МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук,
управління та адміністрування
Кафедра інформаційних технологій

# Кваліфікаційна робота бакалавра

на тему: Розробка утиліти для збору звернень абонентів з подальшою обробкою в інформаційній системі

Виконав студент групи **КН-20**
спеціальності 122 Комп'ютерні науки
**Степанець Михайло Сергійович**

Керівник **асистент**
**Клепатська В.В.**

Консультант **д.т.н., професор**
**Казакова Н.Ф.**

Рецензент **заступник директора**
**- начальник відділу Департаменту**
**інформації та цифрових рішень**
**Одеської міської ради**
**Корчемний П.А.**

Одеса 2023

**ЗМІСТ**

## ТЕРМІНИ, СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

Архітектура веб-додатка – це організація компонентів, модулів та взаємодії між ними веб-додатка. Вона визначає структуру, розподіл функцій та взаємодію між клієнтом та сервером, що дозволяє забезпечити функціональність, безпеку та швидкодію веб-додатка.

База даних – це організована колекція даних, яка зберігається та управляється з використанням спеціального програмного забезпечення. Вона дозволяє зберігати, оновлювати, видаляти та виконувати пошук даних за певними критеріями.

Веб-сервер – це програмне забезпечення або апаратне обладнання, яке надає послуги обробки запитів і надсилання веб-сторінок клієнтам через протокол HTTP. Він відповідає за прийом запитів від клієнтів, обробку цих запитів та відправку відповідей, які можуть бути веб-сторінками, зображеннями, даними тощо.

Інформаційна система – це комплексно організована система, яка забезпечує збір, обробку, зберігання та передачу інформації для підтримки прийняття рішень та виконання різних функцій в організації. ІС може включати в себе бази даних, програми, комунікаційні мережі, апаратне забезпечення та інші компоненти.

Клієнт – це програмне забезпечення (зазвичай веб-браузер), що звертається до веб-сервера для отримання веб-сторінок та інших ресурсів.

Сервер – в контексті веб-технологій, сервер – це комп'ютер або програмне забезпечення, яке надає послуги обробки запитів та надсилання веб-сторінок клієнтам через протокол HTTP.

Apache – є одним з найпоширеніших веб-серверів. Він надає засоби для обробки запитів веб-сторінок і взаємодії з клієнтами.

Babel – це інструмент трансляції та компіляції JavaScript, який дозволяє розробникам використовувати сучасні функції та синтаксис JavaScript, незалежно

від підтримки браузерами. Babel перетворює нові версії JavaScript на сумісний зі старішими браузерами код, що дозволяє розробникам використовувати нові функціональні можливості мови без необхідності чекати на повну підтримку у всіх веб-переглядачах.

CSS – це мова стилів, яка використовується для оформлення веб-сторінок. Вона визначає зовнішній вигляд елементів HTML, таких як кольори, шрифти, розташування тощо.

HTML – це мова розмітки, яка використовується для створення структури та вмісту веб-сторінок. Вона визначає, які елементи (текст, зображення, посилання тощо) присутні на сторінці та як вони взаємодіють між собою.

I18N – це процес підготовки і адаптації програмного забезпечення чи веб-додатків для різних культур і мовних спільнот. I18N включає локалізацію (Localizaton) – переклад тексту програми на різні мови, а також забезпечення відповідності культурним нормам, форматам дати та часу, числам, валютам та іншим локальним вимогам.

IDEF0 – це методологія моделювання, яка використовує функціональні блоки для опису функцій, їх зв'язків та взаємодії в системі. Вона використовується для аналізу та проектування бізнес-процесів.

JS – це мова програмування, яка використовується для створення динамічного та інтерактивного веб-вмісту. Вона дозволяє керувати поведінкою елементів на веб-сторінці, виконувати різні операції та взаємодіяти з користувачем.

SADT – це методологія аналізу та проектування систем, яка використовує графічні символи для моделювання функцій, даних та управління в системі.

XSS – це тип вразливості веб-додатків, при якій зловмисник може впровадити та виконати власний зловредний скрипт на вразливій веб-сторінці. Це може стати причиною крадіжки даних користувачів, підриву безпеки системи або розповсюдження шкідливого вмісту. Щоб запобігти XSS-атакам, важливо правильно

обробляти та екранувати дані, які вводяться користувачем, перед їх відображенням на веб-сторінці.

БД – База даних.

ІС – Інформаційна система.

CSS – Cascading Style Sheets

I18N – Internationalization.

IDEF0 – Integration Definition for Function Modeling.

JS – JavaScript.

SADT – Structured Analysis and Design Technique.

XSS – Cross-Site Scripting.

# ВСТУП

Сучасний світ безперервно залежить від швидкого та ефективного доступу до інформації. Інформаційні технології присутні в усіх сферах життя. Створення інформаційних систем стає ключовим фактором для підтримки сучасного бізнесу, державних органів, наукових досліджень та майже всього повсякденного життя.

Метою кваліфікаційної роботи бакалавра є розробка утиліти для автоматизації процесу збору звернень абонентів з подальшою їх обробкою в інформаційній системі.

Сьогоднішній розвиток технологій надає безліч можливостей для створення інформаційних систем, починаючи від традиційних локальних систем і до розподілених хмарних рішень. Завдяки широкому спектру інструментів та платформ, розробники можуть створювати інформаційні системи, які адаптуються до змінних потреб індивідуальних користувачів та організацій.

Інформаційна система – це комплексний набір програмного забезпечення, апаратних засобів, баз даних та людських ресурсів, які співпрацюють для збору, зберігання, обробки та передачі інформації з метою забезпечення необхідних функцій та завдань. Ці системи можуть бути розроблені для різних цілей, включаючи автоматизацію бізнес-процесів, обробку великих обсягів даних, підтримку прийняття рішень та багато іншого.

Створення інформаційних систем – це складний та багатогранний процес, який вимагає поєднання знань з дисциплін, таких як програмування, бази даних, мережі, безпека інформації, дизайн і багато інших. Від успішної розробки та впровадження інформаційної системи залежить ефективність та конкурентоспроможність організації чи проєкту, а також здатність задовольняти потреби користувачів.

Розуміння основних принципів та методологій розробки інформаційних систем дозволить ефективно використовувати їх потенціал і побудувати майбутнє,

в якому інформація стане ще доступнішою, швидкою та силами трансформувати світ навколо.

Основні аспекти створення інформаційної системи, включають процеси аналізу вимог, проєктування архітектури, розробку програмного забезпечення та його тестування, впровадження та підтримку. Також, проаналізовано основні виклики та тенденції у цій галузі, а також переваги, які можуть бути отримані завдяки ефективним інформаційним системам.

Дана кваліфікаційна робота бакалавра, складається з 52 сторінок, 3 таблиць, 35 рисунків та 10 джерел посилань.

# 1 ОПИС ВИБОРУ АРХІТЕКТУРИ ТА ПРОГРАМНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ ВЕБ-СЕРВІСУ

## 1.1 Аналіз загальної архітектури веб-сервісу

Загальна архітектура веб-сервісу, як інформаційної системи, може варіюватися в залежності від конкретних потреб та вимог проєкту.

Архітектура веб-застосунок – це сукупність ідеї, концепцій та принципів, що визначають моделі, структури, функції та взаємозв'язок компонентів, що складають веб-застосунок. Вона встановлює загальну орієнтацію розробки, дозволяє визначити розподіл обов'язків між компонентами, спрощує розробку, супровід та масштабування системи. Архітектура веб-застосунка визначає структурні засади, які дозволяють веб-застосунку виконувати свої функції, забезпечувати безпеку, масштабованість та ефективність.

Для побудови правильної і надійної архітектури веб-застосунок важливо дотримуватися останніх стандартів та найкращих практик у цій галузі. Без такого дотримання існує ризик створення архітектури, яка не зможе ефективно розвиватися та вдосконалюватися, а також не задовольнить зростаючі потреби користувачів в галузі ІТ. Тому важливо орієнтуватися на актуальні стандарти, рекомендації та технології, які сприяють створенню масштабованих, гнучких і надійних веб-застосунків, забезпечуючи їх сумісність, безпеку та швидкість роботи.

Існує ряд класифікацій програмних систем залежно від їх архітектури. Одним з найпоширеніших типів архітектур є трьохрівнева клієнт-серверна архітектура. Ця архітектура виконує функцію розподілу завдань між різними компонентами системи з метою забезпечення ефективної обробки даних та зменшення навантаження на сервер. Інші типи архітектур включають централізовану архітектуру, архітектури файлових серверів, розподілені системи, веб-застосунки, а також сервіс-орієнтовану архітектуру. Кожна з цих архітектур має свої особливості та застосування в залежності від конкретних потреб та вимог проєкту.

Трирівнева клієнт-серверна архітектура включає три основні компоненти: клієнт, сервер та базу даних. Клієнт – це програмне забезпечення, що виконується на стороні користувача і забезпечує взаємодію з сервером через інтерфейс користувача. Сервер – це програмне забезпечення, що виконується на стороні сервера і відповідає за зберігання та обробку даних, а також взаємодію з клієнтом для обробки його запитів. База даних – це місце зберігання даних, до якого мають доступ як клієнт, так і сервер.

Трирівнева клієнт-серверна архітектура має декілька переваг. Вона дозволяє розподілити завдання між різними компонентами системи, зменшуючи навантаження на сервер та забезпечуючи ефективне використання ресурсів. Також вона надає масштабованість, що дозволяє розширювати систему шляхом додавання додаткових серверів або клієнтів. Архітектура також забезпечує безпеку, дозволяючи обмежити доступ до бази даних тільки авторизованим користувачам. Крім того, "тонкий" клієнт дозволяє знизити вартість апаратного забезпечення робочих станцій, оскільки більша частина обчислювального навантаження виконується на сервері.

Трирівнева клієнт-серверна архітектура широко застосовується в різних системах, включаючи веб-застосунок, електронні поштові сервіси, онлайн ігри та багатокористувацькі програми. Вона дозволяє ефективно розподіляти завдання між компонентами системи та забезпечувати швидкий доступ до даних для користувачів. Схема трирівневої архітектури "Клієнт-сервер" представлена на рис 8. Ця архітектура дозволяє виділити логічні рівні взаємодії між клієнтом, сервером та базою даних, що полегшує розробку, супровід та масштабування системи. Клієнтська частина забезпечує інтерфейс користувача, серверний рівень обробляє логіку застосунку та взаємодіє з базою даних для зберігання та отримання інформації. Така архітектура дозволяє створювати складні системи, які мають гнучкість, масштабованість та надійність.


Рисунок 1 – Трирівнева архітектура «Клієнт сервер»

Вибір трирівневої клієнт-серверної архітектури для реалізації інформаційної системи обґрунтовується її перевагами в масштабованості, гнучкості, безпеці, легкій інтеграції та розподіленості. Ця архітектура дозволяє ефективно керувати навантаженням, забезпечувати модульність і розподіл відповідальності між компонентами системи, а також забезпечує безпеку та захист даних. Завдяки своїй гнучкості, трирівнева архітектура легко підтримує зміни та розвиток системи, а її розподіленість дає змогу забезпечити відмовостійкість і доступність системи для користувачів. У підсумку, трирівнева клієнт-серверна архітектура є надійним і ефективним вибором для розробки та розгортання інформаційних систем.

### 1.2 Аналіз та обґрунтування вибору Веб-сервера

Для розробки застосунку для інформаційної системи був обраний веб-сервер Apache. Apache, веб-сервер з відкритим кодом, створений американським розробником програмного забезпечення Робертом Маккулом. Apache був випущений у 1995 році. На початку 2020-х років сервери Apache розгортали близько 30 відсотків Інтернет-контенту, поступаючись лише Nginx [1].

Як веб-сервер Apache відповідає за прийом запитів каталогу (HTTP) від користувачів Інтернету та надсилання їм потрібної інформації у формі файлів і вебсторінок. Значна частина програмного забезпечення та коду Інтернету розроблена для роботи разом із функціями Apache. Програмісти, які працюють над веб-застосунками, зазвичай використовують домашню версію Apache для попереднього перегляду та тестування коду. Apache також має безпечну функцію обміну файлами, що дозволяє користувачам розміщувати файли в кореневому каталозі програмного забезпечення Apache і ділитися ними з іншими користувачами. Вплив сервера Apache на спільноту програмного забезпечення з відкритим кодом частково пояснюється унікальною ліцензією, за якою розповсюджується програмне забезпечення від Apache Software Foundation [1].

Основні переваги веб-сервера Apache:

– доступність: Apache є безкоштовним програмним забезпеченням з відкритим вихідним кодом (це означає, що можна використовувати його безкоштовно і мати можливість модифікувати його за власними потребами;
– гнучкість налаштування: Apache має розширені можливості налаштування завдяки використанню конфігураційних файлів, це дозволяє точно налаштувати сервер під потреби проєкту;
– розширюваність: Apache має модульну структуру, що дозволяє підключати додатковий функціонал у вигляді модулів, це дозволяє легко додавати нові можливості до сервера без необхідності зміни основного коду;
– кросплатформеність: Apache працює на багатьох операційних системах, включаючи Windows, Unix, Linux та інші, це дозволяє використовувати Apache на будь-якій платформі, яка підтримується;
– сумісність: Apache підтримує багато мов програмування та скриптових мов, таких як PHP, Python, Ruby, Perl і багато інших, це дає можливість використовувати улюблені мови програмування для розробки веб-додатків під Apache;

– масштабованість – Apache підходить як для невеликих, так і для великих проєктів, може ефективно обслуговувати велику кількість запитів і підтримує розподілення навантаження для забезпечення високої продуктивності;
– підтримка спільноти: Apache підтримується великою та активною спільнотою розробників, де буде можливість знайти багато документації, ресурсів та форумів для отримання допомоги та ради щодо використання Apache.

Ці переваги роблять Apache привабливим вибором як веб-сервера для розробки веб-додатку, надаючи надійну та потужну основу, гнучкість налаштування та підтримку для різноманітних потреб проєкту.

### 1.3 Характеристика обраної системи управління базами даних

Microsoft SQL Server – це система керування реляційною базою даних (RDBMS), яка підтримує широкий спектр програм обробки транзакцій, бізнес-аналітики та аналітики в корпоративних IT-середовищах. Microsoft SQL Server є однією з трьох провідних на ринку технологій баз даних разом із Oracle Database і DB2 IBM [2].

Основні риси і особливості MS SQL:

– реляційна модель даних: MS SQL базується на реляційній моделі, що дозволяє організовувати дані у вигляді таблиць зі зв'язками між ними (це забезпечує структуроване та організоване зберігання і обробку даних);
– масштабованість: MS SQL підтримує масштабування як вертикально (збільшення обсягу ресурсів на одній машині) так і горизонтально (розподілення даних на кілька серверів), це дозволяє пристосувати базу даних під зростаючі потреби в обсязі даних і навантаженні;

- резервне копіювання і відновлення: MS SQL надає механізми резервного копіювання та відновлення, що дозволяють забезпечити безпеку та відновлення даних в разі випадку непередбачених ситуацій, таких як збої апаратного забезпечення чи помилкові операції;
- транзакційна підтримка: MS SQL забезпечує підтримку транзакцій, що дозволяють виконувати групу операцій як єдину неділену одиницю, забезпечуючи цілісність даних та відновлення в разі збоїв;
- мова запитів SQL: MS SQL використовує мову запитів SQL (Structured Query Language) для взаємодії з базою даних, дозволяє створювати складні запити для вибору, вставки, оновлення та видалення даних з бази;
- розширені можливості: MS SQL надає розширені функціональні можливості, такі як підтримка збережених процедур, функцій, тригерів, індексів та виділення даних. також підтримує реплікацію даних для забезпечення доступності та віддаленого резервного копіювання;
- інтеграція з іншими продуктами Microsoft: MS SQL має високу ступінь інтеграції з іншими продуктами Microsoft, такими як Windows Server, Active Directory, Azure та іншими, що робить її популярним вибором для розробки веб-додатків, підприємницьких систем та корпоративних рішень.

MS SQL є комерційним продуктом, і доступні різні версії з різними рівнями функціональності та вартості ліцензій. Вона підтримує як малий бізнес, так і великі підприємства з високими вимогами до продуктивності та надійності бази даних.

Загалом, MS SQL є потужним і надійним рішенням для управління базами даних, яке пропонує широкі можливості для розробників та адміністраторів, гнучкість у налаштуванні та інтеграцію з екосистемою продуктів Microsoft.

## 1.4 Аналіз та опис мови програмування для написання проєкту

### 1.4.1 Характеристика мови програмування Python

Python – це високорівнева, інтерпретована та універсальна динамічна мова програмування, яка зосереджується на зручності читання коду. Вона має менше кроків порівняно з Java і C. Python заснований у 1991 році розробником Гвідо Ван Россумом. Python входить до числа найпопулярніших і швидкозростаючих мов у світі; це потужна, гнучка та проста у використанні мова. Дану мову програмування використовують у багатьох організаціях, оскільки вона підтримує кілька парадигм програмування, також виконує автоматичне керування пам'яттю [3].

Python є мовою загального призначення, яка має велику кількість вбудованих функцій і бібліотек для швидкого та ефективного програмування. Вона підтримує різні парадигми програмування, такі як процедурне, об'єктно-орієнтоване і функціональне програмування, що надає розробникам велику свободу у виборі стилю програмування.

Python має широкий спектр застосувань, включаючи розробку веб-застосунків, наукові дослідження, аналіз даних, штучний інтелект, робототехніку та багато іншого. Вона має велику активну спільноту розробників, яка підтримує і розвиває мову.

Однією з переваг Python є його простота використання. Він має простий синтаксис і лаконічні конструкції, що дозволяє писати код швидко і ефективно. Python також відомий своєю читабельністю, що робить його код легко зрозумілим для інших розробників. Крім того, Python має велику кількість сторонніх бібліотек і модулів, що розширюють його можливості і дозволяють розробникам швидко розв'язувати різноманітні задачі. Ця екосистема бібліотек сприяє швидкому розробленню програм і полегшує процес роботи зі складними завданнями. Python є потужним і гнучким інструментом для розробки програмного забезпечення будь-

якого рівня складності. Він поєднує в собі простоту, ефективність і широкі можливості, роблячи його однією з найпопулярніших мов програмування у світі.

Під час розробки буде використовуватись фреймворк Flask, який стане основою веб-застосунку. Flask – це веб-фреймворк, модуль Python, який дозволяє легко розробляти веб-застосунки. Він має невелике ядро, яке легко розширити – це мікрофреймворк, який не містить ORM (Object Relational Manager) або подібних функцій [4]. Він надає базові функції і інструменти, щоб швидко розробляти веб-додатки, зосереджуючись на простоті і зрозумілості коду. Flask дозволяє розробникам будувати веб-додатки шляхом визначення маршрутів (routes), які відповідають URL-адресам та виконують певні функції при отриманні запиту на ці адреси. Він також підтримує вбудований сервер розробки, що дозволяє швидко перевіряти результати своєї роботи без необхідності налаштовувати окремий веб-сервер.

Jinja – це швидкий, виразний, розширюваний механізм створення шаблонів. Спеціальні заповнювачі в шаблоні дозволяють писати код, схожий на синтаксис Python. Потім до шаблону передаються дані для візуалізації остаточного документа [5].

Jinja включає в себе:

- спадкування та включення шаблонів;
- визначення та імпортування макросів в шаблонах;
- шаблони HTML, які використовують автоматичне екранування, щоб запобігти XSS від ненадійного введення користувача;
- ізольоване середовище, яке безпечно відтворює ненадійні шаблони;
- асинхронну підтримку для створення шаблонів, які автоматично обробляють синхронізацію та асинхронні функції без додаткового синтаксису;
- підтримку I18N з Babel;

- шаблони, що компілюються в оптимізований код Python миттєво та кешуються або можуть бути скомпільовані заздалегідь;
- винятки, які вказують на правильний рядок у шаблонах, щоб полегшити налагодження;
- розширювані фільтри, тести, функції та синтаксис [5].

За допомогою Jinja2 можна використовувати умовні вирази, цикли, змінні та інші конструкції для генерації HTML-коду з динамічними даними. Він також підтримує наслідування шаблонів, що дозволяє створювати загальні шаблони, які можна використовувати у багатьох сторінках веб-застосунку.

Взаємодія фреймфорків Flask і Jinja2 дає свої переваги. Flask використовує Jinja2 для рендерингу шаблонів і вставки динамічного контенту в статичні HTML-сторінки. Це дозволяє розробникам легко створювати красиві і динамічні веб-сторінки, що відповідають на запити користувачів. За допомогою Flask і Jinja2 можна швидко створити веб застосунок з розширеними можливостями і лаконічним дизайном, спрощуючи процес розробки і полегшуючи технічну реалізацію ідей.

### 1.4.2 Характеристика мови сценаріїв JavaScript

JavaScript – це мова сценаріїв або програмування, яка дозволяє впроваджувати складні функції на веб-сторінках. Частіше всього використовується для відображення не статичної інформації, а динамічно змінюваної, в тому числі своєчасні оновлення вмісту, інтерактивні карти, анімовані 2D/ 3D-графіка, музичні автомати з прокручуванням відео.[6].

Існують два методи вбудовування коду JavaScript у веб-сторінки: клієнтська та серверна сторона. Клієнтський JavaScript – це скрипти, які вбудовуються безпосередньо в HTML-сторінки та виконуються на браузері користувача. Вони можуть реагувати на події, спричинені користувачем, і виконувати різноманітні завдання, не вимагаючи звернення до веб-сервера.

JavaScript також підтримує взаємодію з сервером шляхом включення операторів мови у код, що виконується на серверній платформі. Цей метод дозволяє виконувати операції з базою даних та інші обчислення на сервері перед відправкою відповіді клієнту.

Однією з основних переваг JavaScript є те, що вона є широко підтримуваною мовою програмування з великою кількістю ресурсів, документації та підтримки спільноти розробників. Вона також сумісна з багатьма іншими технологіями та мовами програмування, що робить її гнучкою та розширюваною. Застосування JavaScript в розробці веб-застосунків дозволяє створювати інтерактивні та динамічні веб-сторінки, які можуть відрізнятися в залежності від дій користувача та обмінюватися даними з сервером без перезавантаження сторінки. Вона використовується в сфері веб-розробки для створення різноманітних застосунків, включаючи соціальні мережі, електронну комерцію, інтерактивні форми та багато іншого.

Одною з важливих доданих бібліотек є Chart.js. Це потужна і проста у використанні бібліотека JavaScript для створення інтерактивних графіків і діаграм на веб-сторінках. Вона надає широкий набір функцій і можливостей для візуалізації даних з легкістю. Основна перевага Chart.js полягає в його легкості використання. Ця бібліотека дозволяє швидко інтегрувати графіки у веб-сторінки за допомогою HTML-канвасу та декларативного синтаксису JavaScript. Це робить його доступним навіть для початківців без глибоких знань програмування. Chart.js підтримує різноманітні типи графіків, включаючи лінійні, кільцеві, стовпчасті, крапкові, площинні, розсіювальні графіки та інші. Можна налаштовувати стиль, кольори, маркери, легенду та інші аспекти графіків, щоб досягти потрібного вигляду та відповідати потребам.

Однією з сильних сторін Chart.js є його інтерактивність. Chart.js дає можливість налаштувати взаємодію з графіками, що в свою чергу дозволяє користувачам масштабувати, навігувати, отримувати додаткову інформацію під час наведення

на точки даних та виконувати інші дії. Це робить графіки більш динамічними та зручними для використання.

Крім того, Chart.js має добре документовану бібліотеку з великою спільнотою користувачів, що забезпечує підтримку, оновлення та покращення. В мережі Інтернет у вільному доступі, можна знайти документацію та приклади її використання. Загалом, Chart.js – це потужний інструмент для створення привабливих та інтерактивних графіків на веб-сторінках. Він дозволяє легко візуалізувати дані та надає великий контроль над їх виглядом і поведінкою.

### 1.4.3 Характеристика мови розмітки гіпертексту HTML

HTML (мова гіпертекстової розмітки) є основним будівельним блоком Інтернету. Визначає значення та структуру веб-контенту. Інші технології, окрім HTML, зазвичай використовуються для опису зовнішнього вигляду веб-сторінки [7]. HTML команди розмітки, спрямовані на веб-контент, які в свою чергу повідомляють програмному забезпеченню браузера про структуру документа і, як він повинен відображатися. Коли браузер читає документ, що містить HTML-розмітку, він визначає, як відобразити його на екрані, враховуючи HTML-елементи, вбудовані в документ.

HTML-документ – це текстовий файл, що містить інформацію, яку потрібно опублікувати. Він також містить вбудовані інструкції, які називаються елементами, що вказують веб-браузеру, як структурувати або представляти документ. Елементи HTML вказують на "розмітку" оточуючого тексту. Вони можуть вказувати значення вкладеної інформації (наприклад, цитата) або те, як вона має бути відображена (наприклад, напівжирним шрифтом). Елементи HTML зазвичай скла-

даються з пари тегів, які називаються контейнерними тегами, оскільки вміст знаходиться між ними. Проте деякі елементи, такі як тег горизонтального вирівнювання або розриву рядка, не мають відповідного закриваючого тегу.

HTML є найкращим вибором для розмітки веб-сторінок при розробці інформаційної системи з наступних причин: його універсальність та підтримка всіма веб-браузерами, простота використання та сумісність з іншими технологіями, підтримка семантики та мультимедіа, а також наявність широкого співтовариства розробників та обширних ресурсів для підтримки та навчання. Усі ці фактори роблять HTML зручним та потужним інструментом для створення веб-сторінок та застосунків.

### 1.4.4 Характеристика мови стилю сторінок CSS

Каскадні таблиці стилів (CSS) – це мова таблиць стилів, яка використовується для опису подання документа, написаного в HTML або XML (включаючи такі діалекти XML, як SVG, MathML або XHTML). CSS описує, як елементи мають відображатися на екрані, на папері, у мовленні чи на інших носіях [8].

CSS працює в парі з мовою розмітки HTML, яка відповідає за структуру сторінки. Завдяки CSS, веб-розробники можуть окремо керувати оформленням сторінки, не змінюючи саму розмітку. Один з основних принципів CSS – каскадність. Це означає, що стилі можуть бути задані на різних рівнях: глобально для всієї сторінки, для окремих елементів, а також вкладені один в одного. Це дозволяє визначати пріоритети стилів і забезпечує більш гнучкий підхід до оформлення.

CSS надає велику кількість властивостей, які можна використовувати для оформлення сторінки. Наприклад, можна змінити фоновий колір, встановити внутрішні та зовнішні відступи, вирівняти текст, змінити шрифт та його розмір, задати тіні, рамки, анімацію та багато іншого. CSS також підтримує різні методи вибору елементів для застосування стилів. Можна використовувати селектори,

щоб вибрати елементи за їх класами, ідентифікаторами, тегами, атрибутами чи їхньою ієрархією. Це дає можливість точно визначати, які елементи повинні бути стилізовані. CSS також підтримує можливість створення адаптивних стилів для різних пристроїв і розмірів екранів. Завдяки цьому, веб-сторінки можуть коректно відображатися як на десктоп-пристроях, так і на мобільних пристроях, забезпечуючи оптимальний досвід користувача незалежно від пристрою, на якому відбувається перегляд.

Загалом, CSS є потужним інструментом для стилізації веб-сторінок, що надає веб-розробникам велику свободу в оформленні та дизайні. Використовуючи CSS, можна створювати красиві та естетично привабливі веб-сторінки, забезпечуючи не тільки функціональність, але й естетичну привабливість для користувачів.

### 1.4.5 Характеристика діаграми Chart.js

Chart.js – це безкоштовна бібліотека JavaScript для створення діаграм на основі HTML. Це одна з найпростіших бібліотек візуалізації для JavaScript, яка містить такі вбудовані типи діаграм як [9]:

- діаграма розкиду;
- лінійна діаграма;
- гістограма;
- кругова діаграма;
- кругла діаграма;
- бульбашкова діаграма;
- діаграма площ;
- радарна діаграма;
- змішана діаграма.

Chart.js постачається зі стандартною конфігурацією за замовчуванням, що спрощує початок і отримання програми, готової до виробництва. Не налаштовуючи майже жодних параметрів, можна отримати діаграму. Наприклад, у Chart.js за замовчуванням увімкнено анімацію, тож можна миттєво привернути увагу за допомогою даних [10] .

Chart.js відтворює елементи діаграми на полотні HTML5, на відміну від кількох інших бібліотек діаграм, переважно на основі D3.js, які відтворюють як SVG. Візуалізація Canvas робить Chart.js дуже продуктивним, особливо для великих наборів даних і складних візуалізацій, які інакше вимагали б тисячі вузлів SVG у дереві DOM. У той же час візуалізація полотна не дозволяє використовувати стилі CSS, тому доведеться використовувати для цього вбудовані параметри або створити спеціальний плагін, або тип діаграми, щоб відобразити відповідно до цілі та задумів [10].

# 2 АНАЛІЗ ІСНУЮЧИХ АНАЛОГІВ УТИЛІТ ДЛЯ ЗБОРУ ЗВЕРНЕНЬ

Для того щоб сформулювати вимоги функцій, які повинен надавати веб-сервіс потрібно здійснити аналіз існуючих застосунків та визначити слабкі та сильні сторони. Нижче приведено декілька прикладів.

«Salesforce» – це одна з найпопулярніших CRM-систем у світі. Вона дозволяє компаніям збирати інформацію про клієнтів, включаючи контактні дані, історію звернень, вподобання та інше. За допомогою «Salesforce» компанії можуть відстежувати взаємодію з клієнтами, аналізувати дані та приймати заходи для поліпшення обслуговування.

«Salesforce» надає широкий спектр функціональних можливостей, включаючи управління контактами, продажами, маркетингом та обслуговуванням клієнтів. Компанії можуть створювати профілі клієнтів, вести журнал взаємодії з ними, планувати завдання та розклади зустрічей, а також відстежувати стан угод та процесів продажів.

«Salesforce» також має вбудовані аналітичні інструменти, які дозволяють компаніям аналізувати дані клієнтів, отримувати звіти та статистику про продажі, прогнозувати дохід та ідентифікувати ключові тренди. Компанії можуть використовувати ці дані для прийняття стратегічних рішень, удосконалення маркетингових кампаній та підвищення задоволеності клієнтів.

Завдяки своїм функціональним можливостям та гнучкості, «Salesforce» став однією з провідних CRM-систем, яка допомагає компаніям зростати, покращувати взаємини з клієнтами та досягати більшої ефективності у своїй діяльності. Головна сторінка CRM-системи «Salesforce» представлена на рис. 2.

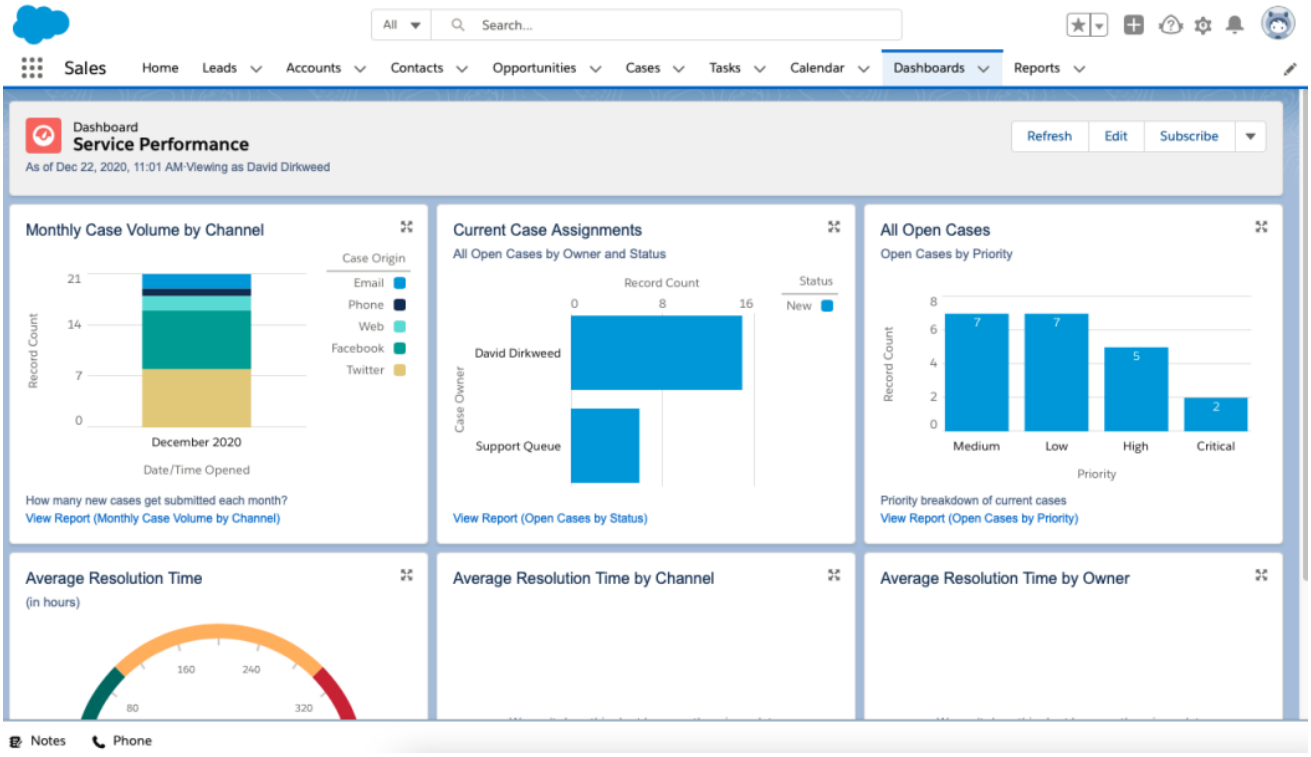


Рисунок 2 – Головна сторінка CRM-системи «Salesforce»

До переваг CRM-системи «Salesforce» можна віднести:

– широкий функціонал для управління контактами, продажами, маркетингом та обслуговуванням клієнтів;
– хмарна платформа, що забезпечує зручний доступ до даних з будь-якого місця та пристрою;
– гнучкість та адаптованість до потреб компаній за допомогою налаштувань та розширень;
– можливість інтеграції з іншими системами, покращуючи обмін даними та розширюючи функціональні можливості;
– вбудовані інструменти аналітики та звітності для отримання цінної інформації та прийняття обґрунтованих рішень.

До недоліків, в свою чергу можна віднести:

– висока вартість в порівнянні з іншими CRM-системами, особливо для невеликих компаній;
– складність в освоєнні та налаштуванні, що вимагає спеціалізованих знань або найму фахівців;
– залежність від Інтернет-з'єднання, оскільки «Salesforce» базується на хмарній інфраструктурі.

Але варто враховувати, що переваги та недоліки «Salesforce» можуть варіюватися в залежності від конкретних потреб і контексту компанії.

Наступна CRM-система це Bitrix24. Bitrix24 є хмарною CRM-платформою, яка поєднує інструменти для управління клієнтськими взаєминами, завданнями, комунікацією та багатьма іншими функціями в одному рішенні. Вона дозволяє компаніям збирати та зберігати інформацію про клієнтів, керувати взаємодією, відстежувати продажі та аналізувати ефективність маркетингових кампаній.

Bitrix24 включає модулі для управління контактами, угодами, пропозиціями, планами продажів, маркетинговими кампаніями, зверненнями клієнтів та іншими аспектами взаємодії з клієнтами. Вона також надає інструменти для організації роботи команди, включаючи завдання, проєкти, документи та внутрішню комунікацію. Bitrix24 має гнучкість та налаштованість, що в свою чергу дозволяє компаніям адаптувати систему до своїх потреб та бізнес-процесів. Вона доступна у різних варіантах, включаючи безкоштовну версію з базовим набором функцій, а також платні плани з розширеними можливостями та додатковими інтеграціями.

Це лише один приклад CRM-системи, яку компанії можуть використовувати для збору та аналізу даних про клієнтів, автоматизації процесів продажів та поліпшення взаємодії з клієнтами. Приклад інтерфейсу Bitrix24 представлений на рис. 10

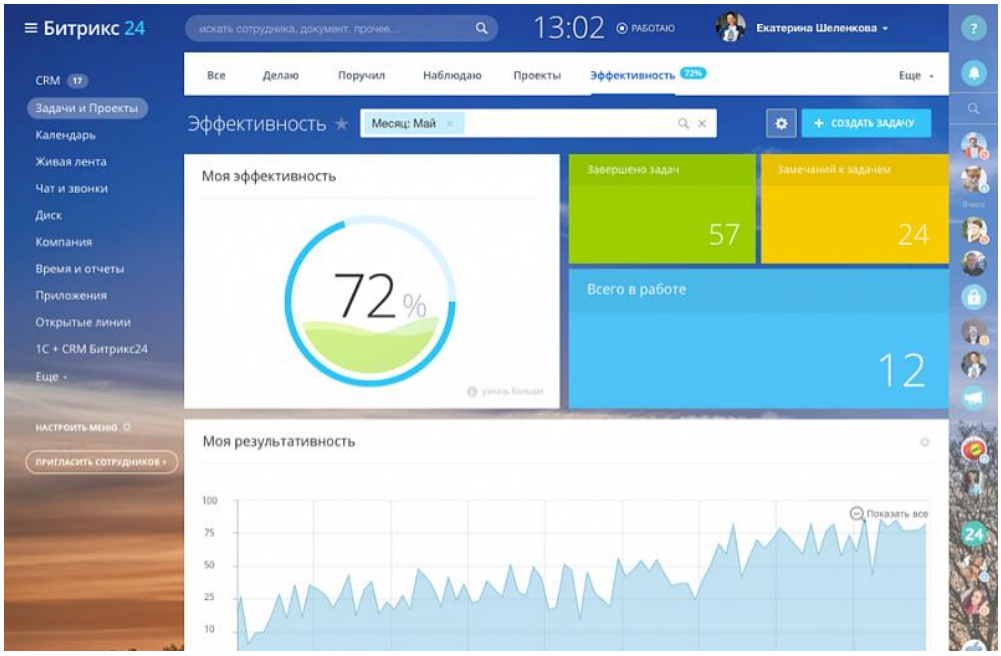


Рисунок 3 – Сторінка з аналізом CRM-системи Bitrix24

Ще одним прикладом буде HubSpot. HubSpot є однією з провідних CRM-платформ, яка надає комплексні рішення для управління клієнтськими взаєминами. Вона спрощує процес залучення, утримання та зростання клієнтської бази, а також автоматизує маркетингові та продажні процеси.

Основні компоненти та функціональні можливості HubSpot включають:

– управління контактами та компаніями – HubSpot дозволяє зберігати та організовувати дані про клієнтів, контакти та компанії на одній платформі (можна відстежувати комунікацію з клієнтами, збирати важливі дані та отримувати повний огляд профілю);
– управління угодами – можливість створювати, відстежувати та керувати угодами протягом усього життєвого циклу (HubSpot надає можливість створювати та назначати завдання для кожної угоди, встановлювати етапи та контролювати прогрес);

- маркетингова автоматизація – HubSpot має ряд потужних інструментів для автоматизації маркетингових кампаній (можна створювати електронні листи, лендінги, форми збору даних, соціальні медіа-публікації та інше. Також є можливість відстежувати та аналізувати результати компаній, щоб покращити ефективність);
- сервісне управління – HubSpot допомагає керувати заявками та запитами від клієнтів (можливо організовувати, відстежувати та вирішувати проблеми клієнтів через систему тікетів. Також можна надавати підтримку через чат-боти та забезпечувати зручну комунікацію з клієнтами);
- аналітика та звіти – HubSpot надає розширені можливості аналізу та звітності (можна відстежувати метрики продажів, маркетингу та сервісного обслуговування, отримувати звіти про ефективність кампаній та прогнозувати результати).

Перевагами HubSpot можна назвати:

- інтегрована платформа з комплексним функціоналом;
- зручний та інтуїтивно зрозумілий інтерфейс;
- широкі можливості для автоматизації маркетингу та продажів;
- доступність безкоштовної версії з базовими функціями.

Недоліки HubSpot:

- деякі функції та додаткові інструменти можуть бути доступні лише у платних планах;
- вимагає часу та навчання для повного освоєння та використання всіх функцій;
- залежність від хмарної платформи та Інтернет-з'єднання.

HubSpot є популярним вибором для багатьох компаній завдяки своїй функціональності та можливостям для управління клієнтськими взаєминами. Інтерфейс застосунку можна переглянути на рис. 11

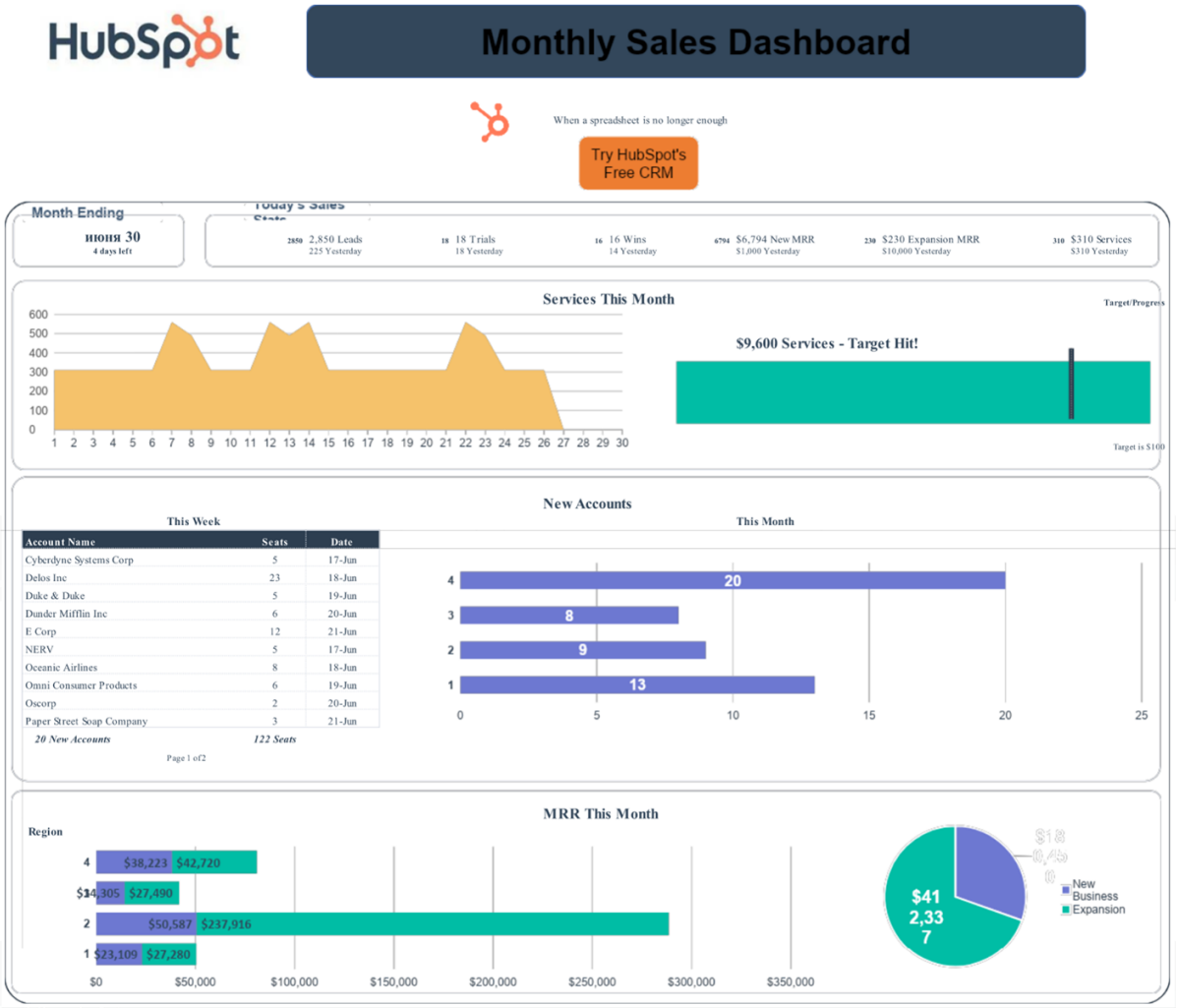


Рисунок 4 – Сторінка з аналізом CRM-системи HubSpot

# 3 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

## 3.1 Проєктування веб-застосунку за допомогою методології SADT

### 3.1.1 Використання мови IDEF0 для проєктування веб-застосунку

При проєктуванні веб-застосунку була обрана методологія функціонального моделювання SADT (Structured Analysis and Design Technique) – стандарт IDEF0.

Нотації SADT складаються з діаграм зі стрілками (блоків), з чотирма стрілками на кожній стороні, визначеними як: вхід, вихід, контроль і механізм, і однією діяльністю посередині. Їх визначення складаються з наступного:

- діяльність – це будь-яка функція або процес, який слугує для перетворення входів у виходи;
- вхідні дані: дані/інформація, необхідні для того, щоб розпочати процес перетворення;
- вихід: дані/інформація, вироблені діяльністю в результаті цього перетворення;
- контроль: будь-яке обмеження, яке певним чином впливає на поведінку діяльності;
- механізм: люди, ресурси або будь-які засоби, необхідні для виконання діяльності.

Контекстна діаграма – це високорівнева модель, яка показує систему або процес у контексті зовнішнього середовища. Це перше представлення системи, що дозволяє зрозуміти взаємодію із зовнішніми факторами і позначити межі системи. Вона не надає детальної інформації про внутрішню структуру і функціонування системи, але слугує загальним оглядом і контекстом для розуміння системи та її взаємодії із зовнішніми факторами. Вона є відправною точкою для більш детального моделювання та аналізу функцій і процесів системи. Контекстна діаграма інформаційної системи наведена на рис. 12.

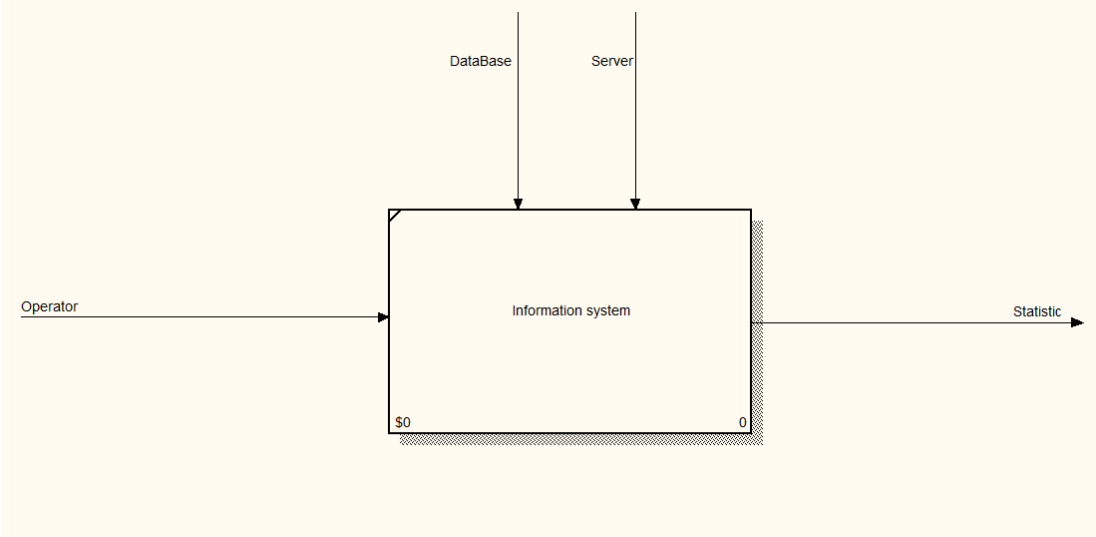


Рисунок 5 – Контекстна діаграма «Інформаційної системи»

На контекстній діаграмі відображається головний процес. На вході є інформація про клієнта. Передається звідки було звернення, та яка була причина звернення.

Після того, як система визначена як єдине ціле, її розбивають на менші частини (проводиться декомпозиція системи). Декомпозиція – це процес розбиття системи на менші компоненти та підзадачі з метою більш детального аналізу та розуміння її структури та функцій. Це дозволяє уточнити моделі і створити більш детальні функціональні моделі для кожної частини системи. Декомпозиція полегшує системний аналіз, проєктування та документування, розбиваючи складні системи на більш керовані та зрозумілі компоненти. Кожен рівень декомпозиції забезпечує глибший рівень деталізації, що дозволяє точніше ідентифікувати функції, процеси та взаємозв'язки в системі. На різних рівнях декомпозиції синтаксис, що використовується у функціональній моделі, може бути однаковим, але може бути модифікований для врахування більш детальної інформації про компоненти системи.

Діаграма декомпозиції розробки «Інформаційна система» (див.рис. 6).

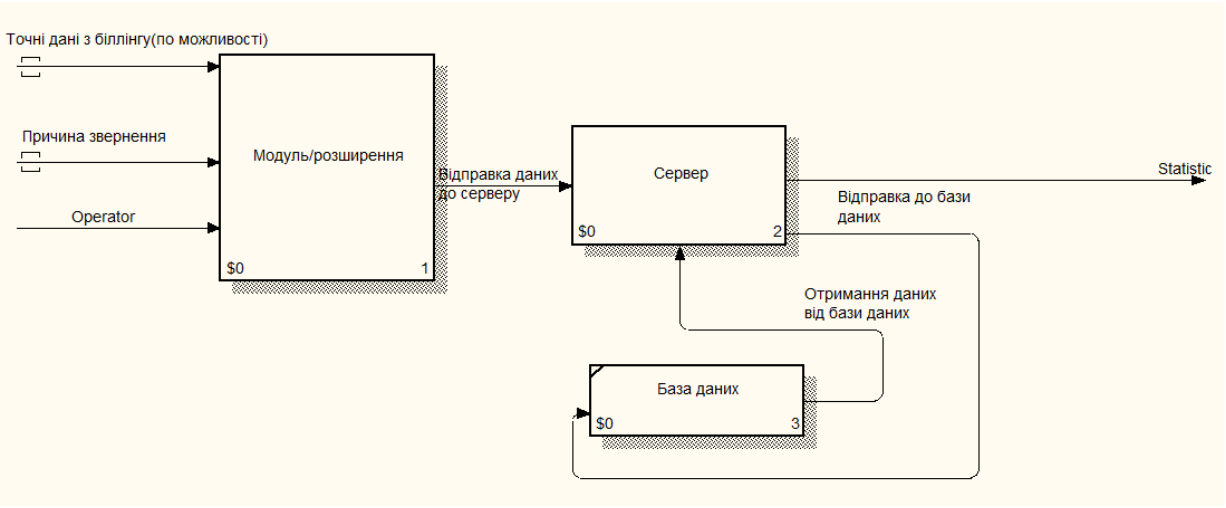


Рисунок 6 – Діаграма декомпозиції розробки «Інформаційна система»

Діаграма декомпозиції блоку «Модуль/розширення» наведена на рис.14.

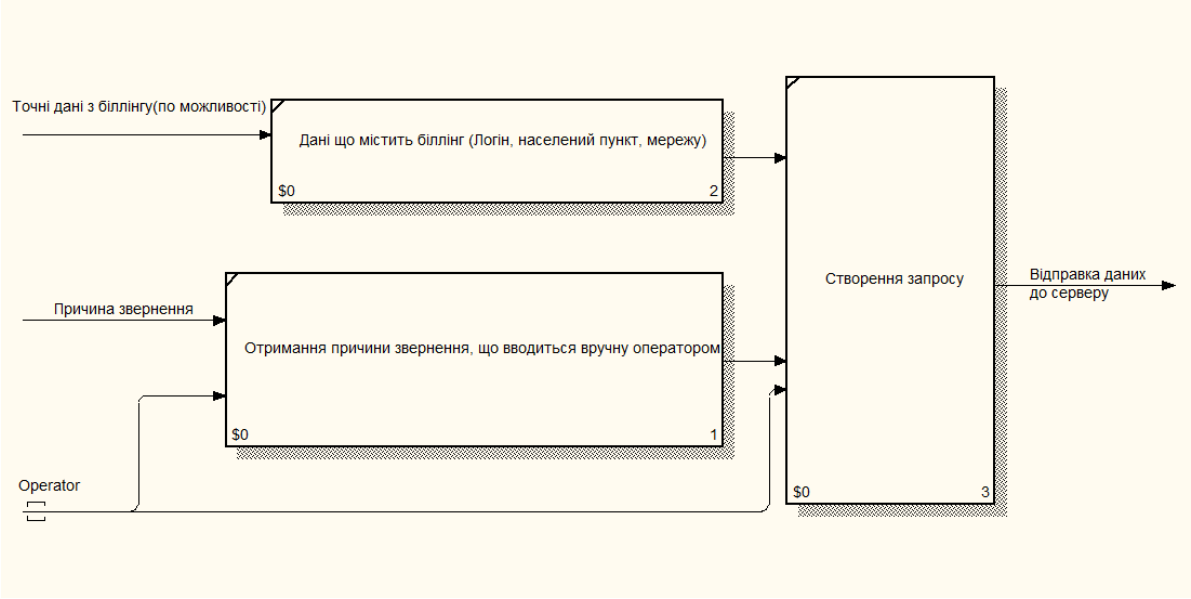


Рисунок 7 – Діаграма декомпозиції блоку «Модуль/розширення»

Діаграма декомпозиції блоку «База даних» наведена на рис. 158.

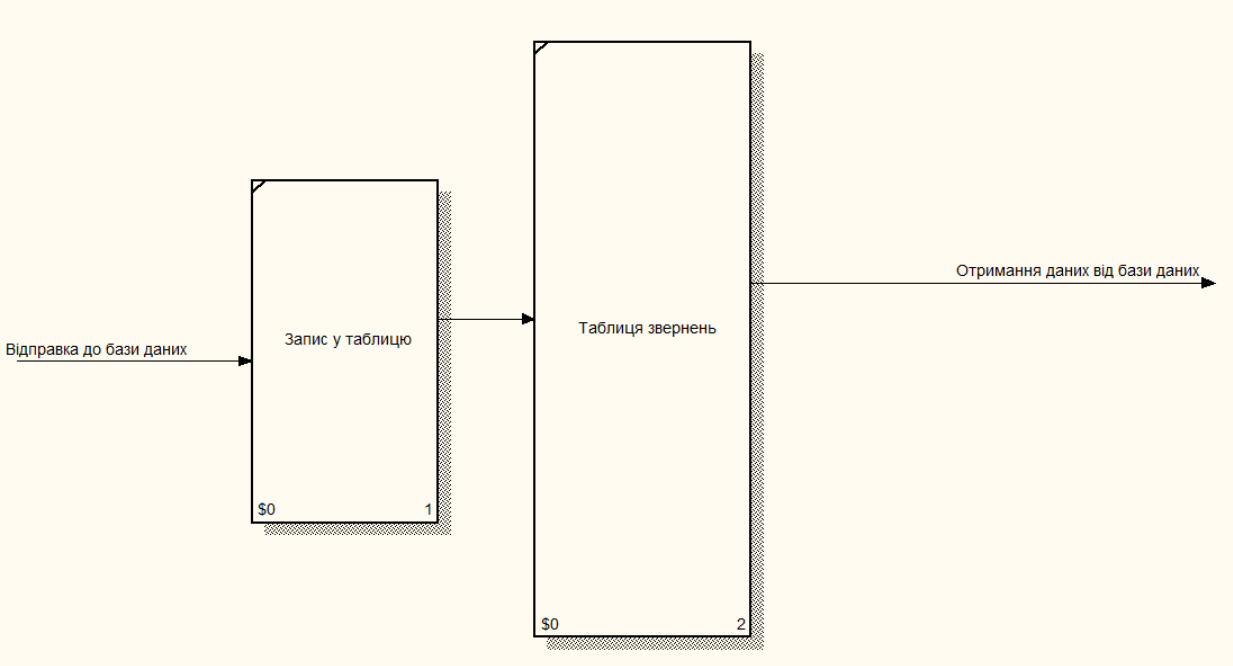


Рисунок 8 – Діаграма декомпозиції блоку «База даних»

### 3.2 Проєктування функціональних вимог і БД

Для представлення логічної структури веб-застосунку обрана модель «Сутність-зв’язок». Сутність-зв'язок (Entity-Relationship) – це методологія моделювання даних, яка використовується для опису сутностей (об'єктів) та їх взаємозв'язків у базі даних. Вона широко застосовується при розробці інформаційних систем і баз даних.

Основна ідея сутність-зв'язок полягає в представленні реальних об'єктів або понять у вигляді сутностей, а їх взаємозв'язки – у вигляді зв'язків між цими сутностями. Сутність описує конкретний об'єкт або поняття, яке може бути ідентифіковане та збережене в базі даних. Зв'язок визначає відношення і взаємодії між сутностями.

У моделі сутність-зв'язок використовуються такі основні елементи:

– сутність (Entity) – описує конкретний об'єкт або поняття, яке має унікальні атрибути (властивості), сутність зазвичай представляється у вигляді прямокутника з назвою;
– атрибут (Attribute) – характеристика або властивість сутності, де кожна має набір атрибутів, які описують її властивості;
– зв'язок (Relationship) – описує відношення між двома або більше сутностями, де зв'язок може бути однонаправленим або двонаправленим і мати різні типи (наприклад, один-до-одного, один-до-багатьох, багато-до-багатьох).;
– кардинальність (Cardinality) – визначає кількість пов'язаних сутностей у відношенні, наприклад, один-до-багатьох означає, що одна сутність пов'язана з декількома іншими сутностями, а багато-до-багатьох означає, що декілька сутностей пов'язані з декількома іншими сутностями.

Модель сутність-зв'язок дозволяє візуалізувати структуру даних і взаємозв'язки між ними, що спрощує розуміння та проєктування баз даних. Це допомагає розробникам і аналітикам створювати ефективні та добре організовані інформаційні системи. Для бази даних веб-застосунку виділимо наступні сутності: Net (мережі), Problems (типи звернень), AppealsInfo (головна таблиця, де зберігається інформація о зверненнях впродовж місяця). Кожна сутність повинна містити атрибути або групи атрибутів, які однозначно ідентифікують кожен екземпляр цієї сутності. Такі атрибути називаються первинними ключами. При проєктуванні бази даних веб-застосунку були розглянути сутності та їх атрибути.

Сутність «Net» (мережі) містить наступні атрибути (табл. 1):

– id_net – ідентифікатор мережі;
– name_net – назва мережі.
– status_net – статус мережі.

Таблиця 1 – Сутність «Net»

| № | Атрибут | Тип |
|---|---|---|
| 1 | 2 | 3 |
| 1 | id_net | smallint |
| 2 | name_net | varchar(25) |
| 3 | status_net | bit |

Сутність «Problems» (типи звернень) містить наступні атрибути (табл. 2):

– id_ problem – ідентифікатор звернення;

– name_ problem – назва звернення.

– status_ problem – статус звернення.

Таблиця 2 – Сутність «Problems»

| № | Атрибут | Тип |
|---|---|---|
| 1 | 2 | 3 |
| 1 | id_ problem | smallint |
| 2 | name_ problem | varchar(25) |
| 3 | status_ problem | bit |

Сутність «AppealsInfo» містить наступні атрибути (табл. 3):

– appeal_Y_M_D – точна дата;

– appeal_type – тип звернення;

– appeal_source – джерело звернення;

– appeal_id_problem – ідентифікатор звернення;

– appeal_login – ідентифікаційний номер абонента, логін;

– appeal_id_net – ідентифікатор мережі;

– appeal_region – назва регіону, де зареєстрований абонент;

– appeal_locality – назва населеного пункту, де зареєстрований абонент.

Таблиця 3 – Сутність «AppealsInfo»

| № | Атрибут | Тип |
|---|---|---|
| 1 | 2 | 3 |
| 1 | appeal_Y_M_D | DATE |
| 2 | appeal_type | varchar(7) |
| 3 | appeal_source | varchar(6) |
| 4 | appeal_id_problem | smallint references Problems |
| 5 | appeal_login | varchar(20) |
| 7 | appeal_id_net | smallint references Net |
| 8 | appeal_region | varchar(50) |
| 9 | appeal_locality | varchar(50) |

Після того, як були визначили всі основні сутності та атрибути бази даних, що розробляється для веб-застосунку, можна визначити зв'язки між сутностями. Представимо базу даних у вигляді моделі «сутність-зв'язок». Тип зв'язку, що було використано – «один-до-багатьох». Діаграма «сутність-зв'язок» для веб-застосунку представлена на рис. 16.

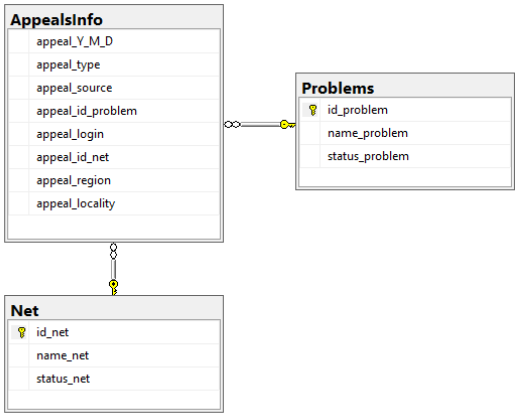


Рисунок 9 – Діаграмма «сутність зв’язок»

На основі проведеного аналізу та розробки діаграми "сутність-зв'язок" для веб-застосунка можна зробити такі висновки. База даних являє собою добре структуровану систему, що відображає основні сутності, їхні атрибути та зв'язки між ними. Діаграма «сутність-зв'язок» є основою для створення та подальшого розвитку бази даних, що відповідає потребам веб-застосунку.

# 4 ОПИС РОБОТИ УТИЛІТ ТА ЇХ ВЗАЄМОДІЯ З ІНФОРМАЦІЙНОЮ СИСТЕМОЮ

Почати опис, слід з модулю, за допомогою якого й додаються дані до бази даних. Модуль був зроблений у якості розширення для браузеру, оскільки був відсутній доступ до вихідного коду. При можливості, в подальшому, можна буде додати цей модуль як частину самого білінгу. Він складається з 2-х елементів, кнопки, та модального вікна.

Кнопка має помаранчевий колір, котрий буде виділятись між інших кнопок на сайті, але не буде виходити з палітри кольорів самого сайту. Вона розташована між полем для пошуку та елементом профілю. Позиція була обрана таким чином, щоб її було зручно натискати як у звичайному перегляді сторінки, так і при використанні телефонії, що також йде як розширення у браузері та закриває приблизно 2 чверті усієї сторінки. Щоб було зручно записати звернення не закриваючи вікно телефонії, була обрана саме така позиція.

Положення кнопки не зміниться, будь-то профіль абонента, чи будь-яка інша сторінка сайту, бо на цьому також була акцентована увага. Наприклад якщо буде телефонувати співробітник, щоб була можливість записати його звернення також. Нижче вказано приклад вигляду кнопки у абонентському профілі, рис. 17, та на іншій сторінці сайту, рис. 18.

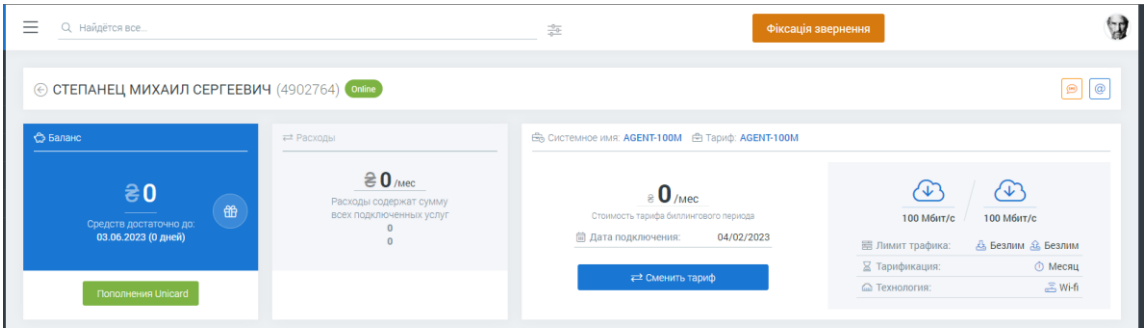


Рисунок 10 – Положення кнопки у обліковому записі абонента

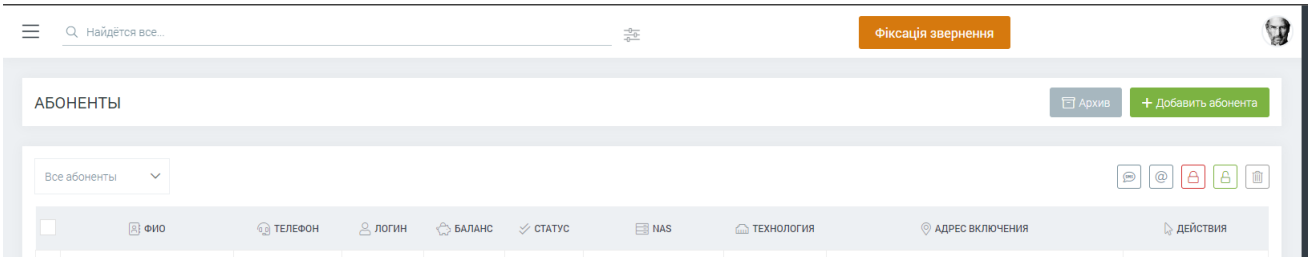


Рисунок 11 – Положення кнопки на будь-якій іншій сторінці сайту

Далі варто розглянути модальне вікно. Його вигляд можна побачити на рис. 19. Він складається з блоку заголовку, кнопки закриття вікна, двох радіокнопок поміщених у групу «Тип звернення», двох радіокнопок поміщених у групу «Джерело звернення», випадаючого меню з елементами звернень, випадаючого меню з елементами мереж, та кнопки відправки запиту на сервер. Для більш комфортного та вдалого інтегрування, вікно було зроблено під дизайн самого білінгу. Радіокнопки були стилізовані під звичайні кнопки, та їх кольори були розділені на синій та зелений, для більш приємного використання, та меншої заплутаності в інтерфейсі.

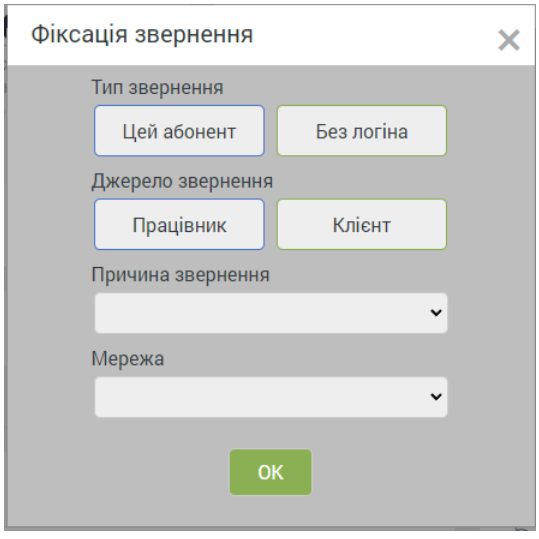


Рисунок 12 – Вид модального вікна

Також було реалізовано стандартні положення усіх елементів. При знаходженні на сторінці абонентської картки, рис. 20, стандартними положеннями буде:

– тип звернення: цей абонент;
– джерело звернення: клієнт;
– причина звернення: фінансове питання / особистий кабінет;
– мережа: обирається автоматично за білінгом.

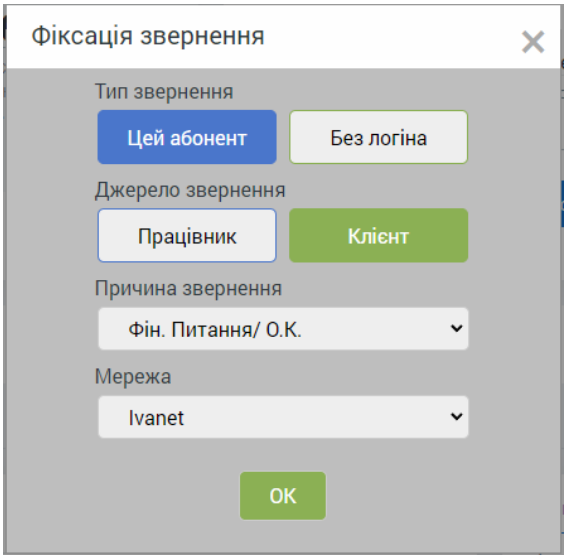


Рисунок 13 – Вигляд модального вікна у профілі абонента

При знаходження на якійсь іншій сторінці сайту, рис 21, будуть такі стандартні положення:

– тип звернення: без логіна;
– джерело звернення: співробітник;
– причина звернення: роботи;
– мережа: N/A.

Також, при знаходженні не у профілі абонента, кнопка «Цей абонент» буде не активна, та буде характерного сірого кольору.

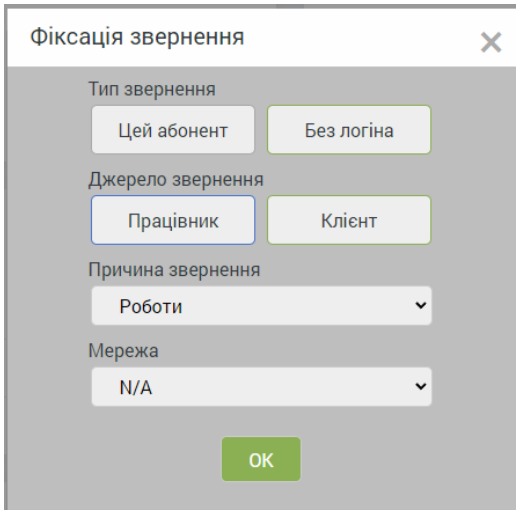


Рисунок 14 – Вигляд модального вікна не у профілі абонента

При необхідності, оператор може змінити стандартні параметри, змінюючи положення кнопок та обираючи варіанти з випадаючих меню рис. 22 та рис. 23

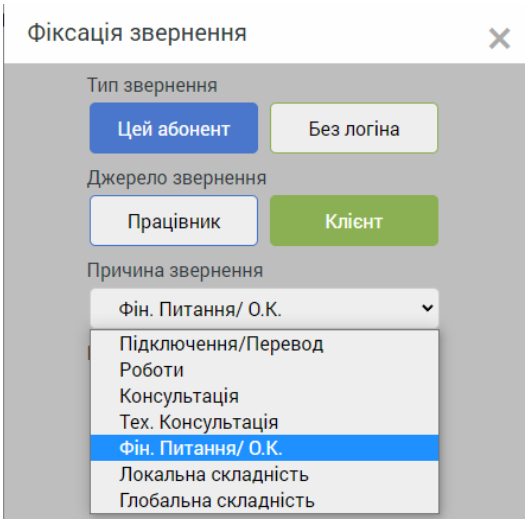


Рисунок 15 – Зміна положення кнопок та випадаючого меню

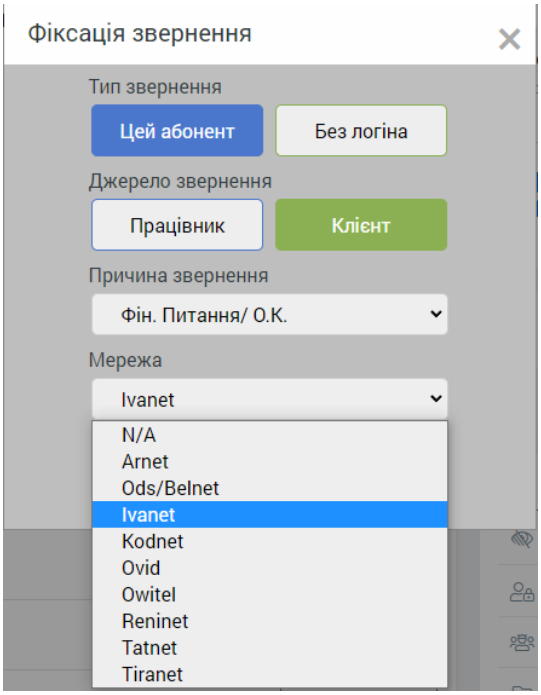


Рисунок 16 – Зміна положення кнопок та випадаючого меню

При натисканні на кнопку «ОК» буде надіслано запит у форматі json. Буде надсилатись інформація по населеному пункту, регіону, мережі, типу проблеми та інше.

```
var data = {
     get_type:have_login,
     get_source:is_abon,
     get_problem:problem_type,
     get_login:login_s,
     get_net:net_type,
     get_region:region,
     get_locality:locality
     };
 var jsonData = JSON.stringify(data);
 xhr.send(jsonData);
```

Далі необхідно розглянути саму інформаційну систему. Вона була зроблена за допомогою фреймворку Flask. Він був обраний завдяки його потужності та лег-

кості у використанні. Для візуалізації було використано бібліотеку Jinja2 для шаблонізації самого сайту. Те, як виглядає сторінка сайту можна побачити на рис. 24 та рис. 25

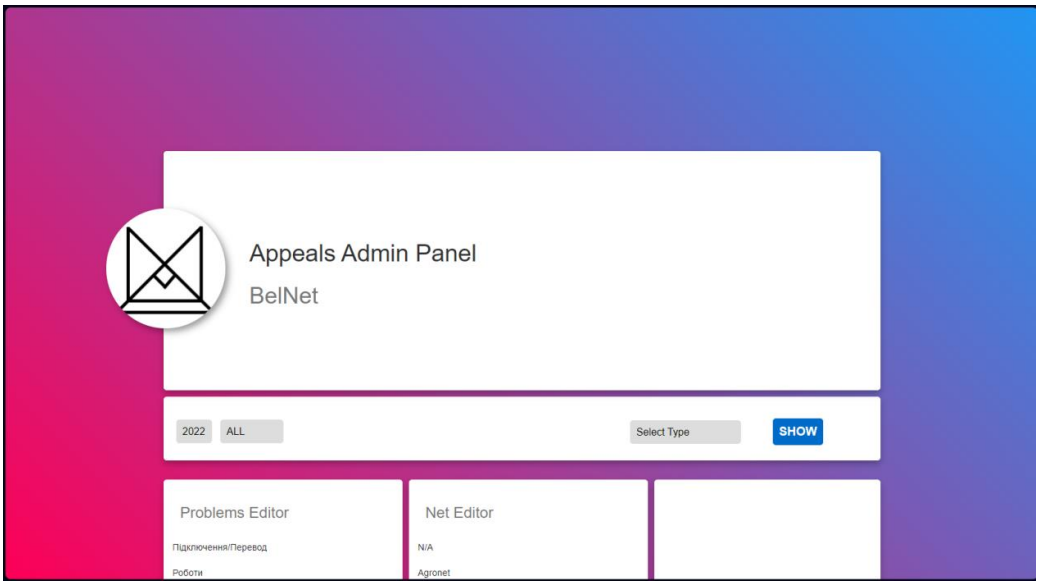


Рисунок 17 – Інтерфейс сайту

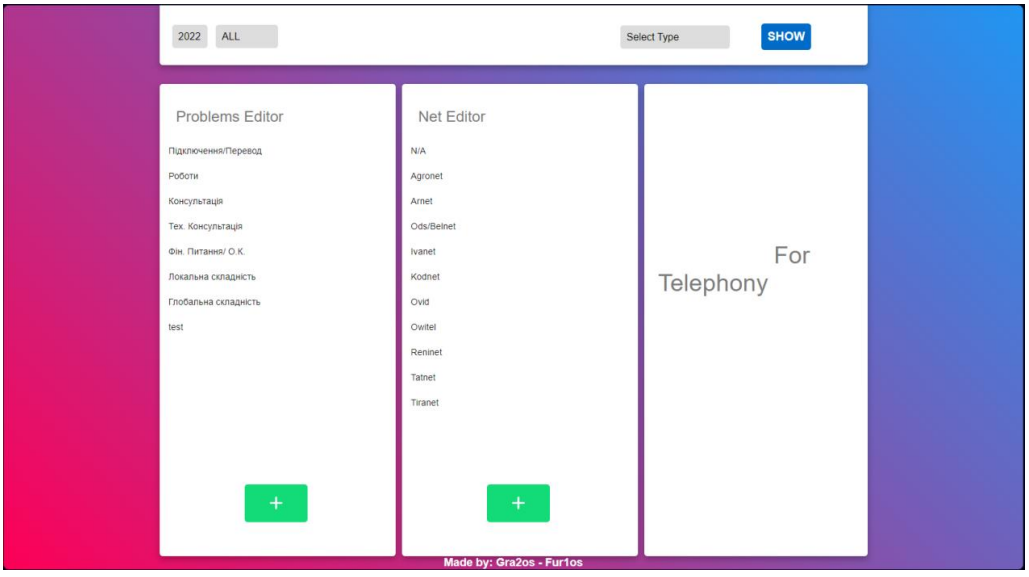


Рисунок 18 – Інтерфейс сайту

Одною з ключових позицій сайту є редактор звернень та мереж. Знайти можна у нижній частині сайту. Виглядає блок як вказано на рис. 26.

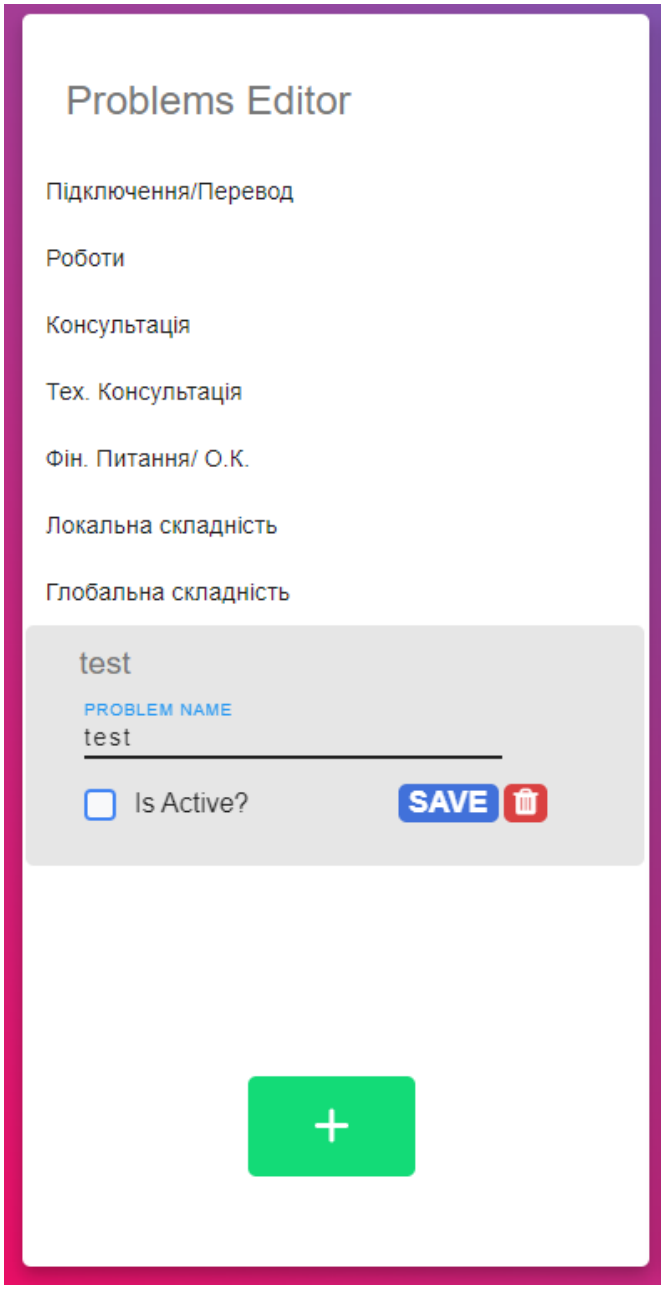


Рисунок 19 – Блок редагування звернень

У даному блоці є список усіх нині існуючих, у базі даних, типів звернень. Якщо натиснути на одне зі звернень, блок розшириться, де будуть елементи керування цим зверненням.

У полі «Problem name» буде відображатись нинішнє ім’я проблеми, його можна буде виправити у цьому ж текстовому полі, але залишити поле пустим не можливо. Нижче розташований чекбокс «Is Active?», він вказує на нинішній активний стан проблеми, якщо стоїть галочка, тобто він активний, тоді ця проблема буде показуватись у оператора в білінгу, якщо ж воно буде не активне, тоді проблема буде відображатись лише на цьому сайті, де при необхідності можна її увімкнути знов. Поруч розташована кнопка «Save».

При її натисканні посеред екрану з’явиться модальне окно для підтвердження змін у параметрах елементу, рис. 27. У цьому вікну буде ще можливість перевірити усі дані, та у цьому ж вікні переробити параметри. Далі можна натиснути кнопку «Yes» для відправки запиту на сервер. Ще є варіант, кнопка «Cancel», для відміни змін та закриття вікна.

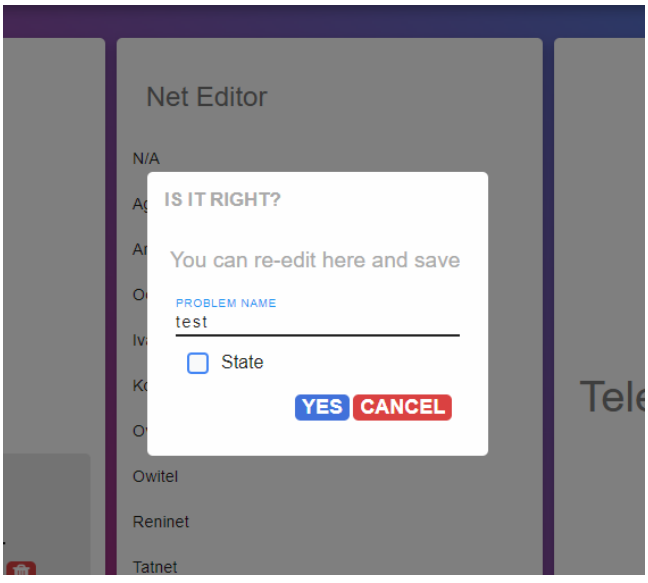


Рисунок 20 – Вікно підтвердження оновлення інформації.

Поруч з кнопкою «Save» розташована кнопка видалення, вона позначена характерною іконкою смітника. При її натисканні виведеться вікно підтвердження операції, рис. 28. У тому вікні присутні дві кнопки «Yes» та «Cancel», котрі підтверджують та відхиляють операцію. Після підтвердження, буде зроблено окремий поток для операції видалення, з відстрочкою до 12 години ночі наступного дня. Це зроблено, щоб під час роботи операторів не зник один з параметрів, та не було зайвих питань.

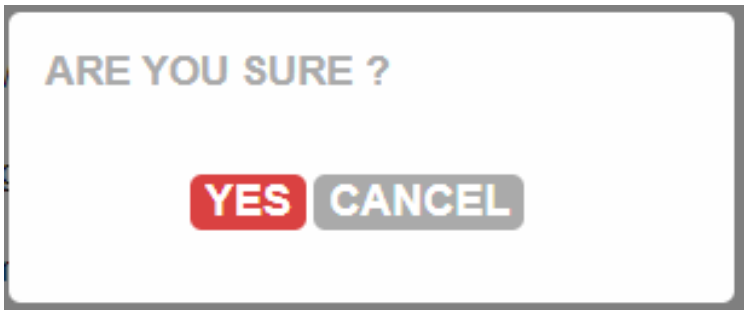


Рисунок 21 – Вікно підтвердження видалення інформації.

Наступною кнопкою є кнопка додавання елементів до бази даних «Add», рис. 29. При натисканні відкриється модальне вікно для додавання проблем. У ньому буде текстове поле, де можна ввести ім’я нового звернення. Також можна буде одразу встановити стан цього звернення.


Рисунок 22 – Кнопка додавання елементів

Та при натисканні кнопки «Add», нове звернення одразу буде додано до бази даних, рис. 30.

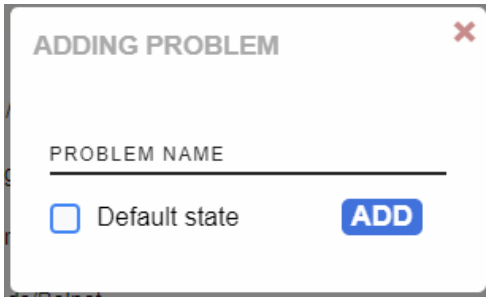


Рисунок 23 – Вікно додавання звернень

Далі необхідно розглянути меню вибору графіків, та самі графіки. Зверху екрану є вузький блок з вибором року та місяця. Є можливість обрати точний місяць, в котрому є бажання обробити інформацію, або ж обробити усі дані за рік. Праворуч можна зафіксувати випадаюче меню, де можна обрати який тип обробки необхідний. В даний час є лише два варіанти: «Загальна кількість» (рис. 31) та «Причина звернення» (рис. 32).

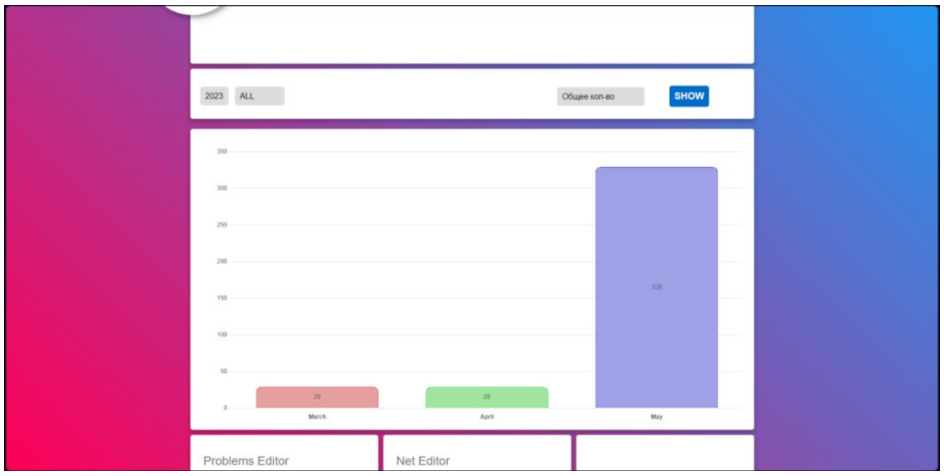


Рисунок 24 – Графік «Загальна кількість» за весь рік

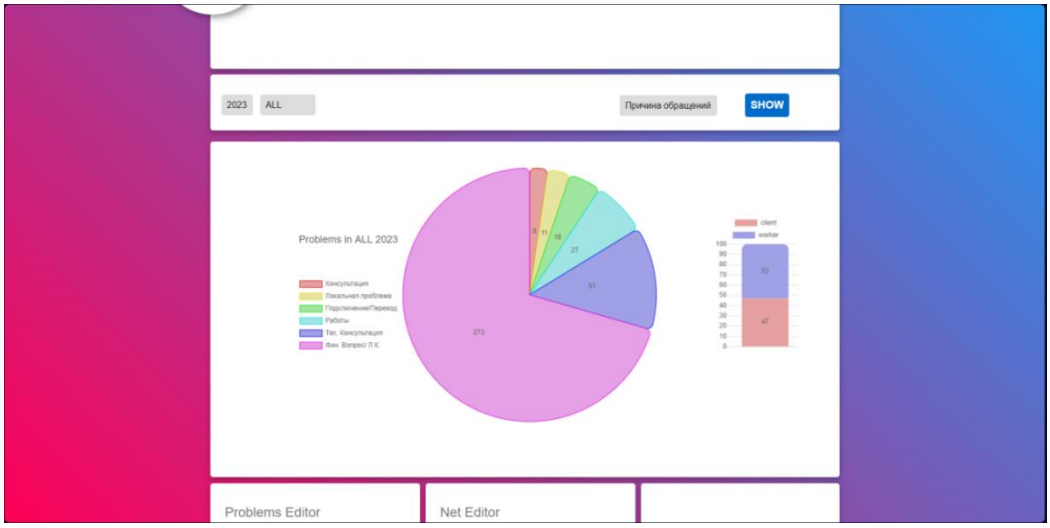


Рисунок 25 – Діаграма «Причина звернень» за весь рік

У першому варіанті, «Загальна кількість», буде відображено вертикальні графіки згідно звернень абонентів. У випадку, коли обрані усі місяці, тобто необхідна інформація за рік, тоді звернення будуть оброблюватись по місячно, рис. 31

У випадку коли за окремий місяць, тоді буде щодобові графіки, рис. 33.

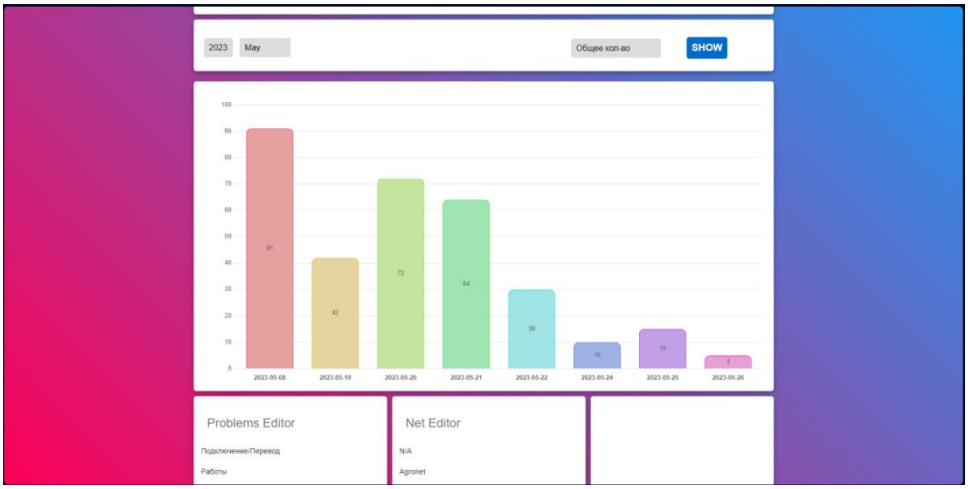


Рисунок 26 – Графік «Загальна кількість» за окремий місяць

При виборі типу графіку «Причина звернень», буде виведено кругову діаграму, де будуть виведені усі причини, що були за місяць та їх кількість. Також, поруч буде ще один вертикальний графік, що буде у процентному співвідношенні вказувати звернення від клієнтів та від працівників. Знов таки є можливість обрати звернення за місяць (рис. 34), або за увесь рік (рис. 32).

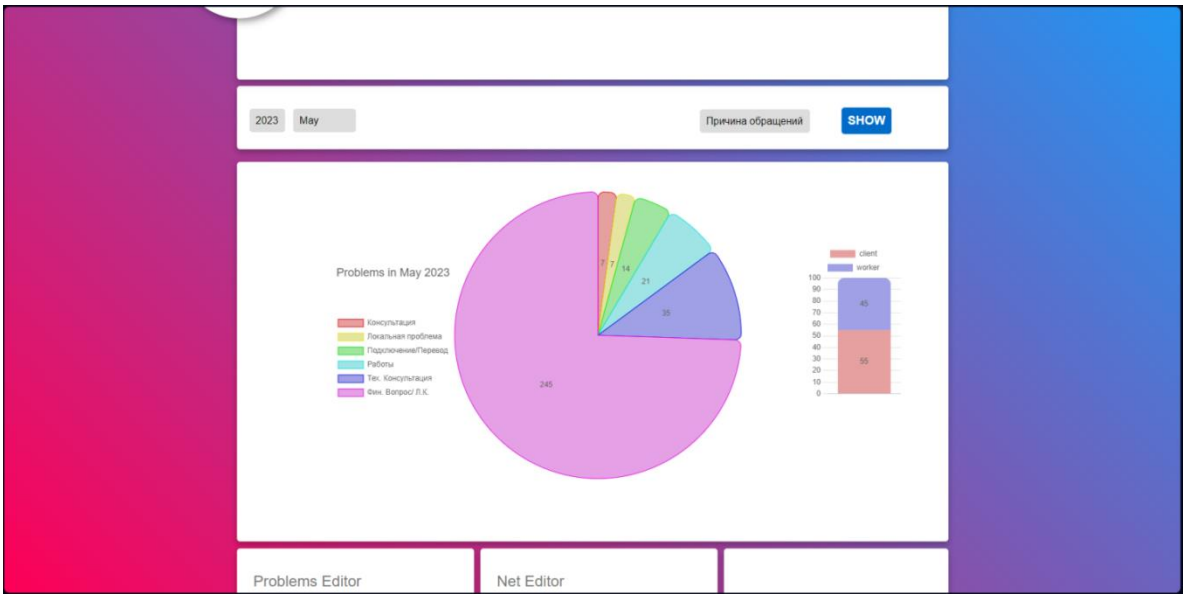


Рисунок 27 – Діаграма «Причина звернень» за окремий місяць

Завдяки бібліотеці Chart.js, що була використана для створення графіків, є можливість приховати якісь окремі елементи графіку, для більш зручного перегляду діаграми (рис. 35)

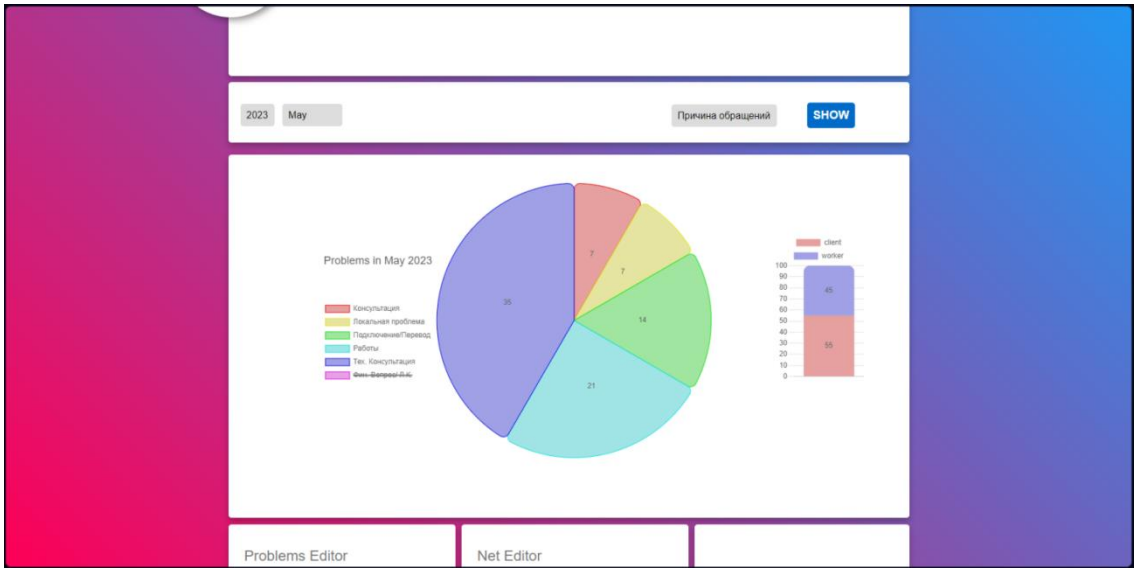


Рисунок 28 – Діаграма «Причина звернень» з приховуванням окремого елемента

Збір звернень буде проводитись весь місяць, до першого числа наступного, коли у той момент, для збереження місця, та збереження цілісності даних. Кожного першого числа місяця, о 12 ночі, система буде автоматично збирати усю інформацію до csv файлу, та очищати базу даних. Таким чином буде зберігатись уся зібрана інформація.

У момент коли один з елементів необхідно буде видалити, як писалось раніше, воно також буде очікувати наступного дня, та щоб інформація не зникла, та не пошкодилась, попередньо уся інформація збереться у файл, та видалить усю інформацію з бази даних. У кінці місяця, коли знов прийде час зберігати дані, якщо файл з назвою цього місяцю вже є, він просто додасть інформацію у файл.

# ВИСНОВКИ

В ході створення інформаційної системи було проведено докладний аналіз вимог користувачів. Це дозволило зрозуміти потреби користувачів та визначити функціональність, яку має мати система. В результаті аналізу були встановлені чіткі вимоги та цілі, які слугували основою для подальшої роботи. Під час розробки інформаційної системи було застосовано передові практики програмування та використано відповідні технології. Було написано якісний програмний код, створено базу даних та розроблено інтерфейси користувача. В результаті розробки було досягнуто якості та функціональності системи.

Загалом, створення інформаційної системи було успішно завершено, відповідно до встановлених вимог та цілей. Результатом роботи є готова система, яка відповідає початковим потребам користувачів і сприяє ефективній роботі бізнесу. Робота над проектом демонструє вміння аналізувати, проектувати та розробляти інформаційні системи з високою якістю та відповідністю вимогам замовника. У результаті створення інформаційної системи були досягнуті позитивні результати. Система забезпечує ефективний обмін даними, зберігання та обробку інформації, що сприяє оптимізації робочих процесів та підвищенню продуктивності бізнесу.

Крім того, інформаційна система відповідає сучасним технологічним стандартам та трендам. Вона побудована на основі надійних та ефективних технологій, що дозволяють швидко реагувати на змінні потреби та вимоги користувачів. Важливим аспектом розробки інформаційної системи була зручність її використання. Інтерфейс користувача розроблений з урахуванням зручності та інтуїтивного сприйняття, що дозволяє швидко орієнтуватися в системі та легко виконувати необхідні завдання.

У результаті, створена інформаційна система стала надійним інструментом для підтримки бізнес-процесів. Вона сприяє автоматизації рутинних операцій, поліпшенню комунікації та забезпечує доступ до актуальної інформації вчасно та ефективно. Завдяки цьому, бізнес може досягати більшої продуктивності, ефективності та конкурентоспроможності.

Найближчим часом планується розширити даний проєкт задля подальшого впровадження та його підтримки. Будуть додані нові більш потужні бібліотеки, планується підключити API телефонії, щоб була можливість порівнювати загальну кількість звернень та кількість оцифрованих звернень. Буде додано більш гнучкий аналіз зібраних даних. Наприклад аналіз по населеним пунктам, або по мережам.

**ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ**


1. Apache Web server. URL: https://www.britannica.com/technology/Apache-Web-server (дата звернення: 01.04.2023)
2. Microsoft SQL Server. URL:https://www.techtarget.com/searchdatamanagement/definition/SQL-Server (дата звернення: 20.05.2023)
3. Python. URL: https://www.geeksforgeeks.org/introduction-to-python/ (дата звернення: 13.05.2023)
4. What is Flask. URL: https://pythonbasics.org/what-is-flask-python/ (дата звернення: 22.04.2023)
5. Jinja. URL: https://jinja.palletsprojects.com/en/3.1.x/intro/ (дата звернення: 22.05.2023)
6. What is JavaScript. URL: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript (дата звернення: 02.05.2023)
7. HTML. URL: https://developer.mozilla.org/en-US/docs/Web/HTML?retiredLocale=uk (дата звернення: 12.04.2023)
8. CSS. URL: https://developer.mozilla.org/en-US/docs/Web/CSS?retiredLocale=uk (дата звернення: 16.03.2023)
9. Chart.js. URL: https://www.w3schools.com/ai/ai_chartjs.asp (дата звернення: 22.05.2023)
10. Chart.js. URL: https://www.chartjs.org/docs/latest/ (дата звернення: 19.05.2023)