**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ**
**Кафедра комп'ютеризованих систем управління**

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

_____Литвиненко О.Є.
«_____» _____2022 р.

# ДИПЛОМНИЙ ПРОЄКТ

**(ПОЯСНЮВАЛЬНА ЗАПИСКА)**

**ЗДОБУВАЧА ОСВІТНЬОГО СТУПЕНЯ**
**"БАКАЛАВР"**

**Тема:** Програмний модуль інформування клієнтів компанії

**Виконавець:** Бровко О.О.

**Керівник:** Росінська Г.П.

**Нормоконтролер:** Тупота Є.В.

**Київ 2022**

**НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ**

Факультет Кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп'ютеризованих систем управління
Спеціальність 123 «Комп'ютерна інженерія»
(шифр, найменування)

Освітньо–професійна програма «Системне програмування»
Форма навчання денна

ЗАТВЕРДЖУЮ
Завідувач кафедри
Литвиненко О.Є.
«_____» _____ 2022 р.

**ЗАВДАННЯ**

**на виконання дипломної роботи (проєкту)**

Бровко Олени Олександрівни
(прізвище, ім'я, по батькові випускника в родовому відмінку)

1. Тема дипломної роботи (проєкту) Програмний модуль інформування клієнтів компанії

затверджена наказом ректора від «15» лютого 2022 р. № 251/ст.
2. Термін виконання роботи (проєкту): з 16.05.2022 по 19.06.2022

3. Вихідні дані до роботи (проєкту): платформа *Openmind Traffic Control 6000* (TC6000), мови програмування *PHP*, *Javascript*, система *Linux*, мова розмітки сторінки *HTML*

4. Зміст пояснювальної записки:

1) аналіз технологій систем для надсилання СМС-повідомлень;

2) вимоги до програмного модуля інформування клієнтів компаній;

3) практична реалізація програмного модуля.

5. Перелік обов'язкового графічного (ілюстративного) матеріалу:

1) Схема відправки та доставки СМС-повідомлення;

2) Структура модуля на UML-діаграмі;

3) Інтерфейс програмного модуля;

4) Загальний алгоритм реалізації проєкту;

5) Приклад графіків створених на Kibana.

6. Календарний план–графік

| № пор | Завдання | Термін виконання | Примітка |
|---|---|---|---|
| 1 | Провести аналіз літератури за темою дипломної роботи та аналіз аналогів на ринку | 16.05.2022 – 20.05.2022 | |
| 2 | Написати розділ 1 | 21.05.2022 – 22.05.2022 | |
| 3 | Провести аналіз та розробку оптимальних алгоритмів роботи програмного модуля | 23.05.2022 – 25.05.2022 | |
| 4 | Написати розділ 2 | 25.05.2022 – 26.05.2022 | |
| 5 | Розробити програмний модуль | 26.05.2022 – 01.06.2022 | |
| 6 | Провести тестування отриманого модуля | 01.06.2022 | |
| 7 | Написати розділ 3 | 01.06.2022 – 02.06.2022 | |
| 8 | Оформити пояснювальну записку | 02.06.2022 – 03.06.2022 | |
| 9 | Підготувати графічний матеріал | 04.06.2022 | |
| 10 | Отримання допуску до захисту та подача роботи в ДЕК | 09.06.2022 | |

7. Дата видачі завдання: «16» травня 2022 р.

Керівник дипломної роботи (проєкту) _____ Росінська Г.П.
(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання _____ Бровко О.О.
(підпис випускника) (П.І.Б)

# РЕФЕРАТ

Пояснювальна записка до дипломного проєкту «Програмний модуль інформування клієнтів компанії»: 50 с., 19 рис., 1 табл., 17 літературних джерел, 1 додаток.

Ключові слова: ПРОГРАМНИЙ МОДУЛЬ, СМС-ПОВІДОМЛЕННЯ, ПЛАТФОРМА, МЕРЕЖА, ЗАПИТ.

Об’єкт дослідження – система для надсилання СМС-повідомлень.

Предмет дослідження – програмний модуль інформування клієнтів компанії.

Мета дипломного проєкту – створити веб застосунок, який дасть користувачеві можливість відправити будь-яку кількість СМС-повідомлень будь-якому отримувачу.

Методи дослідження – теоретичне ознайомлення з існуючими технологіями, які використовують при розробці систем надсилання СМС-повідомлень, програмна реалізація поширення інформацію з будь-якою кількістю отримувачей на основі платформи платформа *Openmind Traffic Control 6000 (TC6000)*.

Результати дипломної роботи рекомендується використовувати при надсиланні СМС-повідомлень з веб-застосунку.

ЗМІСТ

ВСТУП ……..8
РОЗДІЛ 1 АНАЛІЗ ТЕХНОЛОГІЙ СИСТЕМ ДЛЯ НАДСИЛАННЯ СМС-ПОВІДОМЛЕНЬ ……...……...…11
1.1. Загальний опис програмних модулів …...……...11
1.2. Аналіз аналогічних проєктів ……13
1.2.1. *Talne info* ……13
1.2.2. *Message Desk* ……..15
1.3. Загальний алгоритм розробки програмного модуля ……..17
1.3.1. Розроблення архітектури модуля ……17
1.3.2. Аналіз технологій та інструментів для реалізації модуля …….....18
1.4. Інстументи та технології ……...17
1.4.1. Мова програмування *PHP* ……...……17
1.4.2. Мова програмування *JavaScript* ……...20
1.4.3. *HTML* та *CSS* ……..21
1.4.4. *Kibana* ……...……..22
1.5. Висновки до розділу ……23
РОЗДІЛ 2 ВИМОГИ ДО ПРОГРАМНОГО МОДУЛЮ ІНФОРМУВАННЯ КЛІЄНТІВ КОМПАНІЙ ……..24
2.1. Функціональні вимоги програмного модулю ……...24
2.2. Схема працездатності функціоналу ……...26
2.3. Статистика СМС-повідомлень ……...30
2.4. Висновки до розділу ……32
РОЗДІЛ 3 ПРАКТИЧНА РЕАЛІЗАЦІЯ ПРОГРАМНОГО МОДУЛЯ ……33
3.1. Структура модуля ……33
3.2. Алгоритм програмного модуля ……..36
3.3. Тестування програмного модуля ……39
3.4. Висновки до розділу ……42
ВИСНОВКИ ……..43

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ ….45
ДОДАТОК А

## ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

*GSM* – Global System for Mobile Communications

*Kibana* – це плагін з відкритим кодом для візуалізації даних

*Elasticsearch* – це пошукова система в області Big Data

*SMPP* – Short message peer-to-peer protocol

*IMSI* – International Mobile Subscriber Identity

*SMS* – Short Message Service

*Cookies* – частина даних з веб-сайту, яка зберігається у веб-браузері

*CGI* – computer-generated imagery

*IIS* – Internet Information Server

# ВСТУП

Однією з основних ознак життя у 21 столітті є моментальний зв'язок. Більшість людей використовує онлайн месенджери, відеозв'язок, звичайні дзвінки, щоб швидко зв'язатися з опонентом та передати або отримати інформацію. Також, одним з основних видів миттєвої передачі є СМС-повідомлення. СМС-повідомлення перекладається з англійської (short message service) як "служба коротких повідомлень". Ця технологія дозволяє користувачам мобільного зв'язку ділитися між собою міні-листами (всього 160 на латиниці і не більше 70 символів кирилицею).

Багато хто вважає, що зараз СМС-повідомлення не використовуються так часто, як раніше. Але якщо подивитися на відсотки трафіку СМС-повідомлень світом, то можна побачити, що це зовсім не так. З кожним днем додаються нові оператори та хаби до мережі *GSM*. Сьогодні такі повідомлення є невід'ємною частиною сучасного світу та мобільних технологій у цілому. Цим сервісом користується понад 90% клієнтів мобільного зв'язку, а кількість надісланих повідомлень давно перевищила сотні мільярдів на рік. Відправивши короткий текст на номер іншого абонента, можна назначити зустріч, повідомити про важливу подію або привітати з днем народження.

Історія появи СМС-повідомлень бере початок з 1984 року, коли була створена система, яка дозволяє обмінюватися повідомленнями, коли телефон абонента знаходиться поза зоною дії мережі або просто вимкнено. У розробці нової системи передачі повідомлень взяли участь інженери провідних телекомунікаційних компаній, які завершили проєкт нової технології до 1989 року [1].

3 грудня 1992 р. у Великій Британії було відправлено перше sms-повідомлення. Текст цього повідомлення, відправленого з персонального комп'ютера на GSM-телефон «*Orbitel 901*» в GSM-мережі компанії *Vodafone*, звучав так: «*Merry Christmas*» (Щасливого Різдва). Надіслав sms інженер-

випровувач Ніл Папворт на телефон директора компанії *Vodafone* Річарда Джарвіса.!" [2].

А трохи пізніше, 1993-го року, було відправлено перше СМС з мобільного телефону *GSM* — і відправлено воно було через мобільний телефон фінської компанії Nokia, який "навчили" відправляти СМС.

Існує 3 процедури надсилання СМС-повідомлень:

- *P2P* повідомлення (англ. *Person-to-Person*) - "від людини до людини" повідомлення, що передаються передаються від одного користувача іншому.

- *A2P* повідомлення (англ. *Application-to-Person*) - "від програми до людини" повідомлення, які програми надсилають користувачам. Найчастіше така процедура використовується компаніями для масового розсилання користувачів. Найбільш популярними повідомленнями *A2P* є повідомлення з банків, від мобільних операторів, критичні попередження, двофакторні автентифікації на основі *SMS*, автоматичне підтвердження бронювання, програми лояльності, маркетингові повідомлення і т.д.

- *P2A* повідомлення (англ. *Person-to-Application*) - "від людини до додатку" повідомлення, що використовуються для різних кампаній з голосування на телебаченні, конкурсів, лотерейних *SMS*-кампаній, передплат і т. д. За допомогою обміну *SMS*-повідомленнями від людини до програми людина може легко взаємодіяти з брендами, компаніями та постачальниками послуг за допомогою текстових повідомлень, забезпечуючи повсюдне, швидке та безпечний маршрут для спілкування клієнт-бізнес.

Метою дипломного проєкту є створення застосунку який дасть можливість швидкого інформування клієнтів компаній. Додатковими можливостями є обрання кодування, в залежності від мови повідомлення та обрання режиму повідомлення.

Об’єкт дослідження – система надсилання СМС-повідомлень.

Предмет дослідження – програмний модуль інформування клієнтів компанії.

Для реалізації було вирішено використати сучасні методи автоматизації, а саме розробка програмного модуля з використанням мови *PHP*. Для динамічного

відображення вмісту на веб сторінці було додавання технологій *HTML CSS* та *JavaScript*. Платформою було обрано *TC 6000* від *Openmind*. Для графіків відображення трафіку було обрано плагін *Kibana*.

Розроблений проєкт має наступні функції:

– Обрання серверу для відправки повідомлення (*ip*-адреса);

– Обрання системного ідентифікатора *SMPP* сесії, спираючись на базу даних платформи *TC 6000*;

– Обрання пароля *SMPP* сесії, спираючись на базу даних платформи *TC 6000*;

– Введення номера відправника у форматі *IMSI*;

– Введення номера отримувачей у форматі *IMSI*;

– Вибір кодування повідомлення;

– Вибір режиму повідомлення;

– Введення тексту СМС-повідомлення.

Відповідно до поставленої мети необхідно вирішити наступні задачі для розроблювального програмного модуля:

– зробити аналіз існуючих програмних рішень;

– обрати інструменти для програмної реалізації;

– розробити функціонал додатка;

– розробити графічний інтерфейс користувача;

– протестувати додаток на працездатність.

# РОЗДІЛ 1
# АНАЛІЗ ТЕХНОЛОГІЙ СИСТЕМ ДЛЯ НАДСИЛАННЯ СМС-ПОВІДОМЛЕНЬ

## 1.1. Загальний опис програмних модулів

Модуль — функціонально завершений фрагмент програми, оформлений у вигляді окремого файлу з сирцевим кодом або його іменованої частини, призначений для використання в інших програмах. Модулі дозволяють розбивати складні задачі на менші відповідно до принципу модульності. Зазвичай проєктується таким чином, щоб надати програмістам зручну для багаторазового використання функціональність (інтерфейс) [3].

Програмні модулі зручно створювати для того, щоб можна було використовувати їх як окремі програми. Тобто, кожен модуль має свої компоненти, функціонал та логіку. Значною перевагою використання програмних модулей є те, що при редагуванні або модифікації певного модуля або елемента модуля, решта програми не постраждає, адже це впливає тільки на певний модуль.

Існує 2 види програмних модулів: основні та допоміжні. Модульне програмування скаладається з одного основого модуля та декількох допоміжних. З основного модуля можливо викликати функції допоміжних модулів.

Використання програмних модулів при розробці робить розуміння написання коду значно простішим, адже коротка основна функція (модуль) з прив’язкою до допоміжних функцій використовувати простіше, ніж довгу основну функцію.

Приклад використання модулів в реальному житті:

Припустимо, будинок, в якому є електрика, має кілька розеток на стінах. Ця система дозволяє підключати різні електричні прилади, наприклад, мікрохвильову піч, пральну машину, сушарку тощо.

Ці пристрої призначені для виконання свого конкретного завдання при підключенні та ввімкненні незалежно від того, де вони знаходяться. Модулі програми повинні слідувати цій же філософії. Це означає, що вони повинні виконувати своє конкретне завдання, незалежно від того, в якій частині програми вони перебувають, або навіть до якої програми вони підключені. Крім того, так само, як електричний пристрій можна легко відключити від розетки, модуль повинен бути спроєктований таким чином, щоб його можна було легко витягнути з програми. Подібно до того, як видалення електричного пристрою не впливає на функціональність інших підключених пристроїв, видалення модулів із програми не повинно впливати на функціональність інших модулів у цій програмі [4].

Також, є ще декілька переваг використання програмних модулів:

- Багаторазове використання. Модуль, який був створений для однієї програми може бути корисним та доречним до іншої. Його легко можна використовувати в іншій програмі без прив’язки до основної порграми, так як на його функціонал нічого не впливає;
- Просте налагодження та редагування. Оскільки модуль – це коротка програма, яка не має багато коду, його значно легше перевірити. Також, тестування модулів займає небагато часу. При редагуванні одного модуля інші частини програми не будуть страждати;
- Незалежне програмування об’єктів. У проєктах, над якими працюють декілька програмістів дуже зручно використовувати модульне програмування, так як робота одного програміста не залежить від роботи іншого. Кожен робить самостійно свою частину, свій модуль , і потім вони прив’язуються до основного модуля. Таким чином робота проходить значно швидше та ефективніше;

Одним із недоліків можна вважати ймовірно малі модулі, в яких використовується лише одна функція. Таким чином працювати з модулями є не раціональним рішенням, адже кожен модуль вимагає додаткового часу на розбір та обробку. Також модульне програмування може стати проблемою для групи програмістів, які працюють над одним проєктом в тому випадку, якщо стиль та

спосіб написання коду у програмістів дуже відрізняється та вкінці об’єднувати всі модулі може бути складним.

## 1.2. Аналіз аналогічних проєктів

Для порівняння існуючих аналогічних проєктів було обрано 2 сайти, на основі яких був зроблений аналіз актуальності розроблюванного проєкту.

### 1.2.1. *Talne info*

Даний сайт розроблено в Україні. В пошуку за ключовими словами він займає паршу позицію. Сайт містить у собі багато сервісів, таких як: радіо онлайн, СМС, курси валют, ціни на пальне, погода в Тальному та інші.

Було обрано сервіс «СМС» для перевірки відправлення СМС-повідомлень. Нижче приведено скріншоти інтерфейсу сайту (рис.1.1-1.2).

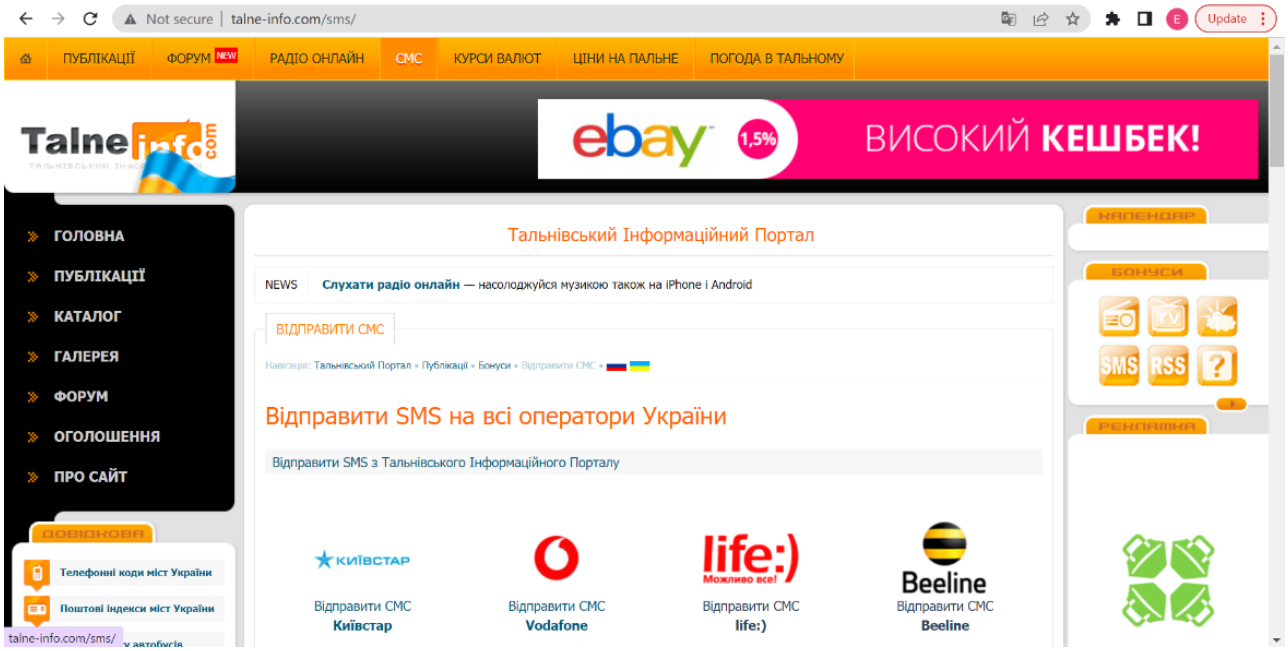


Рис. 1.1. Інтерфейс *Talne info*

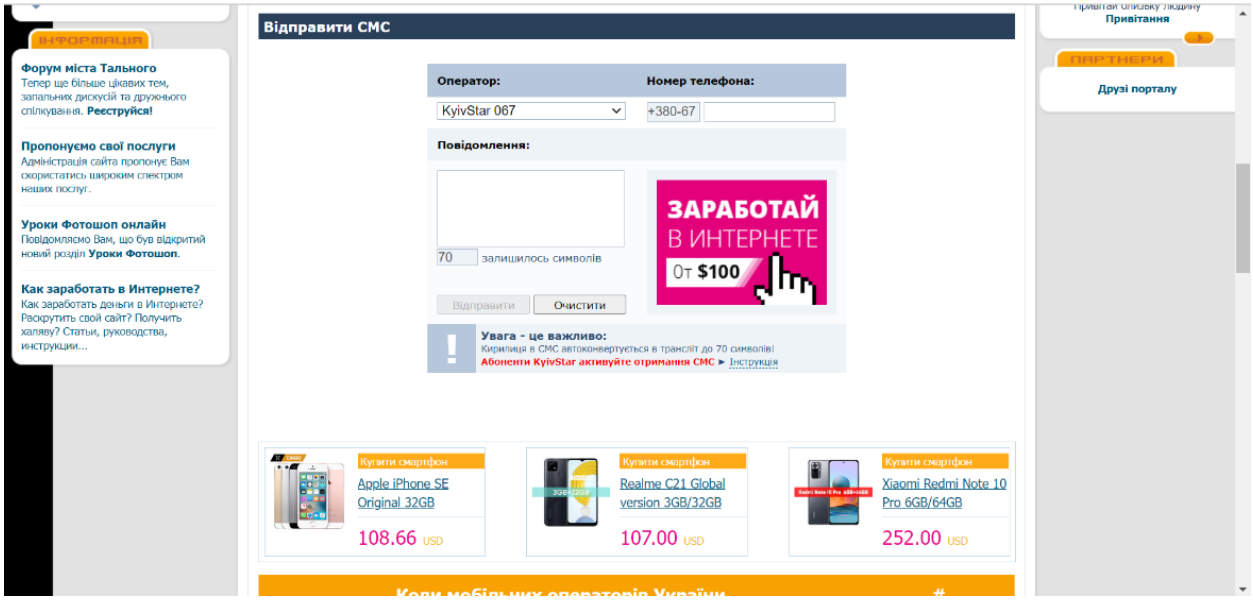


Рис. 1.2. Інтерфейс *Talne info*

Виділені наступні недоліки даного порталу:

1. Інтерфейс. Сайт містить багато реклами, яка заважає користувачеві. Також не дуже зрозуміла навігація, тож мені було складно одразу зорієнтуватися, як можна обрати те, що потрібно безпосередньо користувачеві. Судячи з контенту всього сайту, він належить одному місту, але вміщає у собі досить багато різної інформації, яка стосується не лише міста;

2. Безпека. На рис.1.3 видно, що сайт не використовує безпечний протокол *https*. Залишати свої дані на сайтах з протоколом *http* може бути небезпечним, адже сайт легко зламати та особисті дані користувачів можуть бути вкрадені;


Рис. 1.3. Протокол сайту *Talne info*

3. Працездатність. У ході проведення тестування роботи сервісу надсилання повідомлень було виявлено, що СМС не доставляється. Також, не можна надіслати СМС-повідомлення на всі номери України, так як деяких операторів немає в системі.

1.2.2. *Message Desk*

Цей сайт був створений американською компанією (рис.1.4). Основна його функція – це можливість відправлення текстових повідомлень для автоматизації та інструментів, необхідних для масштабного керування розмовами та щоденними операціями. Нижче приведений скріншот інтерфейсу головної сторінки сайту.

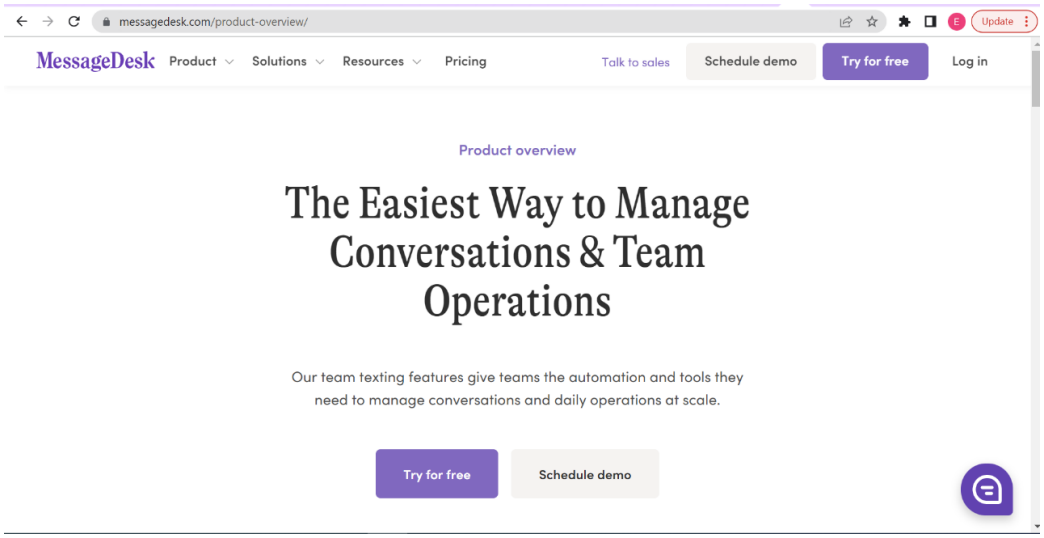


Рис. 1.4. Інтерфейс головної сторінки Message Desk

Для того, щоб протестувати працездатність додатку, необхідно зареєструватись за допомогою електроної пошти. Далі відкривається сторінка, інтерфейс якої схожий на інтерфейс поштової скриньки. Одразу надходить повідомлення від головного менеджера проєкту (рис.1.5).

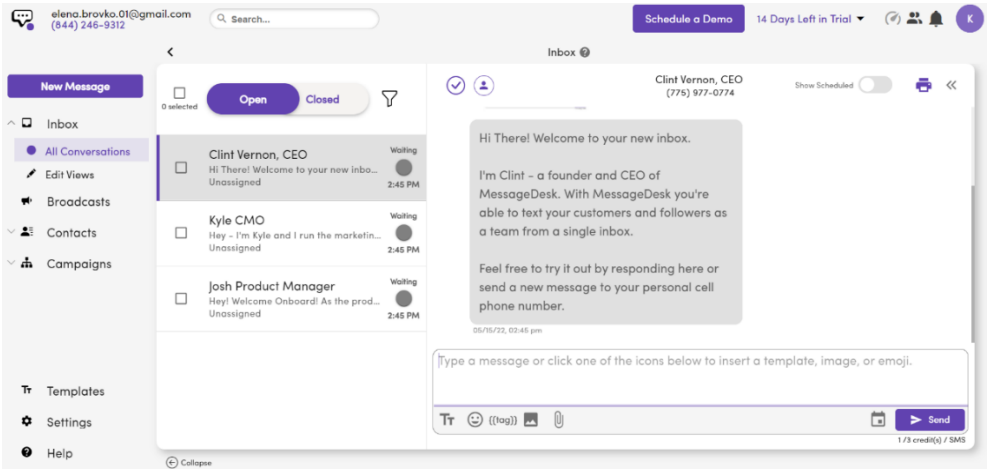


Рис. 1.5. Скріншот авторизованого користувача *Message Desk*

Дизайн сайту є лаконічним та гарним. Спроба надіслати СМС-повідомлення на український та чеський номери не вдалась. Нажаль, система не розпізнає такі

номери. Не можна вводити код країни. Видається помилка при надісланні СМС-повідомлень (рис.1.6).

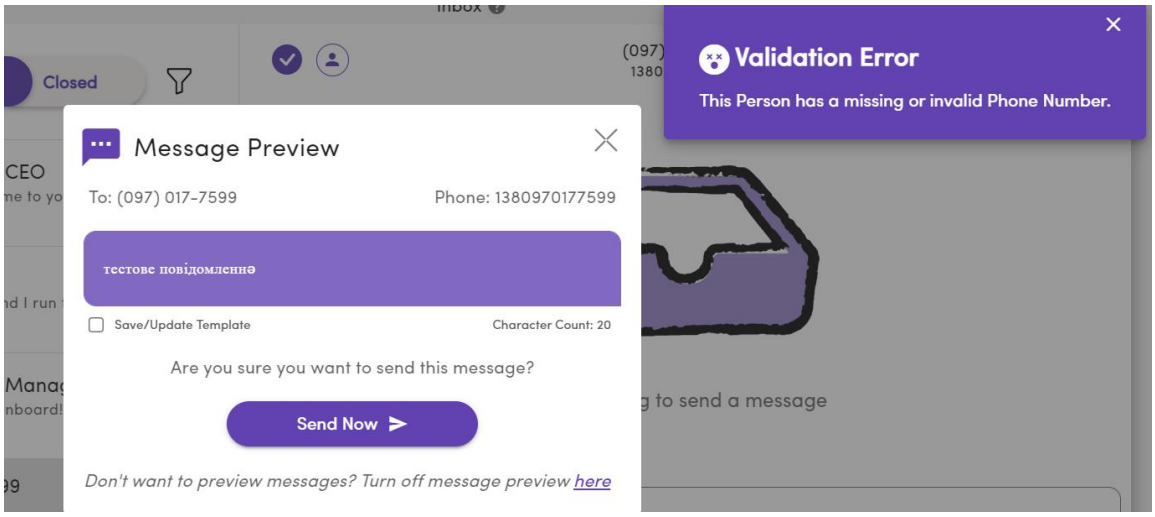


Рис. 1.6. Скріншот помилки при спробі відправки СМС-повідомлення *Message Desk*

Також, одним із значних недоліків сайту є довге завантаження сторінок, яке може займати до двух хвилин.

## 1.3. Загальний алгоритм розробки програмного модуля

Розробка програмного модуля повинна складатися з наступних етапів:

1. розроблення архітектури модуля;
2. аналіз технологій та інструментів для реалізації модуля;
3. розробка та тестування програмного рішення.

### 1.3.1. Розроблення архітектури модуля

План розроблення архітектури модуля розділено на 2 пункти:

1. Створення макету додатку. Для зручнішого використання та мінімальних затрат ресурсів проєкту треба створити лаконічний шаблон та чітку структуру.

2. Проєктування функціоналу. Для того, щоб покращити ефективність додатку, потрібно створити простий та необхідний функціонал. Додаток планується наповнити лаконічним дизайном, який не відволікає користувачів від основної цілі проєкту. Також, завдяки невеликому обсягу інформації в додатку, швидкість його завантаження має мінімальну кількість часу.

### 1.3.2. Аналіз технологій та інструментів для реалізації модуля

Для аналізу технологій та інструментів буде використовуватись 2 пункти:

1. Пошук актуальних готових рішень за темою. Для того, щоб краще розуміти актуальність проєкту потрібно проаналізувати основні засоби розробки існуючих програмних рішень: інструменти розробки та середовища.

2. Вибір оптимальних інстументів для вищої продуктивності роботи. На основі проаналізованої інформації буде обрано середовища та інструменти розробки.

### 1.3.3. Розробка та тестування програмного рішення

В першу чергу потрібно поетапно створити мінімально життєздатний продукт, який містить у собі основні функції.

Також потрібно протестувати весь життєвий цикл продукту. Тестування має відбуватись при умові різних сценаріїв для вищої продуктивності та ефективності додатку. Нижче наведено загальний алгоритм реалізації проєкту (рис.1.7).


Рис. 1.7. Загальний алгоритм реалізації проєкту

## 1.4. Інстументи та технології

Головним інструментом під час створення будь-якої програми, додатку, сайту тощо є мови програмування. Вибір мов програмування залежить від цілей, які повинні виконатись в результаті. Нижче наведено опис мов програмування та інших інструментів, які я застосувала під час написання цієї роботи.

### 1.4.1. Мова програмування *PHP*

*PHP* — це мова сценаріїв на стороні сервера з відкритим вихідним кодом, яку багато розробників використовують для веб-розробки. Це також мова загального призначення, яку можна використовувати для створення багатьох проєктів, включаючи графічні інтерфейси користувача (*GUI*).

Абревіатура *PHP* спочатку означала персональну домашню сторінку. Але тепер це рекурсивний акронім для *Hypertext Preprocessor*. (Це рекурсивне в тому сенсі, що саме перше слово є абревіатурою, тому повне значення не слідує за абревіатурою.)

Перша версія *PHP* була запущена 26 років тому. Зараз це версія 8, випущена в листопаді 2020 року, але версія 7 залишається найбільш широко використовуваною.

*PHP* працює на движку *Zend*, який є найпопулярнішою реалізацією. Є також деякі інші реалізації, як-от папуга, *HPVM (Hip Hop Virtual Machine) і Hip Hop*, створені *Facebook*.

*PHP* в основному використовується для створення веб-серверів. Він працює у браузері, а також може працювати в командному рядку.

*PHP* має деякі переваги, які зробили його таким популярним, і він є основною мовою для веб-серверів уже більше 15 років. Ось деякі з переваг *PHP*:

- Міжплатформенність: *PHP* не залежить від платформи. Не потрібно мати певну ОС, щоб використовувати її, оскільки вона працює на кожній платформі, будь то *Mac, Windows* чи *Linux*;
- *Open Source: PHP* є відкритим вихідним кодом. Оригінальний код доступний для всіх, хто хоче розробити його. Це одна з причин, чому один з її фреймворків, *Laravel*, настільки популярний;
- *PHP* синхронізується з усіма базами даних: ви можете легко підключити *PHP* до всіх баз даних, реляційних і нереляційних. Тому він може швидко підключитися до *MySQL, Postgress, MongoDB* або будь-якої іншої бази даних;
- Підтримуюча спільнота: *PHP* має дуже сприятливу онлайн-спільноту. Офіційна документація містить інструкції щодо використання функцій, і ви можете легко вирішити проблему, коли вона застрягла.

*PHP* є досить популярною мовою програмування та за рейтингом мов програмування у 2022 році він посів 5-е місце (рис.1.8).

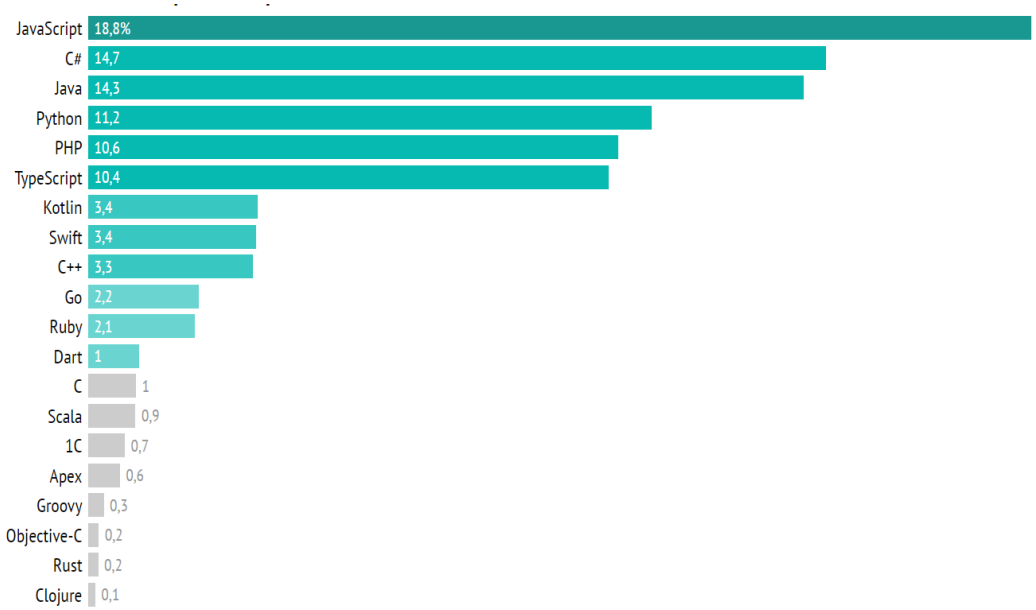


Рис. 1.8. Рейтинг мов програмування у 2022 році

### 1.4.2. Мова програмування *JavaScript*

*JavaScript* – це мова програмування, що дозволяє зробити *Web*-сторінку інтерактивною, тобто такою що реагує на дії користувача [6].

*JavaScript* є третьою за важливістю веб-технологією після *HTML* і *CSS*. *JavaScript* можна використовувати для створення веб та мобільних додатків, створення веб-серверів, створення ігор тощо.

Історія створення *JavaScript* бере початок з 1995 року, коли Брендан Айх з *Netscape* розробив і впровадив нову мову для програмістів. Спочатку він називався *Mocha*, потім *LiveScript* і, нарешті, *JavaScript*.

Зараз *JavaScript* може виконуватися не тільки в браузерах, а й на сервері або будь-якому пристрої з *JavaScript Engine*. Наприклад, *Node.js* — це фреймворк на основі *JavaScript*, який виконується на сервері.

*JavaScript* у 2022 році є найбільш популярною мовою програмування серед розробників, судячи з рейтингу (рис.1.8).

Зазвичай, *JavaScript* застосовується для наступних задач:

- дозволяє веб-сторінкам ставати більш інтерактивними багатьма способами, включаючи відображення анімації, створення функціональних

спадних меню, збільшення та зменшення зображень веб-сторінки або навіть зміну кольору кнопок, коли користувач наводить курсор миші на неї;

- розробники часто використовують фреймворки *JavaScript*, такі як *Vue*, *React* та *Angular*, під час створення веб та мобільних додатків, оскільки вони дозволяють використовувати рутинні завдання та функції, спільні для багатьох програм. Ці функції включають кнопки пошуку та параметри вибору категорій;
- широко використовується в розробці ігор, особливо для нових розробників, які хочуть практикувати та вдосконалювати свої навички;
- деякі розробники також вирішують використовувати *Node.js*, внутрішню інфраструктуру *JavaScript*, для створення базових веб-серверів і розбудови інфраструктури сайту.

### 1.4.3. *HTML* та *CSS*

Інтернет складається з веб-сторінок, для створення яких використовується мова розмітки гіпертексту *HTML*. У той самий час слід зазначити, що *HTML* є мовою програмування.

По суті HTML потрібен для розмітки текстового документа, "визначення" того, як цей текст відображатиметься у браузері. Зазначимо, що *HTML* складається з тегів, які мають “структурувати” документ.

Саме теги вказують браузеру те, де в документі заголовок, новий абзац, таблиця та інші елементи. Серед таких тегів можна виділити кілька ключових підгруп:

- основні (html, head, title, body);
- структурні (div, span);
- текстові (*p, ul, ol, li, h1-h6, br, em, strong, b, i* ).

Структура *HTML*-документа:

```
<html>
    <head>
      <title>Моя сторінка</title>
    </head>
        <body>
            Hello!
        </body>
</html>
```

*CSS* - це мова стилів, яка має визначати, як виглядатиме *HTML*-сторінка. *CSS* дозволяє “впливати” на шрифти, поля, висоту, ширину та інші аспекти оформлення веб-порталів.

### 1.4.4. *Kibana*

*Kibana* — це безкоштовна відкрита програма інтерфейсу, яка знаходиться на вершині *Elastic Stack*, надаючи можливості пошуку та візуалізації даних для даних, індексованих у *Elasticsearch*. Широко відомий як інструмент створення діаграм для Elastic Stack (раніше іменований як *ELK Stack* після *Elasticsearch, Logstash* і *Kibana*), *Kibana* також діє як інтерфейс користувача для моніторингу, керування та захисту кластера *Elastic Stack*, а також централізований хаб для вбудованих рішень, розроблених на *Elastic Stack*. Розроблена в 2013 році в рамках спільноти *Elasticsearch, Kibana* перетворилася на вікно в сам *Elastic Stack*, пропонуючи портал для користувачів і компаній.

За допомогою Kibana можна у графіках продемонструвати трафік СМС-повідомлень, відправлених за допомогою мого додатку. Приклад інтерфейсу *Kibana* наведений на рис.1.9.

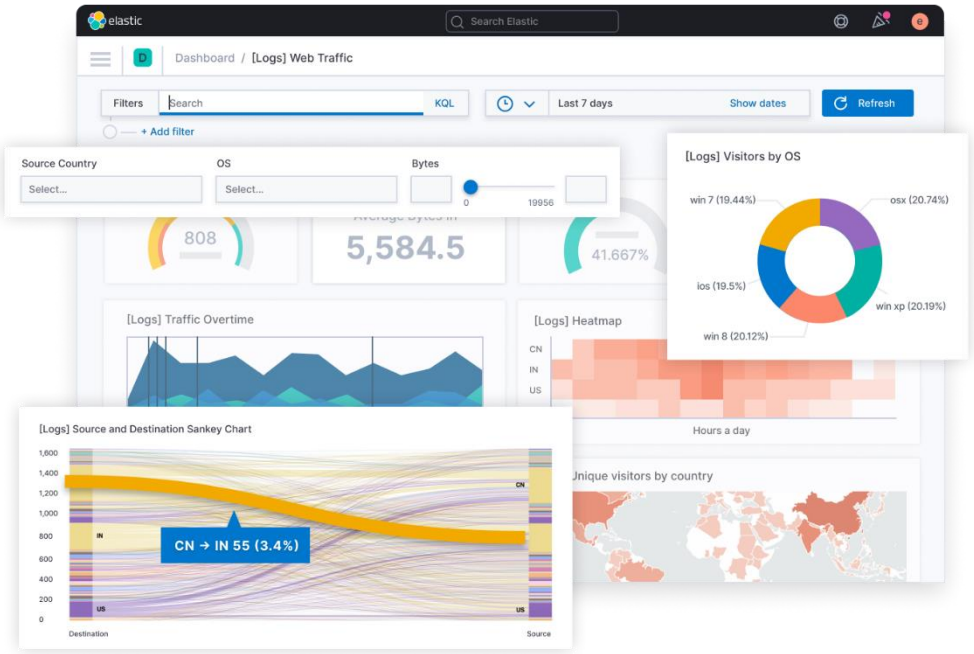


Рис. 1.9. Приклад графіків створених на *Kibana*

## 1.5. Висновки до розділу

В даному розділі було розглянуто основний алгоритм реалізації розроблюваного проєкту, а саме: розроблення архітектури модуля; аналіз технологій та інструментів для реалізації модуля; розробка та тестування програмного рішення.

Також, було проаналізовано схожі готові рішення: *Talne info* та *Message Desk* та виявлено їх основні недоліки.

Було розглянуто інструменти та технології, які будуть застосовані в даному проєкті:

- *PHP;*
- *JavaScript;*
- *HTML, CSS;*
- *Kibana.*

# РОЗДІЛ 2
# ВИМОГИ ДО ПРОГРАМНОГО МОДУЛЮ ІНФОРМУВАННЯ КЛІЄНТІВ КОМПАНІЙ

## 2.1. Функціональні вимоги програмного модулю

Для того, щоб додаток був максимально простим та зрозумілим для користувачів, було обрано мінімальний функціонал, який знадобиться для використання додатку. Нижче приведена таблиця 2.1, в якій розписано який функціонал за що відповідає.

Таблиця 2.1

Опис функціоналу додатка

| № | Знак | Опис |
|---|---|---|
| 1. |  | Сервер (*server*). Ця функція слугує для того, щоб користувач міг вписати сервер платформи, з якої він бажає надіслати повідомлення. Додаток пропонує обрати один із запропонованих серверів (*ip*-адрес). На кожній платформі зберігаються дані (оператори, їх *ip*-адреси, хаби через які комунікують оператори, маршрути та інше). |
| 2. |  | Системний ідентифікатор (*system id*). Кожен оператор smpp сесію на платформі. Під цією назвою мається на увазі логін. Це може бути будь-яка назва, яка узгоджена з оператором. Тобто в поле *system id* користувач може ввести логін *smpp* сесії з оператором. Це поле не є обов'язковим, воно створене для більшої конкретики. |

Продовження таблиці 2.1

| | | |
|---|---|---|
| 3. |  | Пароль (*password*). Це поле також не є обов'язковим. Служить воно для того, щоб користувач ввів пароль конкретної smpp сесії, до якої раніше вводив логін. |
| 4. |  | *OA (originated address)*. В це поле користувач має ввести номер, з якого буде відправлено СМС-повідомлення. Формат номеру наступний: *CC + NDC + SN*, де *CC – country code* (код певної країни), *NDC - national destination code* (національний код напрямку), *SN – subscriber number* (номер абонента). Довжина номера залежить від законів країни оператора, якому він належить. |
| 5. |  | *DA (destination address)*. Це поле слугує для того, щоб користувач ввів номер, на який він хоче відправити СМС-повідомлення. Таких номерів може бути декілька, а також можна занести номери в текстовий файл та потім завантажити списком для простішої роботи з додатком. Формат номерів збігається з форматом для поля *OA*. |
| 6. | – | *Data coding* (кодування). За замовчуванням значення кодування обране пустим, тобто в залежності від символів додаток сам розпізнає яке кодування використалось. Але також, якщо скористатись іншими варіантами списку кодувань, можна обрати будь-яке кодування з запропонованих. Тоді програма буде вважати істиною кодування, яке обрав користувач і підлаштовуватись під нього. Потім можна знайти повідомлення в логах системи та перевірити яке кодування використалось. |
| 7. | – | *Protocol ID* (протокол ідентифікації). Варіанти цього поля користувач має обрати для того, щоб вирішити чи буде повідомлення видимим, чи невидимим. Абонент може його побачити або не побачити на своєму девайсі, але в системі обидва варіанти будуть відображатись однаково. |

Закінчення таблиці 2.1

| | | |
|---|---|---|
| 8. | – | *Text* (текст). В це поле користувач може ввести текст будь-якої довжини. В залежності від кількості символів повідомлення може буде конкатенованим (при умові, що кількість символів досягає 160) або звичайним (кількість символів менша 160). Якщо повідомлення конкатеноване, то воно ділиться на 2 частини та в системі ми бачимо 2 різних повідомлення. Але користувач на своєму девайсі не помічає різниці, для нього в будь-якому випадку доходе одне ціле повідомлення. |
| 9. | – | *Send* (відправити). Ця кнопка слугує для відправки повідомлення. Коли користувач заповнив всі обов'язкові поля, він може натиснути на цю кнопку і СМС-повідомлення прийде до його адресата. |

## 2.2. Схема працездатності функціоналу

Головною ціллю розроблюваного додатку є віправлення СМС-повідомлень з ноутбука або комп'ютера. Розглянемо схему роботи додатку (рис.2.1).


Рис. 2.1. Схема роботи додатку

Ця схема має 3 основні елементи:

1. Введення та вибір певних значень функціоналу. Коли користувач відкриває веб-додаток, він має заповнити обов'язкові поля, інакше програма не дозволить відправити повідомлення;

2. Відправка СМС-повідомлення. Після заповнення основних полів користувач має натиснути на кнопку відправки, аби повідомлення дійшло до адресата;

3. Отримання СМС-повідомлення. Успішним результатом та підтвердженням працездатності додатку є відображення повідомлення на мобільному телефоні у абонента, якому воно було відправлене.

Ця схема відображає найбільш просту картину та «ідеальний» результат. Але також, можуть статись різні події, які призведуть до різних результатів.

Для того, щоб продемонструвати різні варіанти розвитку подій в залежності від вибору значень функціоналу користувачем, представлено схему відправки та доставки СМС-повідомлення при звичайних «ідеальних» умовах (рис.2.2).

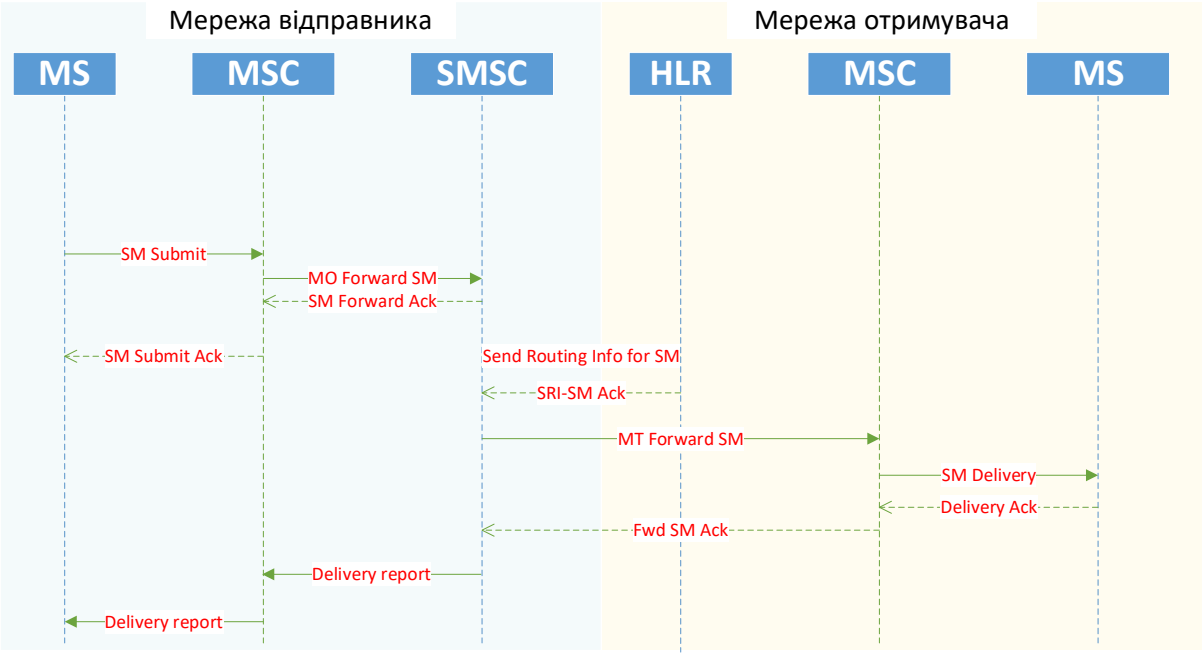


Рис. 2.2. Схема відправки та доставки СМС-повідомлення

На схемі зображено архітектуру проходження СМС-повідомлення між оператором адресантом та оператором адресатом.

Умовні позначення:

- *MS (Mobile Station)* - телефони відправника і одержувача СМС;

- *MSC (Mobile Switching Centre)* - комутатор, який обслуговує рухомі абоненти - отримувача і відправника СМС;

- *SMSC (SMS Centre)* - центр коротких повідомлень. Находиться в мережі відправника;

- *HLR (Home Location Register)* - вузол мережі *GSM*, який серед іншого зберігає інформацію про поточний комуатор (*MSC*) отримувача СМС.

Текстовий опис схеми наведено нижче.

1. Абонент *A* надсилає СМС абоненту *B*;
2. У *MS* абонента *A* вказаний номер центру коротких повідомлень (*SMSC*), через який буде здійснено доставку СМС;
3. Повідомлення надсилається на комутатор абонента А, звідти на *SMSC*;
4. *SMSC* не знає поточного комутатора абонента В, тому запитує цю інформацію у *HLR*. оскільки адреса *HLR* теж не відома, то повідомлення *Send Routing Info for SM* відправляється на адресу абонента В. Мережа одержувача відповідає за те, що повідомлення *SRI-SM* буде змаршрутизовано на відповідний *HLR* (в одній мережі їх може бути кілька);
5. *HLR* повертає як відповідь поточний *MSC* абонента, а також його мобільний номер – *IMSI*;
6. *SMSC* відправляє смс на комутатор абонента *B*, а той своєю чергою на термінал одержувача;
7. Якщо абонент *A* запитав звіт про доставку та доставка була успішною, то *SMSC* генерує звіт про доставку та надсилає його абоненту *A*.

Кожен *SMSC* має свою адресу. Якщо у компанії (оператора чи хаба) є декілька платформ, то їх адреси мають бути зазначені у партнерів задля бекапу або перенаправлення трафіку на декілька платформ (серверів).

Так у моєму додатку можна обрати сервер (адресу платформи), через яку користувач бажає надіслати повідомлення. В такому разі схема може змінюватись в залежності від адреси платформи та її налаштувань.

Якщо користувач обирає кодування, яке не підтримується тим чи іншим оператором або хабом, через який СМС-повідомлення буде проходити, є

ймовірність того, що воно не дійде до адресата та система адресанта також не отримає звіт про доставку. Статус повідомлення може бути «*failed*» (неуспішний).

Якщо користувач обирає протокол ідентифікації як «видимий», тоді схема на рис. 2.2 збігається з подіями, тобто абонент, якому відправили повідомлення, отримає його та побачить на своєму девайсі. А також, у системі можна побачити це повідомлення та статус «*successful*» (успішний), при умові що всі інші налаштування обрані та працюють належним чином. У випадку, коли користувач обирає спосіб доставки як «невидимий», схема працює так само, але абонент, якому було відправлено повідомлення не може побачити його, але у системі воно відображається.

Якщо користувач ввів текст, кількість символів якого складає більше 160, то повідомлення буде вважатись конкатенованим та буде відправлено як 2 різні повідомлення.

Максимальна кількість символів у *SMS*, включаючи цифри, інтервали, розділові знаки, залежить від виду кодування та величини повідомлення (односкладове або конкатеноване), що використовується в телефонному апараті, і становить: латиницею – 160, кирилицею – 70.

Якщо в установках телефону включено кодування *Unicode* (для телефона в цілому або тільки для *SMS*) або в процесі написання *SMS* використовуються як символи латинського, так і кириличного шрифтів, то розмір *SMS* скорочується до 70 символів.

При надсиланні *SMS*, розмір якого перевищує стандартний, телефон надсилає текст кількома повідомленнями. Таке *SMS* називається конкатенованим, тобто. складовим. У цьому випадку розмір одного *SMS* буде меншим за стандартний, оскільки частина символів задіюється для передачі службової інформації. Наприклад, розмір конкатенованого повідомлення, що складається з двох *SMS* кирилицею, буде не 140, а 134 символи.

Схема доставки повідомлення наведена на рис. 2.3.

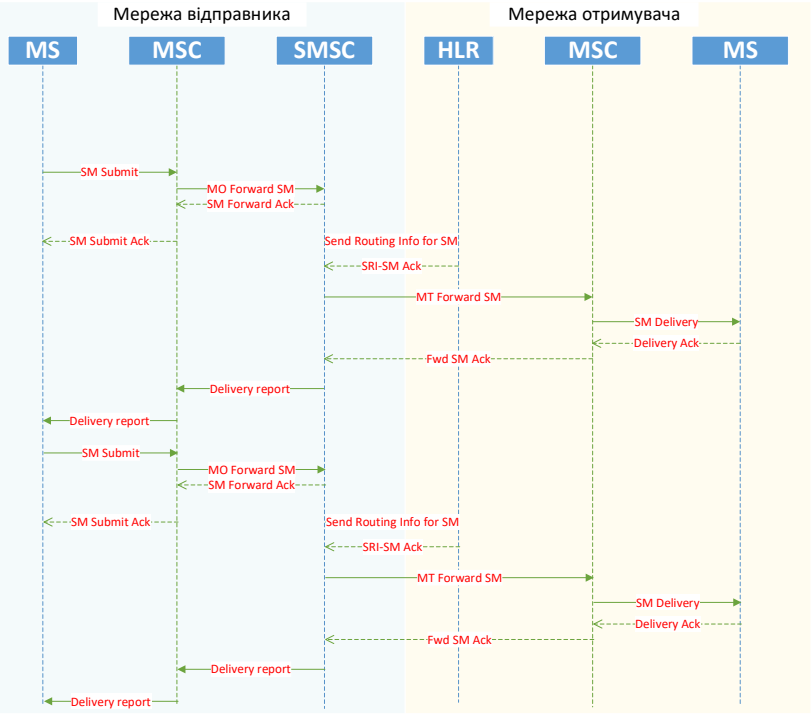


Рис. 2.3. Схема доставки конкатенованого повідомлення

## 2.3. Статистика СМС-повідомлень

Для аналізу СМС-повідомлень, відправлених з програмного модуля, було обрано плагін для візуалізації даних *Kibana*. Цей плагін має можливості створення грфіків та дашбордів з графіками. Він пропонує створення разних типів аналізу трафа (в даному випадку). На рисунку 2.4. продемонстровано дашборд з такими статистиками, як звичайні грфіки, коло з розподіленням даних, загальне число даних, розподілених за фільтрами.

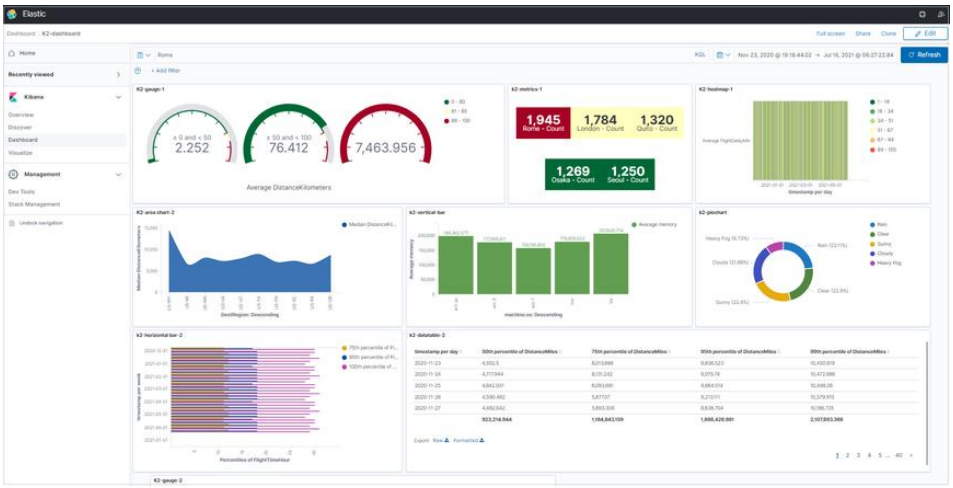


Рис. 2.4. Статистика в *Kibana*

За допомогою візуалізації можна сканувати та вивчати дані, фільтрувати результати, додавати або видаляти поля у сітці результатів, переглядати вміст записів та зберігати результати пошуку.

Для програмного модуля надсилання СМС-повідомлень зручною опцією є виведення кількості доставлених та недоставлених повідомлень, а також загальної їх кількості. Це можна робити за допомогою фільтрів, пропонуємих плагіном *Kibana* . Використовуючи одразу декілька фільтрів можна скоротити обсяг даних та отримати бажаний результат. Наприклад, зручними фільтраим для даного додатку є фільтри *src_op_name* та *dst_op_name*. *Src_op_name* – це оператор, з номера абонента якого було відправлене СМС-повідомлення, а *dst_op_name* – оператор, на номер абонента якого було відправлене повідомлення.

Плагін дає можливість візуалізації будь-яких даних, які присутні на платформі. Для використання декількох типів статистики, зручним рішенням є створення дашборду та додання усіх потрібних статистик на один дашборд. Таким чином, можна аналізувати дані за допомогою декількох графіків.

Також, можна обирати будь-які проміжки часу. Плагін пропонує 2 варіанти обрання формату часу: сьогодні, вчора, завтра тощо або з конкретною датою та часом (рис.2.5).

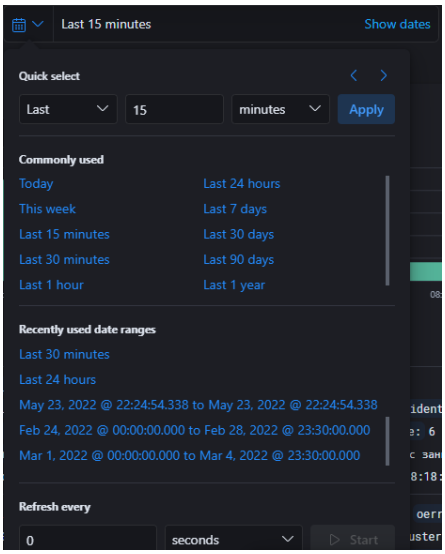


Рис. 2.5. Вибір проміжку часу в *Kibana*

## 2.4. Висновки до розділу

В даному розділі було розглянуто функціональні вимоги програмного модулю, де було представлено таблицю, в якій описано події та відповідності кожного елементу додатку при комунікації з користувачем.

Також, було розглянуто схему працездатності функціоналу. Для того, щоб більш детально показати роботу додатку, було продемонстровано схему проходження СМС-повідомлення від абонента *А* до абонента *Б*.

Крім цього, було розглянуто різні можливі події при тих чи інших налаштуваннях та це було зазначено в описі на схемі (безпосередньо проходження конкатенованих повідомлень).

Було розглянуто візуалізації за допомогою плагіну *Kibana*, основні можливості плагіну, доречні для програмного модуля інформування клієнтів компаній.

# РОЗДІЛ 3
# ПРАКТИЧНА РЕАЛІЗАЦІЯ ПРОГРАМНОГО МОДУЛЯ

## 3.1. Структура модуля

Структура модуля побудована таким чином, що спочатку користувач повинен внести дані (заповнити поля форми), потім натиснути на кнопку відправити (*send*), після чого система обробляє інформацію та відправляє повідомлення враховуючи дані, які ввів користувач. На рис.3.1 приведена *UML*-діаграма, на якій продемонстровано структуру модуля.

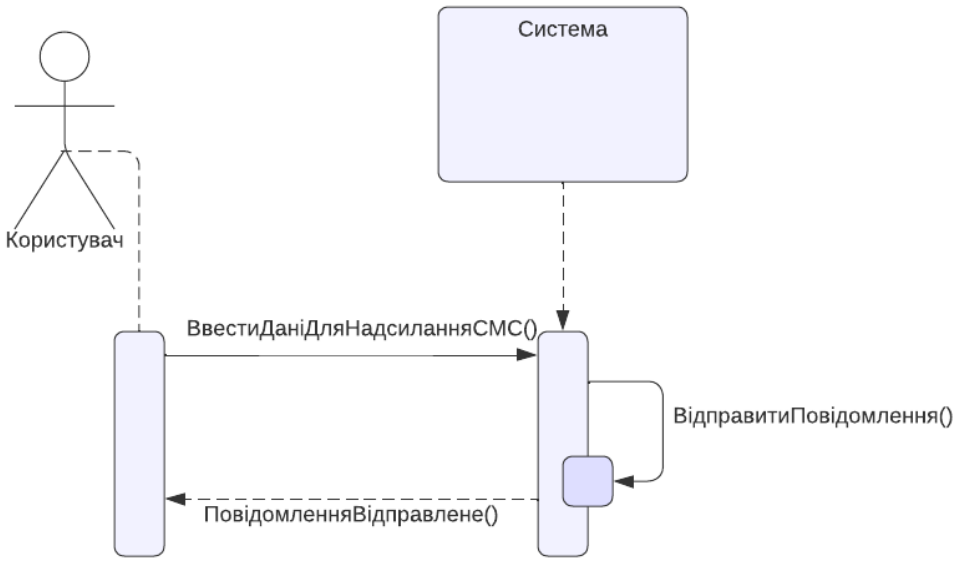


Рис. 3.1. Структура модуля на *UML*-діаграмі

В програмному модулі присутні наступні розділи в приведеній нижче послідовності:

1. Заголовок модуля

Цей розділ містить у собі безпосередньо заголовок модуля, а саме: "Програмний модуль інформування клієнтів компанії". Також він містить короткий опис та посібник користувача. Текст, що використовується як опис:

Програмний модуль інформування клентів компанії створений для того, щоб користувачі могли швидко відправляти необмежену кількість СМС-повідомлень з екрана комп'ютера. Користувач може вибрати різні налаштування для надсилання бажаного СМС-повідомлення. Щоб вибрати налаштування, потрібно використовувати форму нижче.

*1. Server.* Користувач може вибрати сервер, який пропонує йому форму, а також встановити свій сервер;

*2. Port.* Користувач може вибрати порт, який пропонує йому форма, а також встановити свій порт, який буде відповідати серверу, який він вибрав;

*3. System ID.* Користувач може вибрати системний ідентифікатор, який пропонує форму, а також задати свій системний ідентифікатор;

*4. Password.* Користувач може вибрати пароль, який пропонує йому форма, а також задати пароль, який буде відповідати системному ідентифікатору, який він вибрав;

*5. OA.* Користувач може вибрати номер відправника або альфа-ім'я, з якого надсилатиметься СМС;

*6. DA.* Користувач може вибрати один або кілька номерів одержувачів, яким буде надіслано *SMS;*

*7. Data coding.* Користувач може вибрати кодування зі списку форм, що надаються;

*8. Protocol ID.* Користувач може вибрати режим повідомлення зі списку, який надає форма;

*9. Send.* Користувач може надіслати повідомлення.

2. Обробники подій об’єкта

Створений модуль містить в собі графічний інтерфейс та інтерактивність. При натисканні на ту чи іншу кнопку користувачем, виконується певна подія. Подія — це зовнішній вплив на об’єкт, на який цей об’єкт може реагувати певним чином. Для того, щоб об’єкт реагував на події існують обробники подій. Обробник подій – блок програмного коду, який виконується в разі наступання події, з якою він пов’язаний. В даному випадку обробником подій є функції. Функція — це блок коду, який виконується лише тоді, коли вона викликається. У функцію можна передавати дані, відомі як параметри. У результаті функція може повертати дані. Інформація може передаватися у функції як аргументи. Аргументи вказуються після імені функції в дужках. Можна додавати будь-яку кількість аргументів, відокремлюючи їх комою.

Наприкад, функція *get_server* приймає значення змінної *server* та додає текст: «Ім’я серверу» (рис.3.2).

```
def get_server(server):
print(server + "Ім’я серверу")
```

Рис. 3.2. Приклад функції *get_server*

3. Ініціалізація

Ініціалізація — це процес пошуку та використання визначених значень змінних даних, які використовуються комп’ютерною програмою. Наприклад, операційна система або прикладна програма інстальовано зі значеннями за замовчуванням або заданими користувачем, які визначають певні аспекти функціонування системи чи програми. Коли операційна система або прикладна програма вперше завантажуються в пам'ять, частина програми виконує ініціалізацію, тобто переглядає файли ініціалізації, знаходить певні значення для заміни значень змінних і діє відповідно.

## 3.2. Алгоритм програмного модуля

Алгоритм програмного модуля складається з декількох частин. Спочатку було створено шаблон сторінки за допомогою мови розмітки *HTML*.

В першу чергу було створено спільний клас для всіх об’єктів модуля з назвою *«box».*

```
<div class="box"></div>
```

Потім було створено об’єкт форми, до якого було підключено файли *JavaScript* та *PHP*:

```
<form id="contact" action="mail.php" method="post"></form>
```

Форма містить наступні поля: *server, system_id, password, OA, DA, menu1* (що містить форму *form1* для кодування), *menu2* (що містить форму *form2* для системного ідентифікатора), *text*.

```
<div id="fields">
    <div class="server"></div>
    <div class="system_id"></div>
    <div class="password"></div>
    <div class="OA"></div>
    <div class="DA"></div>
    <div class="menu1">
        <form name="form1" id="data_coding"></form>
    </div>
    <div class="menu2">
        <form name="form2" id="protocol_id"></form>
    </div>
    <div class="text"></div>
</div>
```

Відповідно до об'єктної моделі документа («*Document Object Model*», коротко *DOM*), кожен *HTML*-тег є об'єктом. Вкладені теги є дітьми батьківського

елемента. Текст, що знаходиться всередині тега, також є об'єктом. Всі ці об'єкти доступні за допомогою *JavaScript*, ми можемо використовувати їх, щоб змінити сторінку.

*DOM*-дерево програмного модуля показано на рис.3.3.

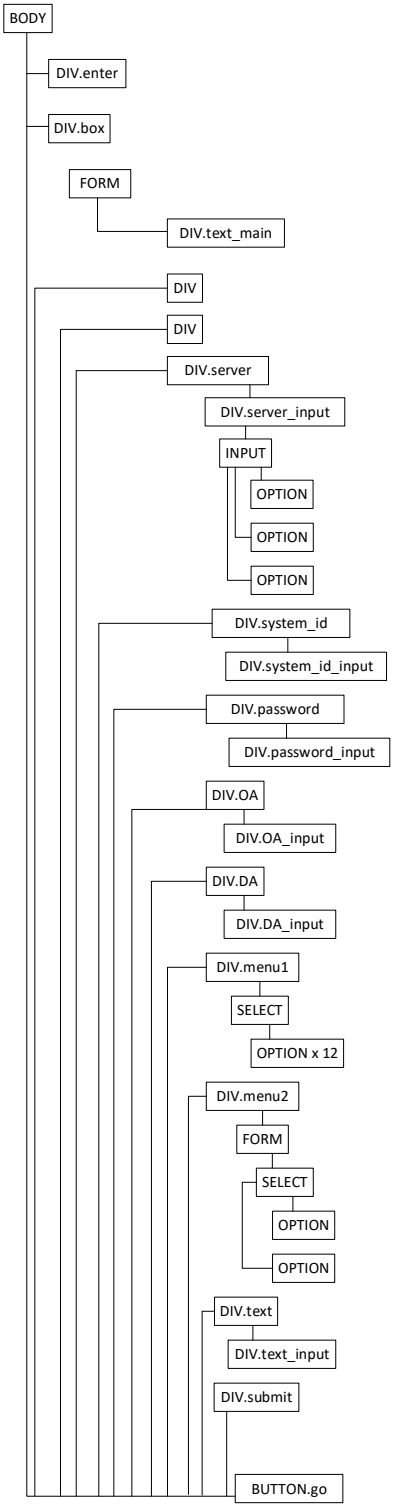


Рис.3.3. *DOM*-дерево програмного модуля

Наступним кроком було створення *PHP* скрипта, який оброблює форму та відправляє дані на сервер. Наприклад, наступний код звертається до файлу *index.html* до поля *server*.

*$server = htmlspecialchars(trim($_POST['server']));*

Для інтерактивності при натисканні на кнопку *send* було створено файл *contact.js* з підключенням *JQuery* скрипта.

Готовий документ *jQuery* використовується для ініціалізації коду JavaScript після того, як *DOM* готовий, і найчастіше використовується під час роботи з *jQuery*. Код *Javascript/jQuery* всередині $(документа). функція *ready*() завантажиться після завантаження *DOM*, але до завантаження вмісту сторінки.

```
jQuery(document).ready(function($) {
        $("#contact").submit(function() {
        var str = $(this).serialize();
        $.ajax({
                type: "POST",
                url: "mail.php",
                data: str,
                success: function(msg) {
                        if(msg == 'OK') {
                result = '<div class="ok">Повідомлення відправлено</div>';
                                $("#fields").hide();
                        }
                        else {result = msg;}
                        $('#note').html(result);
                }
        });
        return false;
});
});
```

Скрипт звертається до файлу *contact.js:*

*$("#contact").submit(function()*

Далі створюємо змінну str , в яку передаємо функцію serialize. Дана функція придатна для зберігання уявлення змінної. Це корисно для зберігання або передачі значень PHP між скриптами без втрати їхнього типу та структури.

*var str = $(this).serialize();*

Функція *$.ajax* робить перевірку надісланного повідомлення. Якщо повідомлення надіслано успішно *if(msg == 'OK'), тоді з'являється текст:* «Повідомлення відправлено». В іншому випадку *else {result = msg;}* , текст повідомлення не зникає.

## 3.3. Тестування програмного модуля

Тестування програмного модуля — це метод перевірки, чи відповідає фактичний проєкт очікуваним вимогам, і переконатися, що програмний проєкт не містить дефектів. Він включає виконання програмних/системних компонентів за допомогою ручних або автоматизованих інструментів для оцінки однієї або кількох властивостей, що цікавлять. Метою тестування програмного модуля є виявлення помилок, прогалин або відсутніх вимог на відміну від реальних вимог.

Головною вимогою є працездатність модуля, а саме успішне відправлення СМС-повідомлень.

Інтерфейс програмного модуля продемонстровано на рис.3.4.

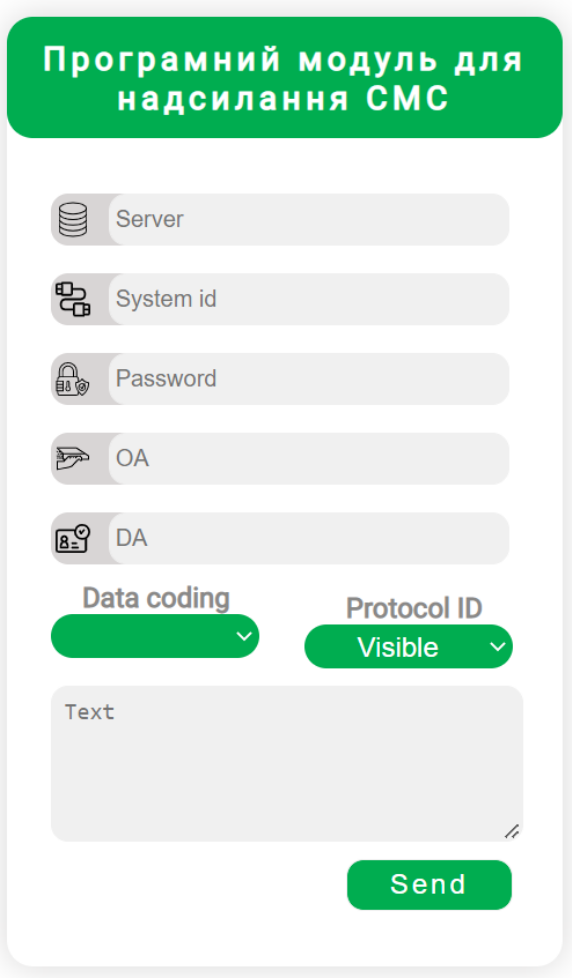


Рис. 3.4. Інтерфейс програмного модуля

В першу чергу заповнимо дані форми. В поле *Server* вводимо *ip*-адресу 10.10.1.83. В поле *System id* вводимо значення *test90*. В поле Password вводимо пароль *q2w1erty*. Для поля *OA* введемо номер іншої країни: 420773972023. В поле *DA* вводимо номмер абонента, якому буде надіслано СМС-повідомлення , в даному випадку вводимо мій особистий номер, починаючи с кода країни: 380970177599. *Data coding* обираємо за замовчуванням. В поле *text* вписуємо текст *test1*. Після цього натискаємо на кнопку *send*. На рис. 3.5. продемонстровано модуль з обраними даними для проведення першого тесту.

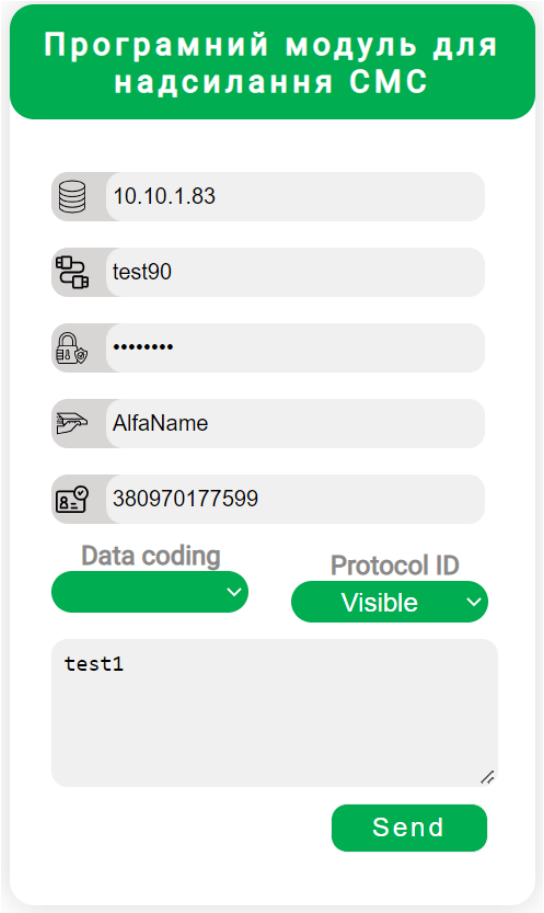


Рис. 3.5. Модуль з обраними даними для тесту 1

Результат отриманого повідомлення продемонстровано на рис. 3.6.

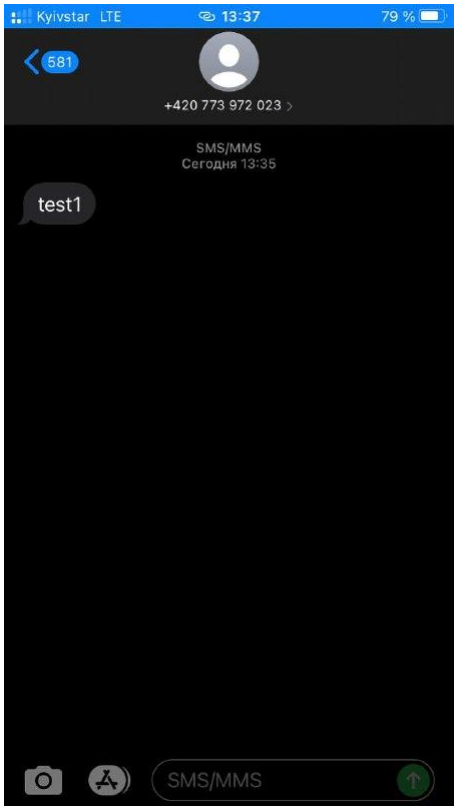


Рис. 3.6. Отримане повідомлення

Далі протестуємо сценарій, коли режим повідомлення обраний як невидимий. На телефон СМС не приходить, але можна побачити лог повідомлення. Текст повідомлення: test2. Лог приведений нижче.

*D,15d92ab8,321C3,45,SRISM,2,18E285DE,,RT_KYIVSTAR,1,1,420773972023 ,1,1,380970177599 ,ModulVidpravkySMS,,18,,0,8,0,1,1,4, ,18,0,0,6,0,1,2,4,380672059441,2,0,0,,Kyivstar_GW_25503,IW Sybase Bulk Hub,,380672020010,CeskyMobil_CZE_23003_OP,Kyivstar_GSM_JSC_UKR_25503_OP0,0,0,0,0,0, 0,0,Kyivstar_GSM_JSC_UKR_25503_OP,cache,,S#test2*

*D,15d92ab8,0,46,MTFSM,2,18E285DE,,RT_KYIVSTAR,1,1,420773972023 ,1,1,380970177599 , ModulVidpravkySMS,,18,,0,8,0,1,1,4,,18,0,0,8,0,1,2,4,380672020010,0,12C0900,0,,Kyivstar_GW_25 503,,380672020010,CeskyMobil_CZE_23003_OP,Kyivstar_GSM_JSC_UKR_25503_OP,0,0,0,0,0,0, 0,0Kyivstar_GSM_JSC_UKR_25503_OP,cache,,S#test2*

В лозі можна побачити, що спочатку відправляється запрос (*SRISM*), він виконаний із статусом *#S*, так позначається що він був успішний. Потім приходе відповідь (*MTFSM*) також із статусом *#S*. Оператори яким належать номери також визначаються та записуються у лог. Також у лозі фігурують номери , які були задіяні в відправці СМС.

Тестування було успішно пройдене. Головна мета, а саме працездатність модуля, успішне відправлення СМС-повідомлень було досягнена.

## 3.4. Висновки до розділу

В цьому розділі описано основні функції та елементи , які було створено для реалізації додатку.

Також було розглянуто *DOM*-дерево модуля. Було продемонстровано інтерфейс програмного модуля.

Тестування пройшло успішно, повідомлення були надіслані, 2 сценарія було розглянуто.

# ВИСНОВКИ

Дипломний проєкт був присвячений розробці програмного модуля для інформування клієнтів компаній. Метою виконання дипломного проекту було створення створення веб застосунку, який дасть користувачеві можливість відправити будь-яку кількість СМС-повідомлень будь-якому отримувачу.

Під час процесу створення програмного додатку були вирішені наступні задачі:

- зроблено аналіз існуючих програмних рішень;
- обрано інструменти для програмної реалізації;
- розроблено функціонал додатка;
- розроблено графічний інтерфейс користувача;
- протестовано додаток на працездатність.

В першому розділі було розглянуто основний алгоритм реалізації розроблюваного проєкту, а саме: розроблення архітектури модуля; аналіз технологій та інструментів для реалізації модуля; розробка та тестування програмного рішення. Також, було проаналізовано схожі готові рішення та виявлено їх основні недоліки та було розглянуто інструменти та технології, які будуть застосовані в даному проєкті. Завдяки базі даних платформи *Openmind Traffic Control 6000 (TC6000),* було створено звернення до операторів, до яких належать номери.

В другому розділі було розглянуто функціональні вимоги програмного модулю, де було представлено таблицю, в якій описано події та відповідності кожного елементу додатку при комунікації з користувачем. Також, було розглянуто схему працездатності функціоналу. Також було розглянуто функціонал плагіну *Kibana* та основні його можливості.

В третьому розділі було розглянуто основні функції та елементи , які було створено для реалізації додатку. Також було розглянуто *DOM*-дерево модуля та продемонстровано інтерфейс програмного модуля.

Розроблений програмний модуль має сучасний інтерфейс, зручний для користувачів.

При розробці були виконані всі задачі, поставлені під час етапу аналізу та проєктування.

Порівнюючи програмний модуль інформування клієнтів компаній з існуючими програмними рішеннями, він є актуальним та має свої переваги та недоліки.

Працездатність модуля була протестована , але в подальшому він потребує розширення функціоналу та удосконалення.

## СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Хабр [Електроний ресурс]. – 2022. – Режим доступу: *https://habr.com/ru/company/microsoftlumia/blog/139793/* (дата звернення 09.05.2022)
2. Національний музей історії України [Електроний ресурс]. – 2022. – Режим доступу: *https://nmiu.org/tsoho-dnia/2582-pershe-sms* (дата звернення 09.05.2022)
3. Вікіпедія [Електронний ресурс]. – 2022. – Режим доступу: *https://uk.wikipedia.org/wiki* (дата звернення 09.05.2022)
4. *WARBLETONCOUNCIL* [Електроний ресурс]. – 2022 – Режим доступу: *https://uk.warbletoncouncil.org/programacion-modular-11773* (дата звернення 09.05.2022)
5. *freeCodeCamp* [Електроний ресурс]. – 2022 – Режим доступу: *https://www.freecodecamp.org/news/what-is-php-the-php-programming-language-meaning-explained/* (дата звернення 10.05.2022)
6. *Web* технології та *web* дизайн [Електроний ресурс]. – 2022 – Режим доступу: *https://sites.google.com/site/webtehnologiietawebdizajn/mova-javascript-ta-ieie-mozlivosti* (дата звернення: 10.05.2022)
7. Руководство по *PHP* [Електроний ресурс]. – 2022 – Режим доступу: *https://www.php.net/manual/ru/intro-whatcando.php* (дата звернення 10.05.2022)
8. ГОСТ 2.301-68. Единая система конструкторской документации. Форматы. – Введ. 2002–01–01. – М. : Вид.-во стандартов, 2006. – 27 с. (дата звернення 11.05.2022)
9. ДСТУ 3008-95. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення / Держстандарт України. – Вид. офіц. – [Чинний від 1995-02-23]. – Київ, 2007. – 86 с. (дата звернення 11.05.2022)
10. *Elastic* [Електроний ресурс]. – 2022 – Режим доступу: *https://www.elastic.co/kibana/* (дата звернення 12.05.2022 )

11. *Visual Studio Code* [Електроний ресурс]. – 2022 – Режим доступу: https://code.visualstudio.com/ (дата звернення 13.05.2022 )
12. *Insider* [Електроний ресурс]. – 2022 – Режим доступу: *https://www.businessinsider.com/what-is-javascript* (дата звернення 15.05.2022 )
13. *WebTune* [Електроний ресурс]. – 2022 – Режим доступу: *https://webtune.com.ua/statti/web-rozrobka/yuzabiliti-sajtu/#id1* (дата звернення 16.05.2022 )
14. *SS7 for all* [Електроний ресурс]. – 2022 – Режим доступу: *http://www.mib.net.ua/2011/02/sms-forwarding.html* (дата звернення 16.05.2022 )
15. Роберт Мартін. Чиста архітектура: Мистецтво розроблення програмного забезпечення / пер. з англ. І. Бондар-Терещенко. – Харків : Вид-во “Ранок” : Фабула, 2020 – 368с.
16. Бойченко С.В., Іванченко О.В. Положення про дипломні роботи(проєкти) випускників Національного авіаційного університету.– К.:НАУ,2017. – 63 с.
17. ДСТУ 3008-95. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення / Держстандарт України. – Вид. офіц. – [Чинний від 1995-02-23]. – Київ, 2007. – 86 с.

# ДОДАТОК А

# ЛІСТИНГ КОДУ ПРОГРАМНОГО МОДУЛЯ

*INDEX.HTML:*

```
<form id="contact" action="mail.php" method="post">
	<div class="text_main">
		<h3>Програмний модуль для надсилання СМС</h3>
	</div>
	<div id="note"></div>
	<div id="fields">
		<div class="server">
			<div class="server_input">
				<img src="images/database.svg" width="20px" height="30px">
				<p><input list="server_chose" name="server" placeholder="Server" autocomplete="on" required>
					<datalist id="server_chose">
						<option>10.10.1.83</option>
						<option>10.11.1.166</option>
						<option>10.10.1.47</option>
					</datalist>
				</p>
			</div>
		</div>

		<div class="system_id">
			<div class="system_id_input">
				<img src="images/cable.svg" width="20px" height="30px">
				<p><input type="text" name="system_id" placeholder="System id" required></p>
			</div>
		</div>

		<div class="password">
			<div class="password_input">
				<img src="images/security.svg" width="20px" height="30px">
				<p><input type="password" name="password" placeholder="Password" autocomplete="on" required></p>
			</div>
		</div>

		<div class="OA">
			<div class="OA_input">
				<img src="images/sender.svg" width="20px" height="30px">
				<p><input type="tel" name="OA" placeholder="OA"></p>
			</div>
```

```
                    </div>

                    <div class="DA">
                        <div class="DA_input">
                            <img src="images/profile.svg" width="20px"
height="30px">
                            <p><input type="tel" name="DA"
placeholder="DA"></p>
                        </div>
                    </div>

                    <div class="menu1" >
                        <form name="form1" id="data_coding">
                            <p><strong>Data coding</strong></p>
                                <p><select name="coding">
                                <option value="" selected> </option>
                            <option value="0x00">SMSC Default</option>
                              <option value="0x01">IA5 (CCITT T.50)/ASCII
(ANSI X3.4)</option>
                              <option value="0x02">Octet unspecified (8-bit
binary)</option>
                              <option value="0x03">Latin 1 (ISO-8859-
1)</option>
                              <option value="0x04">Octet unspecified (8-bit
binary_2)</option>
                              <option value="0x05">JIS (X 0208-
1990)</option>
                              <option value="0x06">Cyrillic (ISO-8859-
5)</option>
                              <option value="0x07">Latin/Hebrew (ISO-8859-
8)</option>
                              <option value="0x08">UCS2 (ISO/IEC-
10646)</option>
                              <option value="0x09">Pictogram
Encoding</option>
                              <option value="0x0A">ISO-2022-JP (Music
Codes)</option>
                              <option value="0x0D">Extended Kanji JIS (X
0212-1990)</option>
                              <option value="0x0E">KS C 5601</option>
                                </select>
                                </p>
                        </form>
                     </div>

                    <div class="menu2">
                            <form name="form2" id="protocol_id">
                            <p><strong>Protocol ID</strong></p>
                            <p><select name="ID">
                                <option value="0">Invisible</option>
                                <option value="64"
selected>Visible</option>
                                </select>
                                </p>
```

```
                                        </form>
                                    </div>
                                <div class="text">
                                    <div class="text_input">
                                        <p><textarea name="message" cols="20"
rows="10" id="comment" placeholder="Text" required></textarea></p>
                                    </div>
                                </div>
                            </div>
                        </form>
```

*CONTACT.JS*

```
jQuery(document).ready(function($) {
 $("#contact").submit(function() {
        var str = $(this).serialize();
        $.ajax({
                type: "POST",
                url: "mail.php",
                data: str,
                success: function(msg) {
                        if(msg == 'OK') {
                                result = '<div class="ok">Сообщение отправлено</div>';
                                $("#fields").hide();
                        }
                        else {result = msg;}
                        $('#note').html(result);
                }
        });
        return false;
 });
});
```

*MAIL.PHP*

```
<?php
$post = (!empty($_POST)) ? true : false;
if($post) {
 $server = htmlspecialchars(trim($_POST['server']));
 $system_id = htmlspecialchars(trim($_POST['system_id']));
 $password = htmlspecialchars(trim($_POST["password"]));
 $OA = htmlspecialchars(trim($_POST["OA"]));
 $DA = htmlspecialchars(trim($_POST["DA"]));
 $data_coding = htmlspecialchars(trim($_POST["data_coding"]));
 $protocol_id = htmlspecialchars(trim($_POST["protocol_id"]));
 $message = htmlspecialchars(trim($_POST['message']));
 $error = '';
```

```
if(!$server) {$error .= 'Enter a server ';}
if(!$system_id) {$error .= 'Enter a system id';}
if(!$password) {$error .= 'Enter a password';}
if(!$OA) {$error .= 'Enter an OA';}
if(!$DA) {$error .= 'Enter a DA';}
if(!$message || strlen($message) < 1) {$error .= 'Enter the text ';}
if(!$error) {
        $address = "elena.brovko.01@gmail.com";
        $mes = "Server: ".$server."\n\nsystem_id: ".$system_id."\n\n$password: "
.$$password."\n\nOA: ".$OA."\n\n&DA: ".$OA."\n\n &text: ".$text."\n\n";
        $send = mail ($address,$mes,"Content-type:text/plain; charset = UTF-8\r\nReply-
To:$email\r\nFrom:$server <contact>");
        if($send) {echo 'OK';}
 }
 else {echo '<div class="err">'.$error.'</div>';}
}
?>
```