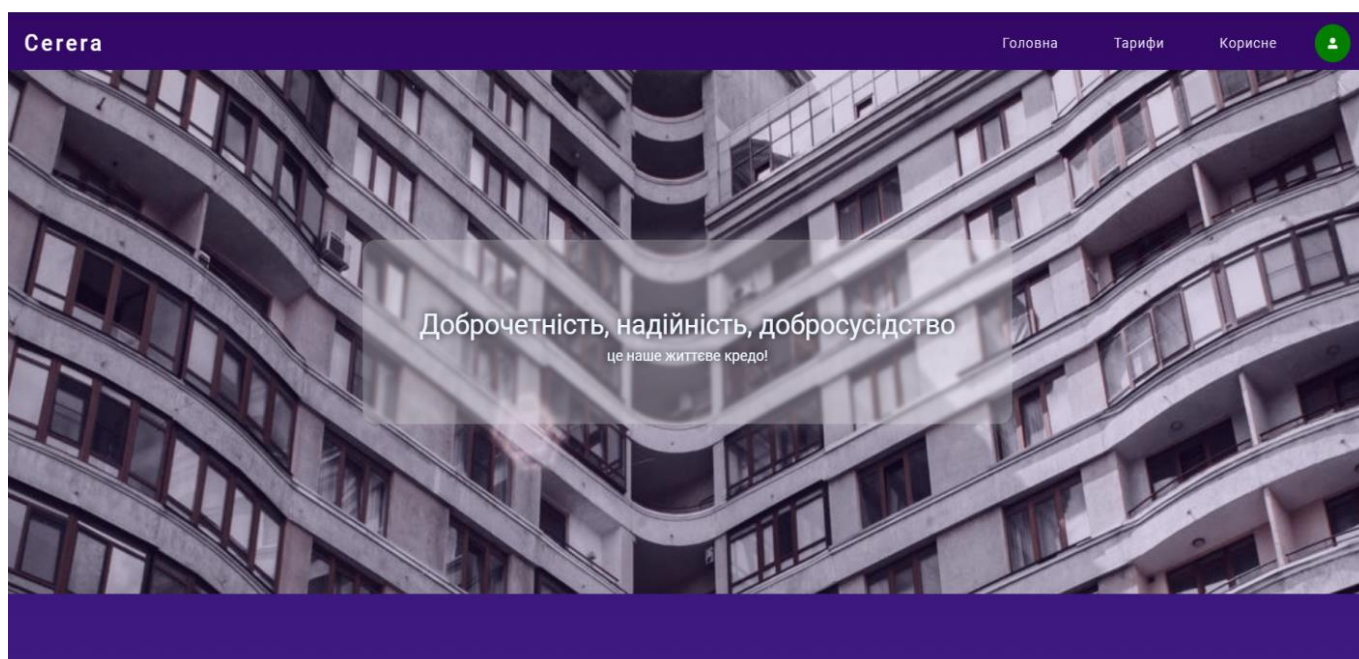


Рисунок 4.5 – Перший розділ головної сторінки сайту

Також на головній сторінці застосунку розміщено відомості про поточні тарифи на комунальні послуги. Зображення цього компонента розміщено на рисунку 4.2.

Далі на головній сторінці присутній розділ з корисною довідковою інформацією. Зокрема номери телефонів невідкладних служб, які реалізовані у вигляді карток, які можна натискати для швидкого набору номера. Вони зображені на рисунку 4.6.



Рисунок 4.6 – Картки невідкладних служб

Також у якості корисної допоміжної інформації наведені посилання на розділи міського інформаційного порталу та інших громадських ініціатив й телефонні номери міських служб. Ілюстрації цих розділів наведено на рисунках 4.7 та 4.8 відповідно.

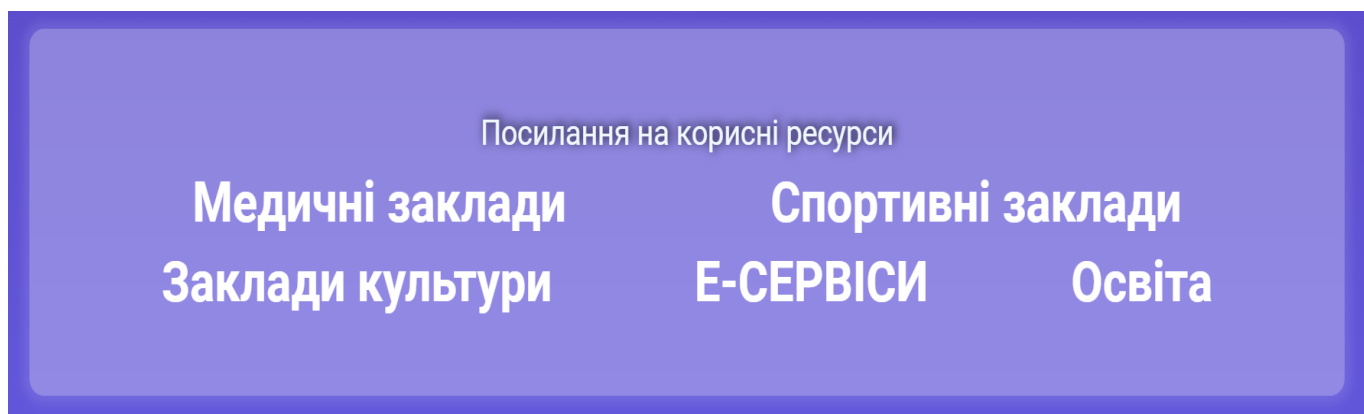


Рисунок 4.7 – «Хмарка» корисних посилань

Медицина, допомога громадянам та ветеринарна служба

(061) 787-60-97; (061) 236-51-52; (061) 769-76-91 – станції Швидкої медичної допомоги
 (061) 64-38-11 – Травмпункт при дитячій обл. лікарні
 (061) 34-51-10 – Консультативна служба (для самостійного надання медичної допомоги на дому)
 1503 – Центральна диспетчерська служба швидкої допомоги
 (061) 65-56-20 – Запорізька міська державна лікарня ветеринарної медицини

Електромережі та світло

(061) 233-13-15; (061) 65-01-05 – Аварійно-диспетчерська служба міськвітло
 (061) 286-36-33; (061) 233-01-42; (061) 286-36-52 – Міські електричні мережі

Газ

1581 – Ремонтна служба «Запоріжгазу»

Надзвичайна ситуація

(061) 62-63-71 – Черговий штаб запорізької цивільної оборони
 (061) 64-67-11; (061) 57-06-94 – Запорізька міська рятувально-водолазна служба
 (061) 233-13-74 – Обласна служба порятунку

Тепломережі, водопостачання та каналізації

(061) 764-21-78; (061) 764-22-95; (061) 236-61-81 – Міськводоканал

ЖКГ та будинки

1586 – Аварійна служба ліфтів
 (061) 224-69-02, (061) 224-02-94 – Управління комунального господарства Запорізької міської ради

Рисунок 4.8 - Телефонні номери міських служб

Для отримання доступу до особистого кабінету користувача необхідно зареєструватись чи, якщо вже маєте обліковий запис, увійти.

Для цього необхідно натиснути на відповідну піктограму в меню навігації сайту, після чого в меню, що з'явилося обрати необхідний пункт. Ця піктограма також відповідає за індикацію успішної автентифікації. Якщо користувача автентифіковано піктограма змінює свій колір на зелений. Цей процес, а також форми реєстрації та входу зображено на рисунках 4.9, 4.10 та 4.11.

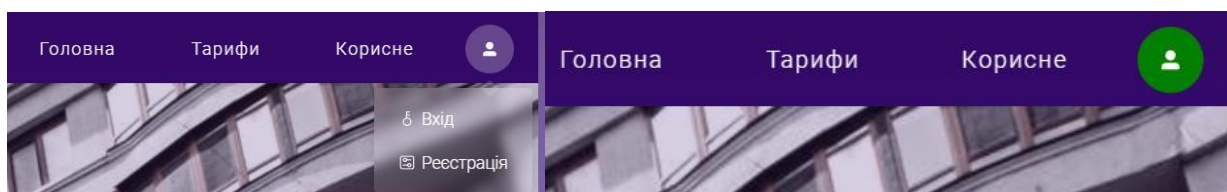


Рисунок 4.9 – Процес авторизації

Реєстрація

* Я ☐ Власник ☐ Орендар

* Квартира №

* Ім'я

По-батькові

* Прізвище

* Email

Телефон

* Пароль

* Підтвердження

Рисунок 4.10 – Форма реєстрації

Вхід

* Email

* Пароль

Рисунок 4.11 – Форма входу

У всіх формах застосунку поля позначені червоною зірочкою є обов'язковими для заповнення. У разі ігнорування таких полів або некоректного їх заповнення застосунок проінформує користувача про помилку. Також поля вводу всіх форм на сайті мають можливості інформування користувача про коректне їх заповнення шляхом відображення позначки

«пташки» на цих полях, швидкого очищення полів та демонстрації та маскування поля введення типу «пароль» натисканням спеціальної кнопки на полі. Все це проілюстровано на прикладі заповненої форми авторизації на рисунку 4.12.

The image shows a login form with two input fields. The first field is labeled '* Email' and contains the text 'це не електронна пошта' (this is not an electronic mail). Below this field is a red error message: 'Введіть коректну адресу електронної пошти' (Enter a correct electronic mail address). The second field is labeled '* Пароль' (Password) and contains three dots '...'. Both fields have a red border and a close button (X) in the top right corner. The password field also has an eye icon and a checkmark icon in the top right corner.

Рисунок 4.12 – Заповнена форма авторизації

Після успішної автентифікації у випадяючому списку, що зображено на рисунку 4.9, поля зміняться на «Кабінет» та «Вийти».

Для переходу до особистого кабінету користувача необхідно натиснути на пункт меню «Кабінет». Після переходу користувач потрапляє на сторінку з оголошеннями, які створила адміністрація. Це зроблено для інформування всіх хто заходить до кабінету. Приклад таких оголошень демонструє рисунок 4.13.

The image shows a section titled 'Оголошення' (Announcements) with a dark blue background. There are two announcement items, each in a light blue box. The first item is titled 'Про відключення водопостачання' (About water supply disconnection) and contains the text: 'На термін проведення ремонтних робіт з 15.06.22 по 22.07.22 буде відключено водопостачання в під'їздах з 1-го по 3-й' (For the duration of repair work from 15.06.22 to 22.07.22, water supply will be disconnected in apartments from 1st to 3rd floor). The second item is titled 'Про загальні збори' (About general assembly) and contains the text: 'Повідомляю мешканців будинку, що 5-го серпня 2022 року у підвалі 3-го під'їзду о 18:00 відбудуться загальні збори співвласників.' (I inform the residents of the building that on August 5, 2022, at 18:00, a general assembly of co-owners will be held in the basement of the 3rd apartment).

Рисунок 4.13 – Розділ з оголошеннями від адміністрації

Навігація розділами здійснюється за допомогою бічного меню. Якщо користувач має права адміністратора, то в меню навігації додається також

розділ для адміністрації. Меню навігації для звичайних користувачів та адміністрації приведено на рисунку 4.14.

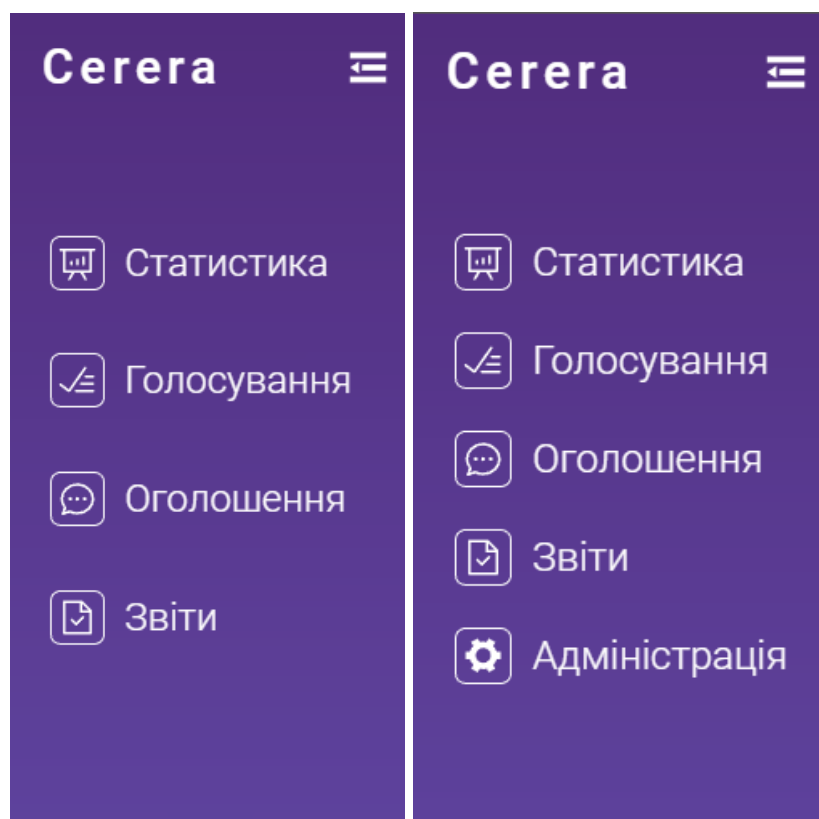


Рисунок 4.14 – Бічне меню навігації

Вся робота з даними у застосунку здійснюється за загальними принципами. Введення даних здійснюється за допомогою різноманітних форм, відображення даних реалізоване за допомогою «плиток», таблиці, а також у вигляді графіків. Приклади відображення даних наведені на рисунках 4.2-4.4.

Як приклад порядку дій розглянемо розділ з відображенням та створенням нових опитувань. Сторінка «Голосування» проілюстрована на рисунку 4.15.

На цій сторінці відображено список активних опитувань. Користувач може віддати свій один голос «За» чи «Проти» у кожному опитуванні шляхом натискання відповідної кнопки. Результати опитувань доступні у вигляді процентних шкал. Зелена демонструє відсоток голосів «За», червона – «Проти».

Для того щоб відобразити форму створення нового опитування необхідно натиснути на кнопку «+» у вигляді блакитного кола навпроти заголовка сторінки.

Після коректного заповнення форми, що проілюстрована на рисунку 4.16, та натискання кнопки «Додати» створиться нове опитування.

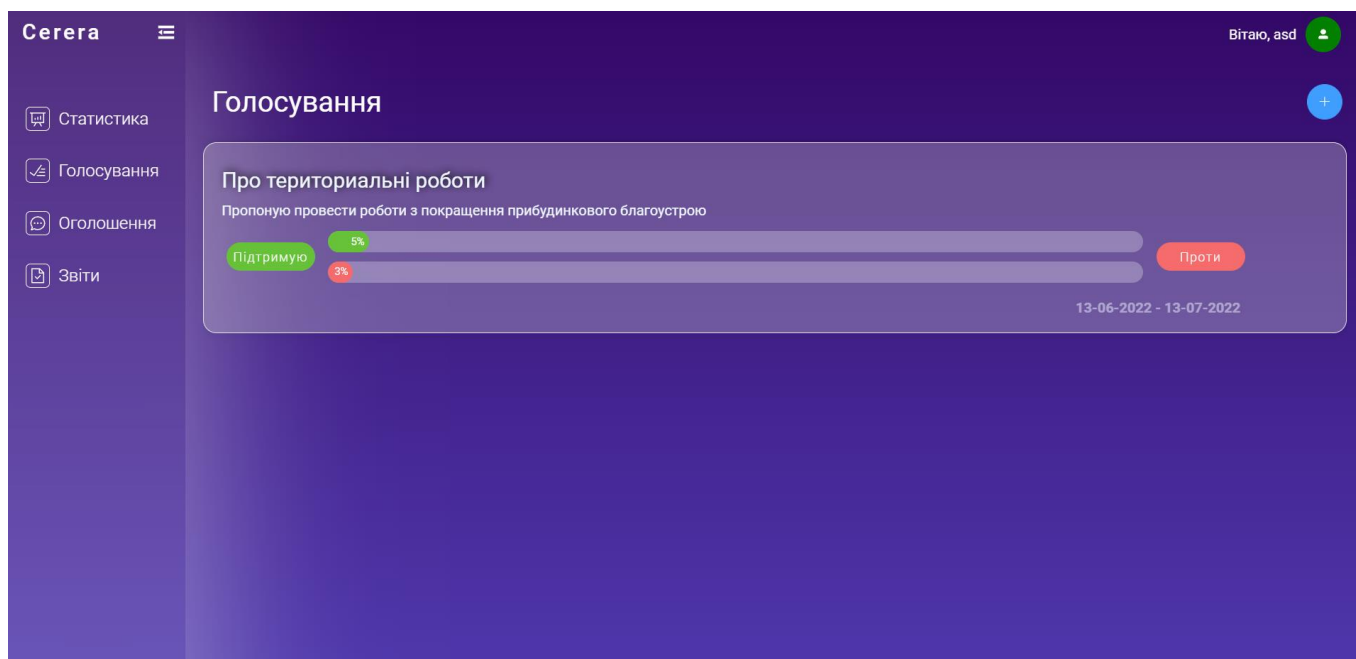


Рисунок 4.15 – Сторінка «Голосування»

Рисунок 4.16 – Форма створення нового опитування.

4.8 Висновки за розділом 4

В цьому розділі було сформульовано призначення програмного застосунку та мінімальні системні вимоги для його функціонування. Було наведено інструкції щодо отримання доступу до програмного продукту, його структури та особливостей розробки. Були визначені вхідні та вихідні дані. Надано перелік повідомлень, що використовує система для взаємодії з користувачем. Представлено загальні концепції роботи з програмним продуктом. Були описані основні елементи інтерфейсу застосунку.

ВИСНОВКИ

Для виконання дипломної було зроблено аналіз предметної області та досліджено існуючі проблеми та шляхи їх вирішення. Темою роботи є розробка онлайн-сервісного застосунку для співвласників багатоквартирного будинку. Що є актуальним, дивлячись на швидкі темпи цифровізації світу.

Також в рамках аналізу предметної області було проведено дослідження вже наявних на сьогодні рішень поставленої задачі. Було сформовано перелік переваг та недоліків для врахування їх при подальшому проектуванні.

За мету було поставлено спрощення комунікації між співвласниками та адміністрацією, прозорість рішень та використання спільних коштів.

На подальших етапах проектування було проведено аналіз мов програмування, типів баз даних та середовищ розробки. Для аналізованих мов програмування було складено порівняльну характеристику на основі виявлених переваг та недоліків. Виходячи з отриманих даних було обрано мову програмування JavaScript, як більш підходящу для виконуваних задач. Для використання у проєкті було обрано документоорієнтовану базу даних, а саме MongoDB. Як середовище розробки було обрано поширюване на безоплатній основі IDE – Visual Studio Code. В якості архітектури програмного продукту було обрано REST. За якою застосунок складається з двох майже незалежних частин – клієнтської та серверної. Ці частини взаємодіють між собою за допомогою протоколу HTTP.

Візуальну складову клієнтської частини було розроблено за допомогою фреймворку Vue.js.

Серверна частина реалізована за допомогою фреймворка Express.js та середовища Node.js.

Під кінець розробки було проведено тестування ПЗ. Його результати показали відсутність критичних помилок та високу відмовостійкість системи..

Як результат розробки було створено онлайн-сервісний застосунок для співвласників багатоквартирного будинку, що вирішує всі поставлені задачі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Закон України «Про об'єднання співвласників багатоквартирного будинку» № 2866-III, від 29.11.01 р. (зі змінами і доповненнями) [Електрон. ресурс] – Режим доступу: <http://zakon1.rada.gov.ua/laws/show/2866-14>

2. Закон України «Про внесення змін до деяких законодавчих актів щодо підвищення якості управління багатоквартирними будинками та вдосконалення діяльності об'єднань співвласників багатоквартирного будинку» № 2606, від 16.12.2019 р [Електрон. ресурс] – Режим доступу: http://search.ligazakon.ua/l_doc2.nsf/link1/JI00959A.html

3. Гаврилов С.А. Система «Экотел» – всевидящий глаз комплексной автоматизированной системы ЖКХ [Текст] / С.А. Гаврилов, Е.К. Жданович // Жилищное и коммунальное хозяйство. – 2008. – Вып. 10. – С. 27–31.

4. Інформаційне та програмне забезпечення САПР : тези доповіді конф. (15–18 жовтня 2003 року). – К., 2007. – С. 155–157

5. Information technology – Software product evaluation – Quality characteristics and guidelines for their use (ISO/IEC 9126) : ISO/IEC 9126. 1991. – [Effective from 1991-12-19] – Geneva : International Organization for Standardization, International Electrotechnical Commission, 1991. – (International Standard).

6. Андрейко В.М. Кваліметрія програмного забезпечення засобів вимірювань: монографія [Текст] /. В.М. Андрейко – Івано-Франківськ: Симфонія форте, 2016. – 120 с.

7. Сайт ОСББ «Бажана-12» [Електрон. ресурс] – Режим доступу: <https://bajana12.com/>

8. Сайт ОСББ «Зоряне» [Електрон. ресурс] – Режим доступу: <http://zoriane.kiev.ua/>

9. Сайт ОСББ «Фундуклеєвський» [Електрон. ресурс] – Режим доступу: <http://fundukleev.kiev.ua/>

10. Nova: native Mac code editor [Electronic resource]. – Режим доступу: <https://nova.app/>
11. WebStorm: The smartest JavaScript IDE by JetBrains [Electronic resource]. – Режим доступу: <https://www.jetbrains.com/ru-ru/webstorm/>
12. Visual Studio Code – Code Editing. Redefined [Electronic resource]. – Режим доступу: <https://code.visualstudio.com>
13. Welcome to Python.org [Electronic resource]. – Режим доступу: <https://www.python.org>
14. JavaScript | MDN [Electronic resource]. – Режим доступу: <https://developer.mozilla.org/ru/docs/Web/JavaScript>
15. PHP: Hypertext Preprocessor [Electronic resource]. – Режим доступу: <https://www.php.net>
16. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures: thesis ... doctor of philosophy / Fielding Roy Thomas. – Irvine, CA: University of California, 2000. 152 p.
17. Rational Software Architect [Electronic resource]. – Режим доступу: <https://www.ibm.com/docs/ru/rsas/7.5.0?topic=standards-soap>
18. Гарсія-Моліна Г. Системи баз даних. Повний курс. [Текст] / Г. Гарсія-Моліна, Дж. Ульман, Дж. Уідом – М.: Вільямс, 2003. – 1088 с.
19. MongoDB documentation [Electronic resource]. – Режим доступу: <https://www.mongodb.com/docs/>
20. Vue.js Framework [Електрон. ресурс]. – Режим доступу: <https://v3.ru.vuejs.org/ru/>
21. Node.js. Official documentation [Електрон. ресурс] – Режим доступу: <https://nodejs.org/uk/about/>
22. Express Framework [Електрон. ресурс]. – Режим доступу: <https://expressjs.com/ru/>

ДОДАТОК А
Технічне завдання

Вступ

В результаті роботи було розроблено онлайн-сервісний застосунок для співвласників багатоквартирного будинку, який вирішує більшість недоліків вже існуючих рішень, простий у використанні та адаптований до використання на портативних пристроях.

A.1 Мета розробки і її призначення

Об'єкт дослідження – об'єднання співвласників багатоквартирного будинку.

Предмет дослідження – методи побудови онлайн-сервісного застосунку для співвласників багатоквартирного будинку .

Мета роботи – розробка програмного продукту на тему «Онлайн-сервісний застосунок для співвласників багатоквартирного будинку»

Матеріали, методи та технічні засоби: інтернет-браузер Google Chrome, сервіс Google Scholar, Adobe Reader, IDE Visual Studio Code, Vue.js, Express.js.

Актуальність розробки обумовлена сучасними тенденціями до цифровізації всіх форм господарювання, ускладненість комунікації між співвласниками та адміністрацією в умовах карантинних чи інших обмежень та вимогам прозорості обігу спільних коштів.

Вищеназвані проблеми вирішені за допомогою створення інструментів дистанційного комунікування, голосувань, різного роду обліків даних тощо.

A.2 Стислий зміст основних теоретичних посилок до розробки на основі проведених пошуків

Об'єднання співвласників багатоквартирних будинків (ОСББ) – це відносно невеликі самостійні суб'єкти господарювання. Вони зазвичай не мають коштів на оплату готових масштабних сервісів з автоматизації

господарювання. А якщо б й мали, як правило, у таких об'єднань немає потреби у більшості функцій, які пропонує така система.

Через специфічність даної форми господарювання, яка характеризується демократичністю та деякою децентралізацією, розроблюваний застосунок має просувати прозорість управління та спрощувати колективне прийняття рішень щодо керуванням будинком.

Також інформаційний сервіс, що проєктується має на меті запропонувати можливість користувачам зручно відслідковувати витрати газу, тепла, електроенергії, води тощо. Як у об'ємах, так і в грошовому еквіваленті. Відображення інформації у вигляді таблиць та графіків робить її наочною і зрозумілою для будь-якого користувача. А фільтрація за датою дозволить слідкувати за динамікою в часі.

А.3 Підстави для розробки

Проєктування програмного продукту виконується на підставі завдання на дипломний проєкт на тему «Розробка онлайн-сервісного застосунку для співвласників багатоквартирного будинку», затверджене наказом по НУ «Запорізька політехніка».

А.4 Основні вимоги до програми, що розробляється, початковим даним і результату

А.4.1 Вимоги до функціональних характеристик

Функціональні вимоги були визначені такі:

- реєстрація мешканців;
- додавання адміністратором інформації щодо тарифів на комунальні послуги;
- додавання адміністратором звітів;
- додавання адміністратором оголошень чи повідомлень;

- додавання користувачем даних про спожиті комунальні послуги;
- сортування даних про спожиті раніше послуги та датою чи типом;
- постановка адміністратором питання на голосування;
- додавання користувачем пропозиції на голосування;
- можливість користувача віддати свій голос за пропозицію.

A.4.2 Вимоги до надійності

Незареєстровані користувачі мають змогу переглядати лише головну інформаційну сторінку ресурсу. Для доступу до звітів, повідомлень та особистого кабінету користувач має зареєструватись.

Застосунок має стабільно працювати у нормальних умовах та надійному з'єднанні з мережею Інтернет.

A.4.3 Умови експлуатації

Програмний застосунок не вимагає персоналу для обслуговування. Інтерфейс програмного продукту інтуїтивно зрозумілий та не вимагає спеціального навчання.

A.4.4 Вимоги до складу та параметрів технічних засобів

Основними мінімальними вимогами для функціонування вищезгаданого застосунку є:

- операційна система Microsoft Windows 7 і вище, або Linux з ядром системи версії Kernel 11 та вище;
- центральний процесор з тактовою чистотою 2.0 ГГц чи вище та кількістю ядер не менше двох;
- об'єм оперативної пам'яті вище за 3 Гб;
- об'єм постійної пам'яті вище за 1 Гб;

- доступ до мережі інтернет, з пропускною здатністю вище за 20 Мб\с.

A.4.6 Вимоги для транспортування та зберігання

Розроблений програмний продукт не вимагає особливих умов для транспортування та зберігання.

A.5 Вимоги до програмної документації

Програмна документація повинна містити в собі:

- технічне завдання;
- керівництво програміста;
- керівництво оператора;
- опис програми.

A.6 Існуючі обмеження для інсталяцію чи доступу до програмного застосунку

Спроектований онлайн-сервісний застосунок не вимагає встановлення. Безперебійне використання гарантується за умови відповідності системи вимогам до технічних засобів.

ДОДАТОК Б

Опис програмного продукту

Б.1 Загальні відомості

В результаті роботи було розроблено онлайн-сервісний застосунок для співвласників багатоквартирного будинку, який вирішує більшість недоліків вже існуючих рішень, простий у використанні та адаптований до використання на портативних пристроях.

Основними мінімальними вимогами для функціонування вищезгаданого застосунку є:

- операційна система Microsoft Windows 7 і вище, або Linux з ядром системи версії Kernel 11 та вище;
- центральний процесор з тактовою частотою 2.0 ГГц чи вище та кількістю ядер не менше двох;
- об'єм оперативної пам'яті вище за 3 Гб;
- об'єм постійної пам'яті вище за 1 Гб;
- доступ до мережі інтернет, з пропускнуою здатністю вище за 20 Мб\с.

Б.2 Функціональне призначення

Функціональні вимоги були визначені такі:

- реєстрація мешканців;
- додавання адміністратором інформації щодо тарифів на комунальні послуги;
- додавання адміністратором звітів;
- додавання адміністратором оголошень чи повідомлень;
- додавання користувачем даних про спожиті комунальні послуги;
- сортування даних про спожиті раніше послуги та датою чи типом;
- постановка адміністратором питання на голосування;
- додавання користувачем пропозиції на голосування;
- можливість користувача віддати свій голос за пропозицію.

Б.3 Опис структури

Проектований застосунок складається з клієнтської частини, що забезпечує введення, первинну валідацію та відображення даних, відповідає за всю взаємодію користувача із програмним продуктом. Та серверної частини, яка забезпечує повторну валідацію, обробку та збереження даних у БД.

Застосунок побудований за архітектурою REST, що говорить про те, що взаємодія між клієнтською та серверною частинами відбувається по протоколу HTTP.

Дані між «клієнтом» та «сервером» передаються у фалах типу JSON. Вибір саме цього типу файлів був зумовлений легкістю маніпуляцій з ними за допомогою обраної мови програмування JavaScript та використання саме цього формату для збереження документів у нереляційній базі даних MongoDB.

Як клієнтська, так і серверна частини були розроблені на мові програмування JavaScript.

Для реалізації клієнтської частини застосунка було використано фреймворк Vuejs. Це сучасний фреймворк, розроблений для створення інтерфейсів користувача. Його перевагою є можливість використання однофайлових компонентів Vue – це формат файлів, що дозволяють зібрати шаблон, логіку та стилі компонента. Це значно спрощує розробку та повторене використання створених компонентів, оскільки в своїй більшості вони виходять повністю незалежними один від одного [20].

Серверна частина була реалізована за допомогою фреймворка Express для застосунків Node.js.

Node.js – це деяке подієве оточення для виконання JavaScript, створене для розробки масштабованих вебзастосунків [21].

Фреймворк Express є надбудовою над класичним Node.js та мінімалістичним та гнучким інструментом для швидкої побудови серверного API [22].

Для зберігання численних даних, необхідних для коректного функціонування застосунку було обрано документоорієнтовану базу даних MongoDB. Її перевагою можливість хмарного розміщення БД та зберігання даних у JSON подібному форматі. Отже кожен запис у такій БД є окремим документом формату, подібного до JSON, що дозволяє зберігати дані про об'єкт як окрему повну сутність, не розбиваючи її на декілька окремих. Це та індексний доступ до екземплярів сутностей дозволяє значно зменшити кількість операцій та часу для отримання необхідних даних.

Ще однією значною перевагою БД цього типу є можливість динамічного визначення параметрів екземплярів сутностей. Тобто різні екземпляри однієї сутності можуть містити різний набір параметрів.

При використанні проєктованого онлайн-сервісного застосунку незареєстрований користувач має доступ лише до базової інформації, що розміщена на головній сторінці вебзастосунку. Для використання всіх можливостей системи користувачу необхідно зареєструватися.

Після проходження реєстрації користувач може отримати доступ до особистого кабінету, де зосереджено основний функціонал системи.

Окремим пунктом йде адміністратор, який має розширені можливості у порівнянні зі звичайним користувачем. Наприклад видаляти звичайних користувачів, створювати оголошення, звіти тощо.

Нижче на рисунку Б.1 наведена загальна схема взаємодії користувачів різних ролей проєктованою системою.

Щоб проілюструвати клієнт-серверну взаємодію на рисунку Б.2 приведено спрощену схему роботи архітектури «клієнт - сервер».

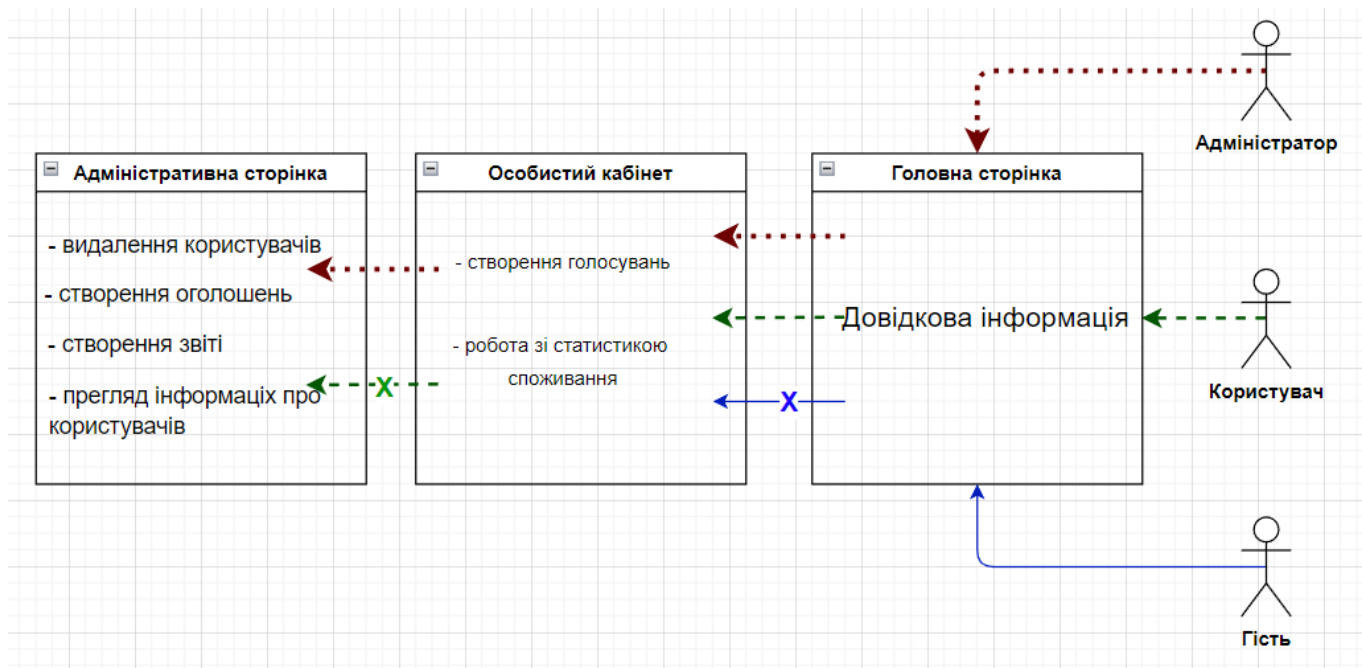


Рисунок Б.1 – Схема взаємодії користувачів з системою

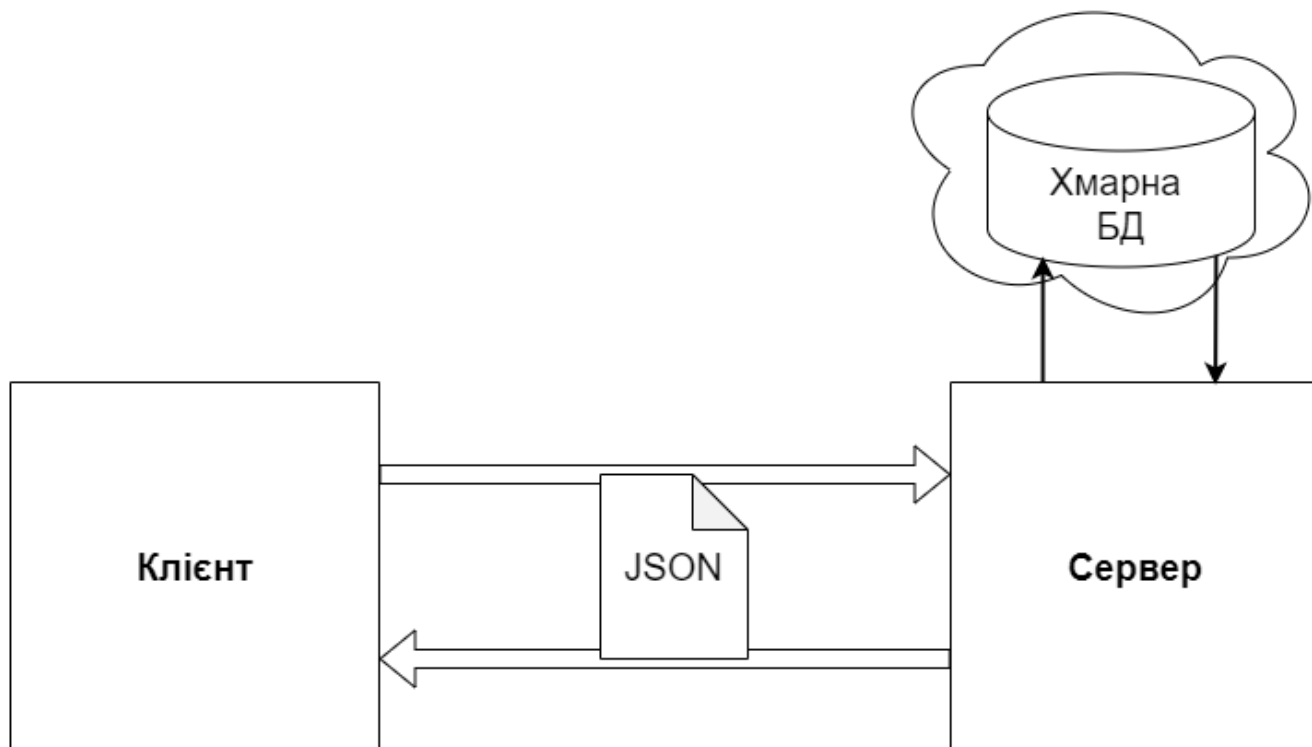


Рисунок Б.2 – Схема клієнт-серверної взаємодії

Б.4 Вибір інструментів та засобів розробки

Для реалізації онлайн-сервісного застосунку було обрано мову програмування JavaScript, як більш підходящу для виконуваних задач. Її перевагами є підтримуваність, швидкість роботи, велика кількість додаткових бібліотек та пристосовуваність як до клієнтської, так і для серверної розробки.

Для використання у проєкті було обрано документоорієнтовану базу даних, а саме MongoDB. Цей вибір був обумовлений гнучкістю цього типу БД, швидкістю роботи та масштабованістю. Як середовище розробки було обрано поширюване на безоплатній основі IDE – Visual Studio Code. Окрім своєї доступності це середовище вирізняється можливістю гнучко налаштовуватись під вимоги користувача, має безліч розширень та широку підтримку.

В якості архітектури програмного продукту було обрано REST. За якою застосунок складається з двох майже незалежних частин – клієнтської та серверної. Ці частини взаємодіють між собою за допомогою протоколу HTTP. Клієнтська частина відповідає за відображення, збір даних та передачу даних на сервер. Сервер з під'єднаною до нього БД обробляє та зберігає дані.

Візуальну складову клієнтської частини було розроблено за допомогою фреймворку Vue.js.

Серверна частина реалізована за допомогою фреймворка Express.js та середовища Node.js.

Б.5 Звернення до програмного продукту

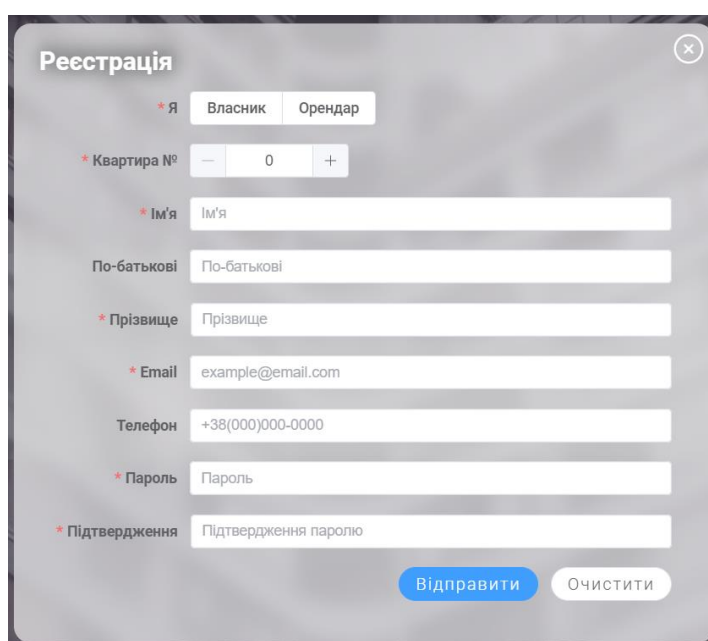
Щоб отримати доступ до стабільної загальнодоступної версії сайту необхідно перейти за посиланням у браузері.

Для отримання доступу до вихідного коду застосунку, що розроблюється необхідно за допомогою утиліти системи контролю версій Git підключитись до віддаленого репозиторію та клонувати проєкт на свою локальну станцію. Після чого, викликавши системну консоль в директорії

проєкту на локальній станції, встановити пакети, яких бракує за допомогою команди «npm i». Дочекавшись завершення встановлювання всіх необхідних пакетів, можна почати розробку запустивши виконання проєкту на локальній машині командою «npm run dev». Зробити це необхідно як для клієнтської частини, так і для серверної. Це дозволить системі виконувати автоматичну перекомпіляцію коду після кожної збереженої зміни в файлах застосунку. Отримати доступ до результатів виконання клієнтської частини програмного застосунку на локальній станції можна за адресою «http://localhost:3000» в браузері. Запити до сервера будуть доступні за локальною адресою «http://localhost:5000/api».

Б.6 Вхідні та вихідні дані

Як вхідні дані онлайн-сервісного застосунку для співвласників багатоквартирного будинку можна розглядати текстові дані, які вводяться у різноманітні форми, на кшталт форми реєстрації нового користувача, що зображена на рисунку Б.3. Також вхідними даними можна вважати будь-які файли, які користувачі завантажують у певних формах.



The image shows a web form for user registration. At the top, it has a title 'Реєстрація' and a close button. Below the title, there is a radio button group for 'Я' (I) with options 'Власник' (Owner) and 'Орендар' (Tenant). The form includes several input fields: 'Квартира №' (Apartment No.) with a numeric input and minus/plus buttons; 'Ім'я' (Name); 'По-батькові' (Patronymic); 'Прізвище' (Surname); 'Email'; 'Телефон' (Phone) with a country code dropdown and numeric input; 'Пароль' (Password); and 'Підтвердження' (Confirmation) for password confirmation. At the bottom, there are two buttons: 'Відправити' (Send) and 'Очистити' (Clear).

Рисунок Б.3— Форма реєстрації користувача

Вихідні дані представлені текстовою інформацією на вебсторінках, графіками, діаграмами, таблицями тощо. Приклади деяких з них наведено на рисунках Б.4, Б.5, Б.6.

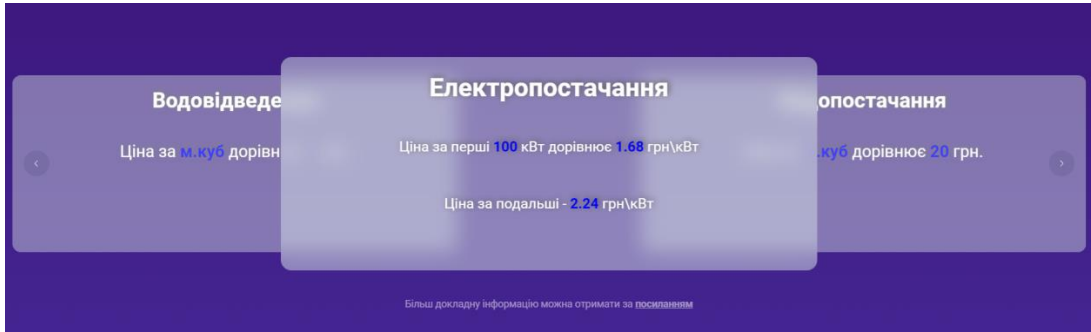


Рисунок Б.4 – Відомості щодо тарифів на енергоносії

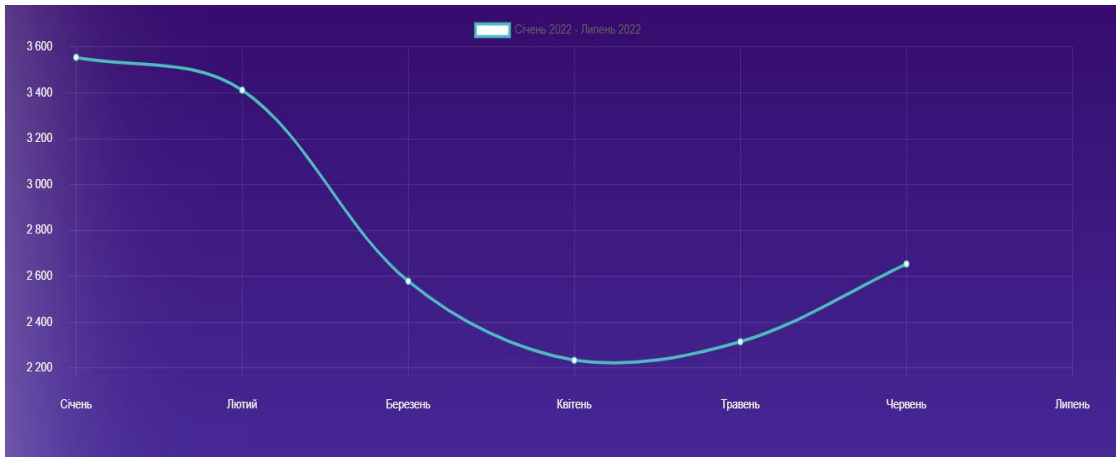


Рисунок Б.5 – Графік витрачених коштів

Кв.	Ім'я	Email	Статус	Телефон	Дії
1	Карамболь Олена Василівна	ex@ecx.com	Власник	+380971111111	Редагувати Видалити
2	Щериця Анатолій Андрійович	owner@ecx.com	Власник	+380972222222	Редагувати Видалити
3	Живченко Світлана Миколаївна	ex9988@ecx.com	Власник	+380973333333	Редагувати Видалити
4	Андерсон Реджинальд Томас	асрох@ecx.com	Власник	+380974444444	Редагувати Видалити
5	Занечуй Віктор Романович	gist_body@ecx.com	Орендар	+380975555555	Редагувати Видалити

Рисунок Б.6 – Таблиця «Будинкової книги»

ДОДАТОК В
Лістинги файлів роботи

B.1 Файл main.js

```
import { createApp } from 'vue'
import { createPinia } from 'pinia'
import { ElNotification } from 'element-plus'

import App from './App.vue'
import router from './router'

const app = createApp(App)
app.use(ElNotification);
app.use(createPinia())
app.use(router)
app.mount('#app')
```

B.2 App.vue

```
<script setup>
import { RouterLink, RouterView } from 'vue-router'
import TheFooter from "@/components/TheFooter.vue";
import { onMounted } from 'vue';
import {useAuthStore} from "@/stores/AuthStore.js"

const {checkAuth} = useAuthStore();
onMounted(()=>{
  if(localStorage.getItem('token')){
    checkAuth();
  }
});
</script>

<template>
  <div class="container">
    <el-backtop :right="40" :bottom="100" style="color:#fff; background-color: rgba(255,
255, 255, 0.15);"/>
    <el-container direction="vertical">

      <router-view/>
```

```

    <the-footer/>
  </el-container>

</div>
</template>

<style>
@import '@/assets/base.css';
@import
url('https://fonts.googleapis.com/css2?family=Roboto:ital,wght@0,100;0,300;0,400;0,500;
0,700;0,900;1,100;1,300;1,400;1,500;1,700;1,900&display=swap');

html, body {
  height: 200vh;
  scrollbar-color: #6969dd #e0e0e0;
  scrollbar-width: thin;
}
body::-webkit-scrollbar-thumb {
  background-color: rgba(255, 255, 255, 0.24);
  box-shadow: inset 2px 2px 5px 0 rgba(#fff, 0.5);
  border-radius: 100px;
}
body::-webkit-scrollbar-thumb:hover {
  background-color: rgba(255, 255, 255, 0.64);
}
body::-webkit-scrollbar {
  width: 7px;
}

body::-webkit-scrollbar-track {
  background-image: linear-gradient(to top, #5349dfc8 0%, rgba(51,8,103,0.9) 100%);
  opacity: .5;
}
#app {
  /* max-width: 1280px; */
  background-image: linear-gradient(to top, #5349dfe4 0%, #330867 100%);
  margin: 0 auto;
  /* padding: 2rem; */

```

```

    font-family: 'Roboto', sans-serif;
    font-weight: normal;
    background-color: #fff;
    /* height:100%; */
}
a{
    text-decoration: none;
    color: #fff
}
/* */
.container{
    display: flex;
    flex-direction: column;
    height: 100%;
}

</style>

```

B.3 MainPage.vue

```

<template>

    <the-header/>
    <el-main class="main">
        <MainView />

    </el-main>

</template>
<script>

import TheHeader from "@components/TheHeader.vue"
import MainView from "../views/MainView.vue"
export default{
    components:{TheHeader, MainView},
    data(){
        return{

        }
    }
}

```

```

    }
</script>
<style>

    .main{
        height:100%;
        width: 100%;
        padding:0;
        padding-top:60px;
        padding-bottom:60px;

    }

</style>

```

B.4 MainView.vue

```

<template>
    <div class = "main-view">
        <div id="first" >
            <div class="card">
                <h1>Доброчетність, надійність, добросусідство</h1>
                <h2>це наше життєве кредо!</h2>
            </div>
            <div class="background-layout"></div>

        </div>
        <div id="second" >
            <el-carousel indicator-position="none" type="card">
                <el-carousel-item v-for="item in prices.prices" :key="item.id">
                    <h3 text="2x1" justify="center">{{ item.type }}</h3>
                    <div class="limit-price" v-if="item.discountValueLimit">
                        <div >Ціна за перші <strong>{{item.discountValueLimit}}</strong>
                        {{item.measure}} дорівнює <strong>{{item.discountPrice}}</strong>
                        грн\{{item.measure}}</div>
                        <div>Ціна за подальші - <strong>{{item.standartPrice}}</strong>
                        грн\{{item.measure}}</div>
                    </div>
                    <div class="standart-price" v-else>

```

```

        <div>Ціна за <strong>{{item.measure}}</strong> дорівнює
<strong>{{item.standartPrice}}</strong> грн.</div>
    </div>
</el-carousel-item>
</el-carousel>
    <div style="color:#fff; width:100%; text-align:center; padding: 40px 0 0;
opacity: .5;font-size: 14px;">
        Більш докладну інформацію можна отримати за
        <strong style="font-weight:bold;text-decoration: underline;">
            <a href="https://zp.gov.ua/uk/page/perelik-tarifiv">посиланням</a>
        </strong>
    </div>
</div>
<div id="third">
    <div>
        <div class="important-numbers-base">
            <div class="card">
                <div class="important-number" ><a style="color:yellowgreen"
href="tel:112">112</a></div>
                <div class="important-number-description">Центр громадської безпеки
(універсальний)</div>
            </div>
        </div>
        <div class="important-numbers-base">
            <div class="card">
                <div class="important-number" ><a style="color:rgb(214, 6, 6)"
href="tel:101">101</a></div>
                <div class="important-number-description">Пожежна допомога</div>
            </div>
        </div>
        <div class="important-numbers-base">
            <div class="card">
                <div class="important-number" ><a style="color:darkblue"
href="tel:102">102</a></div>
                <div class="important-number-description">Поліція</div>
            </div>
        </div>
        <div class="important-numbers-base">
            <div class="card">
                <div class="important-number" ><a style="color:red"
href="tel:103">103</a></div>

```

```

        <div class="important-number-description">Швидка медична допомога</div>
    </div>
</div>
<div class="important-numbers-base">
    <div class="card">
        <div class="important-number" ><a style="color:orange"
href="tel:104">104</a></div>
        <div class="important-number-description">Аварійна служба газової
мережі</div>
    </div>
</div>
<div class="important-numbers-base">
    <div class="card">
        <div class="important-number" ><a style="color:#fff"
href="tel:109">109</a></div>
        <div class="important-number-description">Довідкова служба</div>
    </div>
</div>
<div>
    <!-- Collaps -->
    <div class="collaps-numbers">
        <div class="collaps-numbers-line"></div>
        <div class="collaps-numbers-button" @click="extendedNumbersToggle">
            <el-icon><ArrowDownBold /></el-icon>
        </div>
        <div class="collaps-numbers-line"></div>
    </div>
    <!-- Collaps END -->
    <transition name="fade">
    <div class="extended-numbers" style="width:90%" v-show="!isCollapsed">
        <div class="card">
            <div class='content-extended-numbers'>
                <h5 style="color: darkred;font-size: 24px">Медицина, допомога
громадянам та ветеринарна служба</h5>
                <br/>
                <p><b>(061) 787-60-97; (061) 236-51-52; (061) 769-76-91</b> - станції
Швидкої медичної допомоги<br>
                <b>(061) 64-38-11</b> - Травмпункт при дитячій обл. лікарні<br>
                <b>(061) 34-51-10</b> - Консультативна служба (для самостійного надання
медичної допомоги на дому)<br>

```

1503 – Центральна диспетчерська служба швидкої допомоги

 (061) 65-56-20 – Запорізька міська державна лікарня ветеринарної
 медицини</p>

 <h5 style="color: darkred; font-size: 24px;">Електромережі та
 світло</h5>

 <p>(061) 233-13-15; (061) 65-01-05 – Аварійно-диспетчерська
 служба міськсвітло

 (061) 286-36-33; (061) 233-01-42; (061) 286-36-52 – Міські
 електричні мережі</p>

 <h5 style="color: darkred;font-size: 24px">Газ</h5>

 <p>1581 – Ремонтна служба «Запоріжгазу»</p>

 <h5 style="color: darkred;font-size: 24px">Надзвичайна ситуація</h5>

 <p>(061) 62-63-71 – Черговий штабу запорізької цивільної
 оборони

 (061) 64-67-11; (061) 57-06-94 – Запорізька міська рятувально-
 водолазна служба

 (061) 233-13-74 – Обласна служба порятунку</p>

 <h5 style="color: darkred;font-size: 24px">Тепломережі, водопостачання
 та каналізації</h5>

 <p>(061) 764-21-78; (061) 764-22-95; (061) 236-61-81 –
 Міськводоканал</p>

 <h5 style="color: darkred;font-size: 24px">ЖКГ та будинки</h5>

 <p>1586 – Аварійна служба ліфтів

 (061) 224-69-02, (061) 224-02-94 – Управління комунального
 господарства Запорізької міської ради</p>

 <h5 style="color: darkred;font-size: 24px">Транспорт</h5>

 <p>15-05; (061) 224-40-60 – Довідкова вокзалу “Запоріжжя-1”

 (061) 224-40-46 – Довідкова запорізького Автовокзалу

 (061) 62-61-51 – Електротранспорт</p>

```

        <br/>
    </div>
</div>
</div>
</div>
</transition>
</div>
<div>
    <!-- Collaps -->
    <div class="collaps-numbers">
        <div class="collaps-numbers-line"></div>
        <div class="collaps-numbers-button" @click="extendedLinksToggle">
            <el-icon><ArrowDownBold /></el-icon>
        </div>
        <div class="collaps-numbers-line"></div>
    </div>
    <!-- Collaps END -->
    <transition name="fade">
        <div v-show="!isLinksCollapsed" class="extended-numbers" style="width:90%">

            <div class = "card">
                <h1>Посилання на корисні ресурси</h1>
                <div style="display:flex; justify-content:space-around; align-items:center; width:100%; flex-wrap:wrap;">
                    <div class="important-number"><a href="https://zp.gov.ua/uk/page/likuvalni-zakladi">Медичні заклади</a></div>
                    <div class="important-number"><a href="https://zp.gov.ua/uk/page/sportivni-zakladi">Спортивні заклади</a></div>
                    <div class="important-number"><a href="https://zp.gov.ua/uk/page/zakladi-kulturi">Заклади культури</a></div>
                    <div class="important-number"><a href="https://zp.gov.ua/uk/page/e-services">Е-СЕРВИСИ</a></div>
                    <div class="important-number"><a href="https://zp.gov.ua/uk/page/osvita">Освіта</a></div>

                </div>

            </div>
        </div>
    </transition>
</div>

```

```

        </div>
    </div>
</template>
<script>

import {usePriceStore} from "@stores/priceStore.js";
import { ArrowDownBold} from '@element-plus/icons-vue';
export default {

    setup(){

        const prices = usePriceStore();
        return {prices}

    },
    components:{
        ArrowDownBold
    },

    data() {
        return {
            isCollapsed:false,
            isLinksCollapsed:false,
        }
    },
    methods:{
        extendedNumbersToggle(){
            const e = document.getElementsByClassName('collaps-numbers-button')[0];
            e.style.transform = `rotate(${180*this.isCollapsed*1}deg)`
            this.isCollapsed = !this.isCollapsed;
        },
        extendedLinksToggle(){
            const e = document.getElementsByClassName('collaps-numbers-button')[1];
            e.style.transform = `rotate(${180*this.isLinksCollapsed*1}deg)`
            this.isLinksCollapsed = !this.isLinksCollapsed;
        }
    },

    mounted(){

```

```

        this.prices.getPrices();

    }

}
</script>
<style >

.fade-enter-active,
.fade-leave-active {
    transition: opacity 0.5s ease;
}

.fade-enter-from,
.fade-leave-to {
    opacity: 0;
}

.content-extended-numbers{
    line-height: 20px;
    font-size: 16px;
}

.content-extended-numbers p>b{
    font-weight: bold;
}

.collaps-numbers{

    margin: 25px 25px;
    display: flex;
    flex-direction: row;
    align-items: center;
    justify-content: center;
    cursor: pointer;
    width: 100%;
}

.collaps-numbers-line{
    width:100%;
    height: 10px;
    border: 1px solid rgba(255, 255, 255, 0.3);
    border-radius: 15px;
    background: rgba(255, 255, 255, 0.5);
    backdrop-filter: blur(2px);
}

```

```

    box-shadow: 0px 0px 9px 7px rgba(255,255,255,.1);
    -webkit-box-shadow: 0px 0px 9px 7px rgba(255,255,255,.1) ;
    -moz-box-shadow: 0px 0px 9px 7px rgba(255,255,255,.1) ;
}
.collaps-numbers-button{
    width: 40px;
    height: 40px;
    border-radius: 100%;
    border:3px solid rgba(255, 255, 255, 0.5);
    text-align: center;
    background: rgba(255, 255, 255, 0.3);
    transition:.3s;
    cursor:pointer;
    margin: 0 10px;
    padding: 0 10px;
    display: flex;
    flex-direction: row;
    align-items: center;
    justify-content: center;
    color:#fff;
    box-shadow: 0px 0px 9px 7px rgba(255,255,255,.1);
    -webkit-box-shadow: 0px 0px 9px 7px rgba(255,255,255,.1) ;
    -moz-box-shadow: 0px 0px 9px 7px rgba(255,255,255,.1) ;

}
.collaps-numbers-button:hover{
    transform: scale(1.2);
    transition:.3s;
}
.important-numbers-base{
    padding:10px;
}
.important-number{
    padding: 10px 0;
    transition: .3s;
}
.important-number:hover{
    transform: scale(1.1);
    transition: .3s;
}
.important-number a{

```

```

    font-size: 48px;
    font-weight: bold;
    padding: 10px;

}
.important-number-description{
    color:#fff;
    text-align:center;
}
.main-view{
    display: flex;
    flex-direction: column;
    height: 100%;
    width:100%;
    align-items: center;
    justify-content: center;
}
#first{
    display: flex;;
    height: 250%;
    width:100%;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    position: relative;
    min-height: 550px;

}
.background-layout{
    width: 100%;
    height: 100%;
    position: absolute;
    top:0;
    left:0;
    background-image:url("/home-page-photo.jpeg");
    background-size: cover;
    opacity: .80;

    z-index: 1;
}
#second{

```

```

    height: 100%;
    width: 100%;
    padding: 80px 0 50px;
}
#third{
    height: 100%;
    width: 100%;
    display: flex;
    flex-wrap: wrap;
    justify-content: center;
    align-items: center;
}
#third>div{
    height: 100%;
    width: 100%;
    display: flex;
    flex-wrap: wrap;
    justify-content: center;
    align-items: center;
}
.card{
    position: relative;
    min-width: 350px;
    /* width: 100%; */
    /* height: 30em; */
    display: flex;
    flex-direction: column;
    color: aliceblue;
    align-items: center;
    justify-content: center;
    border-radius: 15px;
    background-color: rgba(255,255,255,.3);
    backdrop-filter: blur(2px);
    padding: 4rem;
    box-shadow: 0px 0px 9px 7px rgba(255,255,255,.1);
    -webkit-box-shadow: 0px 0px 9px 7px rgba(255,255,255,.1) ;
    -moz-box-shadow: 0px 0px 9px 7px rgba(255,255,255,.1) ;
    z-index: 2;
}
.card h1{
    font-size: 30px;

```

```

padding: 10px 0;
text-align: center;
text-shadow: 2px -2px 9px rgb(0 0 0)
}
.card h2{
text-shadow: 2px -2px 9px rgb(0 0 0)
}
.el-carousel{
margin: 0 10px;
}
.el-carousel__item h3 {
color: #ffff;
font-size: 36px;
line-height: auto;
margin: 0;
text-align: center;
padding: 25px 0;
font-weight: bold;
}

.el-carousel__item {
background-color: #99a9bf52;
backdrop-filter: blur(5px);
/* background: transparent; */
color: #fff;
text-shadow: 2px -2px 9px rgb(90, 90, 90)
}

.el-carousel__item--card.is-active {
background-color: #d3dce685;
backdrop-filter: blur(13px);
}

.el-carousel__item--card{
border-radius: 15px;
}
.limit-price{
display: flex;
justify-content: center;
align-items: center;
flex-direction: column;

```

```

    font-size: 20px;
}
.limit-price div{
    padding: 30px;
}
.standart-price{
    display: flex;
    justify-content: center;
    align-items: center;
    flex-direction: column;
    font-size: 28px;
}
.standart-price div{
    padding: 30px;
}
.standart-price strong, .limit-price strong {
    font-weight: bold;
    color:blue
}
/* .limit-price strong{
    font-weight: bold;
} */
@media (max-width:760px) {
    .standart-price div{
        padding: 10px;
        text-align: center;
    }
    .limit-price div{
        padding: 10px;
        text-align: center;
    }
    .el-carousel__item h3 {
        font-size: 20px;
        padding: 25px;
    }
    .card{
        min-width: auto;
    }
    .important-numbers-base{
        width:100%
    }

```

```

}
}
</style>

```

B.5 NewLineChart.vue

```

<template>
  <Line
    :chart-options="chartOptions"
    :chart-data="chartData"
    :chart-id="chartId"
    :dataset-id-key="datasetIdKey"
    :plugins="plugins"

    :css-classes="cssClasses"
    :styles="styles"
    :width="width"
    :height="height"
  />
</template>

<script>
import { Line } from "vue-chartjs";

import {
  Chart as ChartJS,
  Title,
  Tooltip,
  Legend,
  LineElement,
  LinearScale,
  CategoryScale,
  PointElement,
} from "chart.js";

ChartJS.register(
  Title,
  Tooltip,
  Legend,
  LineElement,

```

```

    LinearScale,
    CategoryScale,
    PointElement
  );

export default {
  name: "LineChart",
  components: {
    Line,
  },
  props: {
    chartId: {
      type: String,
      default: "line-chart",
    },
    datasetIdKey: {
      type: String,
      default: "label",
    },
    width: {
      type: Number,
      default: 400,
    },
    height: {
      type: Number,
      default: 400,
    },
    cssClasses: {
      default: "",
      type: String,
    },
    styles: {
      type: Object,
      default: () => {},
    },
    plugins: {
      type: Array,
      default: () => [],
    },
    chartData: {
      type: Object,

```

```

    default: ()=>{},
  }
},
data() {
  return {

    chartOptions: {
      maintainAspectRatio: false,
      legend: {
        display: false
      },
      responsive: true,
      tooltips: {
        backgroundColor: '#f5f5f5',
        titleFontColor: '#333',
        bodyFontColor: '#666',
        bodySpacing: 4,
        xPadding: 12,
        mode: "nearest",
        intersect: 0,
        position: "nearest"
      },
      tension: 0.35,
      scales: {
        y: {
          barPercentage: 1.6,
          grid: {
            drawBorder: false,
            color: 'rgba(255,255,255,0.1)',

            zeroLineColor: "transparent",
          },
          ticks: {
            suggestedMin: 50,
            suggestedMax: 110,
            padding: 20,
            fontColor: "#ff8a76",
            color:"white"
          },
        },
      },
    },
  },
},

```

```

x: {
  barPercentage: 1.6,
  grid: {
    drawBorder: false,
    color: 'rgba(255,255,255,0.1)',
    zeroLineColor: "transparent",
  },
  ticks: {
    padding: 20,
    fontColor: "#faa091",
    color:"white"
  },
}

},
};
},
};
</script>

```

B.6 TheHeader.vue

```

<template>
  <el-header :class="{scrolled: isUserScroll}" class="header">
    <div class="header-left">
      <div v-if="!isMobile">
        <router-link to="/">
          <div class="logo">
            <!--  -->
            <div>Cerera</div>
          </div>
        </router-link>
      </div>
      <div v-else>
        <el-icon style="font-size:2rem" @click="drawer = true"><Expand /></el-
icon>
        <el-drawer v-model="drawer" direction="ltr" title="Menu" :with-

```

```

header="false" size=45%>
    <div class="nav">
        <ul class="nav-list">
            <router-link to="#first" class="nav-item"><el-button round
color="transparent" >Головна</el-button> </router-link>
            <router-link to="#second" class="nav-item"><el-button round
color="transparent" >Тарифи</el-button></router-link>
            <router-link to="#third" class="nav-item"><el-button round
color="transparent" >Корисне</el-button></router-link>
        </ul>
    </div>
</el-drawer>
</div>
</div>
<div class="header-midd">
    <div v-if="isMobile">
        <router-link to="/">
            <div class="logo">
                <!--  -->
                <div>Cerera</div>
            </div>
        </router-link>
    </div>
</div>
<div class="header-right">
    <div class="nav" v-if="!isMobile">
        <ul class="nav-list">
            <router-link to="#first" class="nav-item"><el-button round
color="transparent" >Головна</el-button> </router-link>
            <router-link to="#second" class="nav-item"><el-button round
color="transparent" >Тарифи</el-button></router-link>
            <router-link to="#third" class="nav-item"><el-button round
color="transparent" >Корисне</el-button></router-link>
        </ul>
    </div>
    <UserDropdown/>
</div>
</el-header>
</template>

```

```

<script>
import {Expand} from '@element-plus/icons-vue';
import UserDropdown from './UserDropdown.vue';
import { useBehaviorsStore } from '../stores/behaviors'
import { storeToRefs } from 'pinia'
export default {
  components:{
    UserDropdown, Expand
  },
  setup(){
    const behaviors = useBehaviorsStore();
    const {isUserScroll, isMobile} = storeToRefs(behaviors);
    return {isUserScroll,isMobile,behaviors};
  },
  data() {
    return {
      isLoggedIn: false,
      drawer:false,
    }
  },
  methods:{
  },
  mounted () {
    window.addEventListener('scroll', this.behaviors.handleScroll);
    window.addEventListener('resize', this.behaviors.handleResize);
    this.behaviors.handleResize()
  },
  beforeDestroy () {
    window.removeEventListener('scroll', this.behaviors.handleScroll);
    window.removeEventListener('resize', this.behaviors.handleResize);

  },
}
</script>
<style >
.header{
  /* background-color: #1d3043; */
  height: 4rem;
  display: flex;
  flex-direction: row;
  justify-content: space-between;

```

```

    align-items: center;
    color: #fff;
    font-weight: 600;
    font-size: 16px;
    position: fixed;
    width: 100%;
    z-index: 99;
}
.scrolled{
    background: inherit;
    transition: .5s;
}
.scrolled:before {
    background: inherit;
    box-shadow: inset 0 0 200px rgba(255, 255, 255, .4);
    filter: blur(2px);
    transition: .5s;
    content: '';
    position: absolute;
    top: 0;
    left: 0;
    right: 0;
    bottom: 0;
    animation-duration: .5s;
    animation-name: slide;
}
@keyframes slide {
    from {
        box-shadow: none;
        filter: none;
    }
    to {
        box-shadow: inset 0 0 200px rgba(255, 255, 255, .4);
        filter: blur(2px);
    }
}
.el-drawer{
    background-color: rgba(255, 255, 255, .15);
    backdrop-filter: blur(10px);
    transition: .5s;
}

```

```

.el-drawer__body{
  display: flex;
  justify-content: flex-start;
  align-items: flex-start;
}
.header-left{
  width:auto;
  display: flex;
  flex-direction:row;
  align-items: center;
  justify-content: flex-start;
  /* padding:0 2% 0 2%;          */
}
.logo{
  display: flex;
  flex-direction:row;
  align-items: center;
  justify-content: flex-start;
  font-size: 24px;
  font-weight: 500;
  cursor:pointer;
  letter-spacing: .2rem;
}

.header-midd{
  width:100%;
  display: flex;
  flex-direction: row;
  align-items: center;
  justify-content: center;
  padding:0 2% 0 2%;
}
.header-right{
  width:auto;
  display: flex;
  flex-direction: row;
  align-items: center;
  justify-content: flex-end;
  /* padding:0 2% 0 2%; */
}
.nav{

```

```

padding: 0 4%;
z-index: 98;
}
.nav-list{
display: flex;
flex-direction: row;
align-items: flex-start;
justify-content: flex-start;
}

.nav-item{
padding: 0 10px;
cursor: pointer;
}

.el-button>span {
font-size: 1rem;
letter-spacing: .1em;
font-weight: 300;
font-family: 'Roboto', sans-serif;
}
.el-button:not(.is-text,.is-link,.el-button--text):focus, .el-button:not(.is-text,.is-
link,.el-button--text):hover {
color: var(--el-button-hover-text-color);
border-color: hwb(0deg 100% 0% / 11%);
background-color: rgb(255 255 255 / 16%);
outline: 0;
}
@media(max-width: 760px){
.nav-list{
flex-direction: column;
align-items: flex-start;
justify-content: center;
}
.el-button>span{
font-size: 1.2rem;
font-weight: 400;
font-family: 'Roboto', sans-serif;
}
.nav-item{
padding:.7rem 0

```

```

    }
    .nav{
        padding:35% 4%;
    }
    .scrolled{
        background-color: rgba(255, 255, 255, .25);
        backdrop-filter: none;
    }
}
</style>

```

B.7 UserDropdown.vue

```

<template>
    <el-dropdown ref="dropdown_user" trigger="click" style="margin-right: 0"
    size="large">
        <div >
            <el-avatar @click.prevent="showClick" :class = "isAuth ? 'authorized' :
            'unauthorized'" ><el-icon><UserFilled/></el-icon></el-avatar>
        </div>
        <template #dropdown>
            <el-dropdown-menu v-if="isAuth" >
                <router-link to="/dashboard"><el-dropdown-item ><el-icon><Histogram
                /></el-icon>Кабінет</el-dropdown-item> </router-link>
                <el-dropdown-item @click.prevent="auth.logout()"><el-icon><SwitchButton
                /></el-icon><strong style="font-weight: bolder;">Вийти</strong></el-dropdown-item>
            </el-dropdown-menu>
            <el-dropdown-menu v-else >
                <el-dropdown-item @click="isModalLoginVisible=true"><el-icon><Key
                /></el-icon>Вхід</el-dropdown-item>
                <el-dropdown-item @click="isModalRegisterVisible=true"><el-icon><SetUp
                /></el-icon>Реєстрація</el-dropdown-item>
            </el-dropdown-menu>
        </template>
    </el-dropdown>
    <custom-modal v-model="isModalLoginVisible" @confirm = "modalConfirm"
    @cancel="closeModal">
        <template v-slot:title>Вхід</template>

```

```

    <div>
      <LoginForm @closeLoginModal = "closeLoginModal"/>
    </div>
  </custom-modal>
  <custom-modal v-model="isModalRegisterVisible" @confirm = "modalConfirm"
@cancel="closeModal">
    <template v-slot:title>Рееcтpация</template>
    <div>
      <RegisterForm @closeRegisterModal = "closeRegisterModal"/>
    </div>
  </custom-modal>
</template>
<script>
import { UserFilled,Histogram, SwitchButton,Key, SetUp, Expand} from '@element-
plus/icons-vue';
import CustomModal from './CustomModal.vue';
import RegisterForm from './RegisterForm.vue';
import LoginForm from './LoginForm.vue';
import {useAuthStore} from "@/stores/authStore.js"
import { storeToRefs } from 'pinia'
export default {
  setup(){
    const auth = useAuthStore();
    const {isAuth} = storeToRefs(auth);
    return {isAuth, auth}
  },
  components:{
    UserFilled, Histogram, SwitchButton, Key, SetUp, Expand,
CustomModal,RegisterForm,LoginForm
  },
  data() {
    return {
      isModalLoginVisible: false,
      isModalRegisterVisible:false,
    };
  },
  methods:{
    showClick() {
      this.$refs.dropdown_user.handleOpen()
    },
    modalConfirm() {

```

```

        this.isModalVisible = true;
    },
    closeLoginModal() {
        this.isModalLoginVisible = false;
    },
    closeRegisterModal() {
        this.isModalRegisterVisible = false;
    }
}
}
</script>
<style>
.el-dropdown-menu{
    font-family:'Roboto', sans-serif;
    background-color: rgba(255, 255, 255, 0.15);
    backdrop-filter: blur(5px);
    font-weight: 300;
}
.el-dropdown{
    cursor: pointer;
}
.el-dropdown__popper.el-popper{
    background: rgba(255, 255, 255, 0.15);
    backdrop-filter: blur(5px);
    border:none;
}
.el-popper.is-light{
    background: inherit;
}
.el-popper.is-light .el-popper__arrow::before{
    background-color: rgba(255, 255, 255, 0.15);
    border:none
}
.el-dropdown-menu__item{
    font-size: 16px;
    color:#fff
}
.el-dropdown-menu__item:not(.is-disabled):focus{
    color:#fff;
    background-color: rgba(255, 255, 255, 0.25);
}

```

```
.authorized{
  background-color: rgb(4, 128, 0);
}
.unauthorized{
  background-color: rgba(255, 255, 255, .25);
}
</style>
```

B.8 Dashboard.vue

```
<template>
<el-container>
  <el-aside :class="isAsideOpen ? 'aside open-aside' : 'aside collapsed-aside'" >
    <div class="aside-header">
      <div @click="asideSwitcher" class="aside-switcher">
        <el-icon v-if="!isAsideOpen"><Expand /></el-icon>
        <el-icon v-else><Fold /></el-icon>
      </div>
      <div v-if="isAsideOpen" class="logo">
        <router-link to="/">
          <div >
            <div>Cerera</div>
          </div>
        </router-link>
      </div>
    </div>
    <div class="nav-container">
      <ul>
        <li><router-link to="/dashboard/statistic"><el-icon class = "nav-
icon"><DataAnalysis /></el-icon><span>Статистика</span></router-link></li>
        <li><router-link to="/dashboard/votes"><el-icon class = "nav-
icon"><Finished /></el-icon><span>Голосування</span></router-link></li>
        <li><router-link to="/dashboard/alerts"><el-icon class = "nav-
icon"><ChatDotRound /></el-icon><span>Оголошення</span></router-link></li>
        <li><router-link to="/dashboard/reports"><el-icon class = "nav-
icon"><DocumentChecked/></el-icon><span>Звіти</span></router-link></li>
        <li v-if = 'user.role'><router-link
to="/dashboard/administration_service"><el-icon class = "nav-icon"><Tools /></el-
icon><span>Адміністрація</span></router-link></li>
      </ul>
```

```

        </div>
    </el-aside>
    <el-container>
        <el-header class = "dashboard-header">
            <div>
                <div v-if="(isMobile && !isAsideOpen) || !isAsideOpen"
style="padding:15px;" >
                    <router-link to="/">
                        <div class="logo">
                            <div>Cerera</div>
                        </div>
                    </router-link>
                </div>
            </div>
            <div class="user-header">
                <div style="padding:0 10px">Бирю, {{this.user.firstName}}</div>
                <UserDropdown/>
            </div>
        </el-header>

        <el-main>
            <el-scrollbar>
                <DashboardView />
            </el-scrollbar>
        </el-main>

    </el-container>
</el-container>
</template>
<script>
import DashboardView from "../views/DashboardView.vue"
import UserDropdown from '@components/UserDropdown.vue';
import {Expand,Histogram,
Fold,Tools,DataAnalysis,Finished,ChatDotRound,DocumentChecked} from '@element-
plus/icons-vue';
import {useBehaviorsStore} from "@stores/behaviors.js"
import {useAuthStore} from "@stores/authStore.js"
import { storeToRefs } from 'pinia'
export default{
    components:{
DashboardView,UserDropdown,Histogram,Expand,Fold,Tools,DataAnalysis,Finished,ChatDotRou

```

```

nd,DocumentChecked},
  setup(){
    const behaviors = useBehaviorsStore();
    const auth = useAuthStore();
    const {user} = auth;
    const {isMobile} = storeToRefs(behaviors);
    return {isMobile,behaviors,user};
  },
  data(){
    return{
      isAsideOpen: !this.isMobile,
    }
  },
  methods:{
    asideSwitcher(){
      this.isAsideOpen = !this.isAsideOpen
      console.log(this.isAsideOpen,this.isMobile)
    }
  },
  watch:{
    isMobile(){
      this.isAsideOpen= !this.isMobile
    }
  },
  mounted () {
    window.addEventListener('resize', ()=>{this.behaviors.handleResize()});
    this.behaviors.handleResize()
  },
  beforeDestroy () {
    window.removeEventListener('resize', );
  },
}
</script>
<style scoped>
  *{color:white}
  .dashboard-header{
    display: flex;
    flex-direction: row;
    align-content: center;
    justify-content: space-between;
    align-items: center;

```

```

        padding: 10px 2%;
    }
    .user-header{
        display: flex;
        flex-direction: row;
        justify-content: center;
        align-items: center;
    }
    .aside-header{
        display: flex;
        flex-wrap: nowrap;
        flex-direction: row-reverse;
        align-items: center;
        justify-content: space-between;
    }
    .logo{
        display: flex;
        flex-direction: row;
        align-items: center;
        justify-content: center;
        font-size: 24px;
        font-weight: 500;
        cursor: pointer;
        letter-spacing: .2rem;
        padding: 8%;
        animation-duration: 2s;
        animation-name: showin;
    }
    @keyframes showin {
        from {
            opacity: 0;
        }
        to {
            opacity: 1;
        }
    }
    .aside{
        background-color: rgba(255, 255, 255, .15);
        backdrop-filter: blur(15px);
        box-shadow: 1px -30px 2000px rgb(255 255 255 / 50%);
        transition: 1s;
    }

```

```

        overflow: hidden;
    }
    .collapsed-aside{
        width:40px;
        font-size:1.5rem
    }
    .collapsed-aside span{
        display:none;
    }
    .collapsed-aside .nav-container{
        justify-content: center;
        align-items: center;
    }
    .open-aside{
        width:200px;
        font-size:1.2rem;
    }
    .open-aside span{
        display:inline;
        animation-duration: 2s;
        animation-name: showin;
    }
    .nav-container{
        height: 20%;
        display: flex;
        justify-content: flex-start;
        padding: 20% 0 ;
    }
    .nav-container ul{
        display: flex;
        flex-direction:column;
        align-items: flex-start;
        justify-content: space-evenly;
        height: 100%;
        width:100%;
    }
    .nav-container ul>li{
        display: flex;
        flex-direction:row;
        align-items: center;
    }

```

```

        justify-content: flex-start;
        cursor: pointer;
        padding: 0 10%;
        width: 100%;
        height: 100%;
    }
    .nav-container ul>li:hover{
        box-shadow:inset -20px 0px 60px rgb(255 255 255 / 50%);
        transition: .3s;
    }
    .nav-container ul>li> a{
        display: flex;
        flex-wrap: nowrap;
        align-items: center;
        width: 100%;
    }
    .nav-container ul>li> a span{
        padding:10px;
        font-weight: 300;
    }
    .nav-icon{
        border:1px solid white;
        border-radius:5px;
        padding: 3px;
    }
    .aside-switcher{
        font-size: 1.5rem;
        display: flex;
        justify-content: flex-end;
        align-items: center;
    }
    .aside-switcher i{
        cursor: pointer;
        padding: 10px;
        transition: 1s;
    }
    .el-main{
        height: 150vh;
    }
}
</style>

```

ДОДАТОК Г
Слайди презентації

Національний університет «Запорізька Політехніка»

Факультет комп'ютерних наук і технологій
Кафедра програмних засобів

РОЗРОБКА ОНЛАЙН-СЕРВІСНОГО ЗАСТОСУНКУ ДЛЯ СПІВВЛАСНИКІВ БАГАТОКВАРТИРНОГО БУДИНКУ

Виконав
Студент гр. КНТ-129сп
Керівник
Доцент

К.Д. Цибульський
О.О. Степаненко

Рисунок Г.1 – Слайд №1

Мета, об'єкт, предмет дослідження

Об'єкт дослідження – об'єднання співвласників багатоквартирного будинку.

Предмет дослідження – методи побудови онлайн-сервісного застосунку для співвласників багатоквартирного будинку .

Мета роботи – розробка програмного продукту на тему «Онлайн-сервісний застосунок для співвласників багатоквартирного будинку».

Актуальність розробки обумовлена сучасними тенденціями до цифровізації всіх форм господарювання, ускладненість комунікації між співвласниками та адміністрацією в умовах карантинних чи інших обмежень та вимогам прозорості обігу спільних коштів.

Рисунок Г.2 – Слайд №2

Постановка завдання до роботи

Функціональні вимоги були визначені такі:

- реєстрація мешканців;
- додавання адміністратором інформації щодо тарифів на комунальні послуги;
- додавання адміністратором звітів;
- додавання адміністратором оголошень чи повідомлень;
- додавання користувачем даних про спожиті комунальні послуги;
- сортування даних про спожиті раніше послуги та датою чи типом;
- постановка адміністратором питання на голосування;
- додавання користувачем пропозиції на голосування;
- можливість користувача віддати свій голос за пропозицію.

Рисунок Г.3 – Слайд №3

Огляд існуючих мов програмування

Характеристики	Python	JavaScript	PHP
Призначена для веб розробки	-	+	+
Швидкість виконання	-	+	+
Документованість	+	+	+
Простота	+	+	+
Розширюваність	+	+	–
Різноманіття вбудованих методів	+	+	+
Асинхронність	+	+	+
Можливість інтеграції та взаємодії з іншими мовами	++	+	+
Кросплатформеність	+	+	+

Рисунок Г.4 – Слайд

Огляд існуючих середовищ розробки

Критерії порівняння	Visual Studio Code	WebStorm	Nova
Кросплатформеність	+	+	–
Оплатність	–	+	+
Автодоповнення	+	+	+
Інтеграція з Git	+	+	+
Підтримка сторонніх плагінів	+	–	+
Аналіз коду	+	+	+
Рефакторинг коду	+	+	+
Вбудована консоль	+	+	+
UX/UI	+	±	+
Налаштовуваність	+	±	+

№4

Рисунок Г.5 – Слайд №5

Схема клієнт-серверної взаємодії

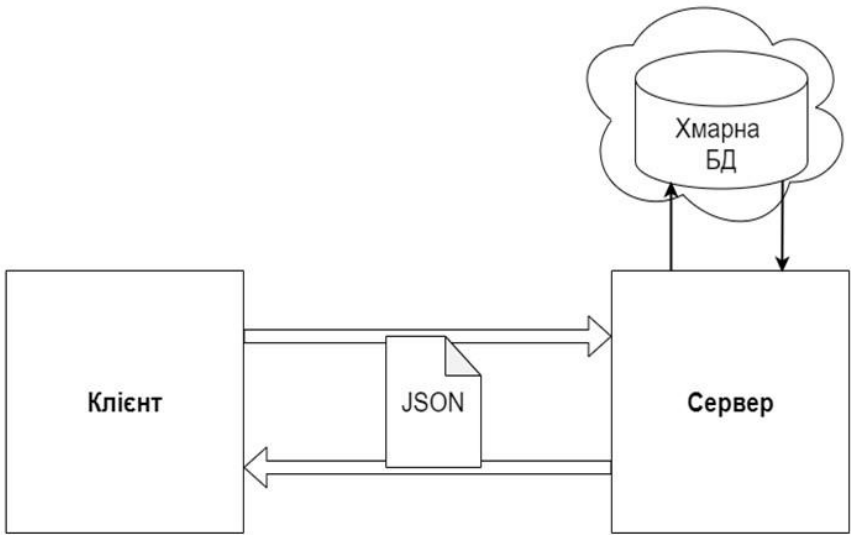
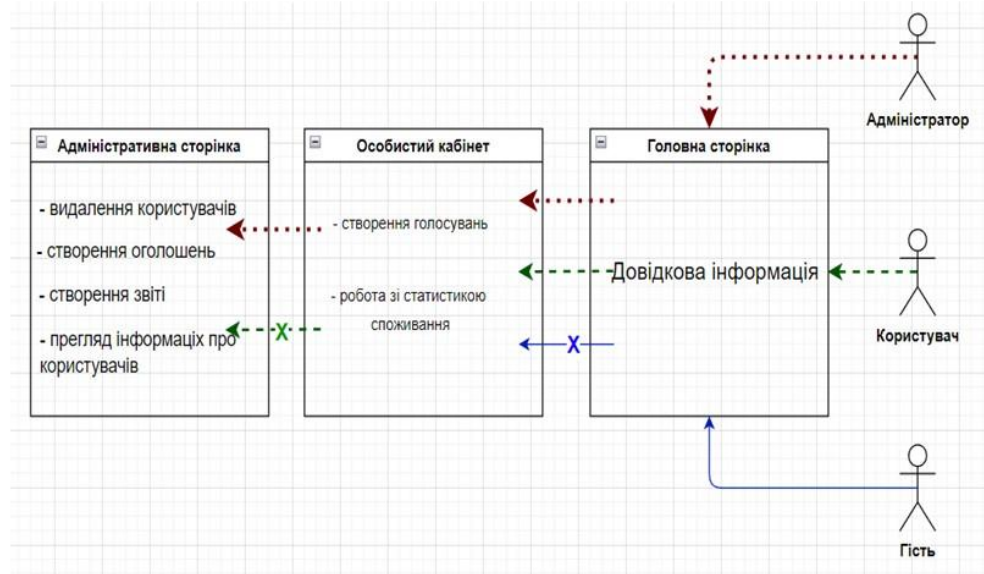


Рисунок Г.6 – Слайд

Схема взаємодії користувачів з системою



№6

Рисунок Г.7 – Слайд №7

Блок-схема процесу авторизації користувача

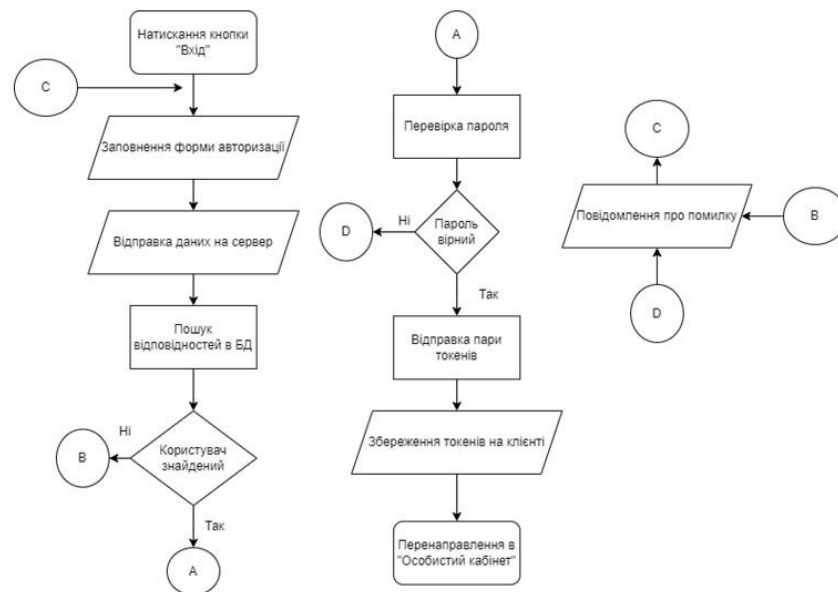
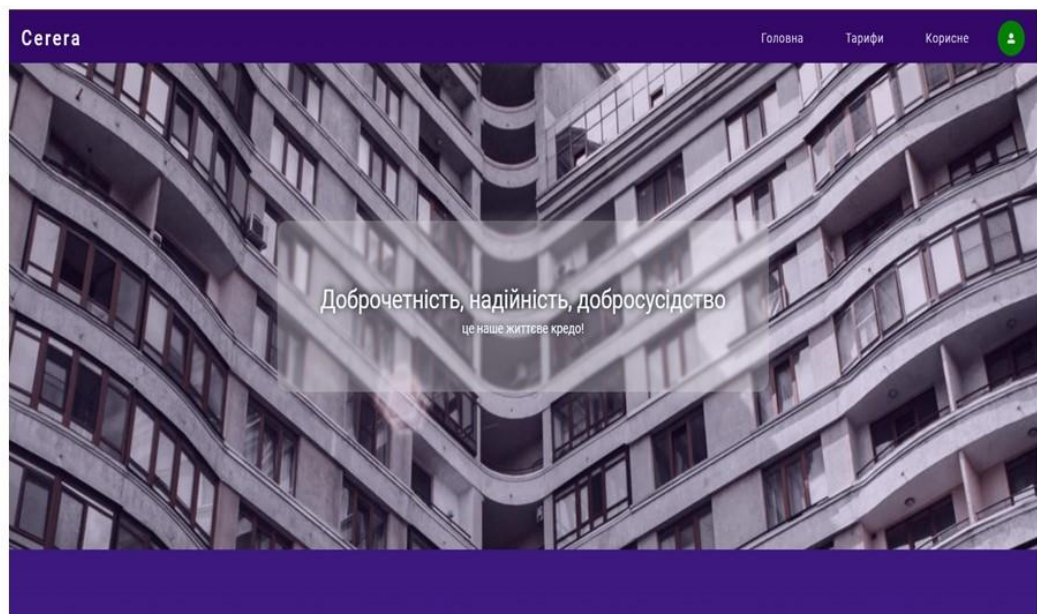


Рисунок Г.8 – Слайд

Графічний інтерфейс онлайн-сервісного застосунку



№8

Рисунок Г.9 – Слайд №9

Графічний інтерфейс онлайн-сервісного застосунку

Реєстрація

* Я: ☐ Власник ☐ Орендар

* Квартира №:

* Ім'я:

По-батькові:

* Прізвище:

* Email:

Телефон:

* Пароль:

* Підтвердження:

Вхід

* Email:

* Пароль:

Рисунок Г.10 – Слайд

Графічний інтерфейс онлайн-сервісного застосунку

Кв.	Ім'я	Email	Статус	Телефон	Дії
1	Карамболь Олена Василівна	ex@ecx.com	Власник	+380971111111	Редагувати Видалити
2	Щериця Анатолій Андрійович	owner@ecx.com	Власник	+380972222222	Редагувати Видалити
3	Живченко Світлана Миколаївна	ex9988@ecx.com	Власник	+380973333333	Редагувати Видалити
4	Андерсон Реджинальд Томас	asrox@ecx.com	Власник	+380974444444	Редагувати Видалити
5	Занечуй Віктор Романович	gist_body@ecx.com	Орендар	+380975555555	Редагувати Видалити

№10

Рисунок Г.11 – Слайд №11

Висновки

Для виконання дипломної було зроблено аналіз предметної області та досліджено існуючі проблеми та шляхи їх вирішення.

Також в рамках аналізу предметної області було проведено дослідження вже наявних на сьогодні рішень поставленої задачі. Було сформовано перелік переваг та недоліків для врахування їх при подальшому проектуванні.

За мету було поставлено спрощення комунікації між співвласниками та адміністрацією, прозорість рішень та використання спільних коштів.

Як результат розробки було створено онлайн-сервісний застосунок для співвласників багатоквартирного будинку, що вирішує всі поставлені задачі.

Рисунок Г.12 – Слайд №12