МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

на тему:

«Інформаційна технологія виявлення шахрайських оголошень»

Завідувач випускаючої кафедри — **Довбиш А. С.**

Керівник роботи — **Шаповалов С. П.**

Студента групи ІН.м – 02 — **Лопатка К. Р.**

СУМИ 2021

Сумський державний унівірситет
(назва вузу)

Факультет *ЕлІТ* Кафедра Комп'ютерних наук
Спеціальність *«122 -Комп'ютерні науки»*

Затверджую:
зав.кафедрою _____
_____
"_____" _____ 20___р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТОВІ

Лопатці Кирилу Романовичу
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) *«Інформаційна технологія виявлення шахрайських оголошень»*
затверджую наказом по інституту від "____" _____ 20 ___ р. № _____

2. Термін здачі студентом закінченого проекту (роботи) _____
3. Вхідні данні до проекту (роботи) _____
_____
_____
_____

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Огляд інформаційних порталів для моніторингу шахраїв; 2) Постановка завдання й формування завдань дослідження; 3) Дослідження алгоритмів для прогнозування оголошення на шахрайство; 4) Вибір стеку технологій; 5) Розробка серверної частини; 6) Проектування й реалізація користувацького інтерфейсу; 7) Аналіз результатів*
_____
_____
_____
_____

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

6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання _____

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів дипломного проекту (роботи)	Термін виконання проекту (роботи)	Примітка
1	Огляд інформаційних порталів для моніторингу шахраїв		
2	Постановка завдання й формування завдань дослідження		
3	Дослідження алгоритмів для прогнозування оголошення на шахрайство		
4	Вибір стеку технологій		
5	Розробка		
6	Оформлення пояснювальної записки до кваліфікаційної магістерської роботи		

Студент – дипломник _____
(підпис)

Керівник проекту _____
(підпис)

РЕФЕРАТ

Записка: 49 стор., 20 рис., 1 табл., 1 додаток, 19 джерел.

Об’єкт дослідження — вузько-орієнтований веб-ресурс.

Мета роботи — Розробка інформаційного ресурсу для моніторингу та зберігання даних про шахраїв.

Методи дослідження — інформації аналіз, дослідження інформаційних ресурсів, вибір стеку технологій

Результати — проведений аналіз літератури, дослідження інформаційних ресурсів, вибір стеку технологій

ІНФОРМАЦІЙНИЙ ПОРТАЛ.НЛП. ШАХРАЙСТВО.NODE.JS.NEST.JS.MONGODB.REACT.JS. DI PATTERN, MOONGOSE, FULLTEXT SEARCH. FASTI.UML-ДІАГРАМИ.JAVASCRIPT.TYPOSCRIPT.

ЗМІСТ

ВСТУП

На сьогоднішній день, людство дуже розвинулось в технологічному плані. З кожним роком, з’являються нові технології, які кардинально зміню напрямок життя. У свій час, одним з таких вагомих відкриттів в житті людства, встав інтернет. Завдяки інтернету, люди можуть спілкуватись між собою, перебуваючи на різних континентах, знаходити, зберігати чи передавати інформацію, створювати електроні бібліотеки, доступ к яким можливо здійснити з будь-якої точці та в будь-який час на Землі. Розробляти соціальні мережі, створювати онлайн банки, інтернет-магазинів, онлайн ресурси та багато іншого. Але є інша сторона інтернету – віруси, шахрайство, фішингові вебсайт та багато іншого.

Одним з найпопулярніших інтернет-ресурсів є інтернет-магазини. Їх ідея дуже проста. Інтернет-магазин – це дуже легкий вебресурс у використанні. Навіть людина, яка цвілому далека від ай-ті сфери чи яка користується рідко інтернетом зможе розібратись. Людина за допомогою комп’ютера, телефона чи планшета заходить на вебсайт, обирає товар, якій їй до вподоби, або якій їй потрібний, додає цей товар до кошика, заповню адрес доставлення й остається останній й головний етап: це покупку потрібно зробити оплату. В деяких інтернет-магазинах цим не дуже складно, в них налаштовані платіжні системи, наприклад: LiqPay, Google Pay, Apple Pay чи інші.

LiqPay – це платіжний сервіс, який надає функції Інтернет-еквайрингу – приймання платежів у мобільних додатках, на вебсайтах, підключених до Інтернету.[1]

Google Pay – це швидкий і зручний спосіб оплачувати покупки будь-де: у магазинах, на вебсайт тощо. З ним ваші дані надійно захищені, а у вас є все, що потрібно для оплати. Крім того, ви можете керувати своїм обліковим записом на вебсайт чи в додатку. [2]

Apple Pay – система мобільних платежів та електронний гаманець від корпорації Apple. Була представлена 9 вересня 2014 року. За допомогою програм Apple Pay користувачі iPhone (починаючи з iPhone 6) та Apple Watch можуть

оплачувати покупки за технологією NFC («ближній безконтактний зв'язок») у поєднанні з програмою Wallet. [3]

Якщо в інтернет-магазинах є такий функціонал, то скоріш за все, надійність того, що це надійний ресурс дуже велика. Деякі інтернет-магазини не мають таких налаштованих систем платежів, або використовують не дуже надійні системи, або ж використовують практику переводу коштів с карти покупця на картку продавця. І завдяки цьому, люди, які вирішили нажитись внаслідок інших, розробили багато алгоритмів щодо шляхом обману заробляти собі на життя. [4]

Інтернет шахрайств існує дуже багато, але як з цим можна боротись? Так ніяк, але цьому можна запобігати, наприклад створити інформаційний ресурс, на якому можна буде створювати об’яви, як їх обманював той чи інший продавець.

Мета роботи – розробка інформаційного ресурсу для моніторингу та зберігання даних про шахраїв з можливістю виявлення оголошень на шахрайство. Користувачі зможуть надсилати свої об’явлення, в яких вони будуть вказувати: інтернет посилання на об’яву, назву об’яви, телефонний номер шахрая, і текст, як додаткову інформацію. Ця заява буду фільтруватись адміністратором порталу, який буде перевіряти й після цього буде публікувати його. Буде можливість користуватись пошуковим полем, щоб перевірити, чи буле вже опубліковане об’явлення раніше чи ні.

Завданнями роботи є: розробка інформаційного ресурсу для моніторингу та зберігання даних про шахраїв з можливістю виявлення оголошень на шахрайство. Користувачі зможуть:

- надсилати оголошення;
- публікувати оголошення (зі сторони адміністратора);
- видаляти оголошення (зі сторони адміністратора)
- користуватись пошуком, за допомогою якого буде можливість фільтрувати або шукати оголошення по певному параметру;
- Перегляд вже створених оголошень.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Дослідження проблеми й статистика

Кожного року, онлайн-шопінг набирає популярності. Можна порівнювати ціни на різних інтернет ресурсах, вибрати найдешевший варіант та замовити доставлення додому. Але є один суттєвий мінус – шахраї. Вони особливо полюбили інтернет через анонімність та розробили безліч схем, за допомогою яких наживаються на людях. Мережа наповнюється фейковими онлайн-магазинами та відсутніми товарами.

З найвідоміших видів шахрайства:

- **Фішинг**. Це крадіжка персональних даних, наприклад логінів та паролів. Дані крадуть за допомогою масових розсилок нібито від імені банків та брендів або просто в месенджерах соціальних мереж. У таких листах буде посилання на сайт після переходу на який жертву обманом змусять ввести свої конфіденційні дані. Посилання на сайт, на який пропонують перейти потенційній жертві, буде схожим на посилання офіційного сайту банку або платформи онлайн-продажу, але з мінімальними змінами, наприклад, зайвою літерою на адресі сайту. Жертва переходить за посиланням, залишає свої дані - і шахраї отримують доступ до її засобів
- **Фейкові інтернет-магазини.** Шахраї створюють фейкові (не справжні) інтернет-магазини або інтернет-магазини популярних брендів, і продають «повітря». Покупець робить оплату за товар, якого не існує.
- **Покупець – аферист.** Жертвою шахраїв може стати, не тільки коли ви придбаєте товар, а й коли продаєте. Адже аферисти часто можуть виступити у ролі покупців. Вони промишляють в основному на майданчиках для продажу речей і мають на озброєнні багато схем. Практично всі готові надіслати свій товар в інше місто, заздалегідь отримавши оплату на банківську картку. Цим аферисти й користуються: вони вибирають дорогі речі, дзвонять і розпитують,

можуть поторгуватися для припинення пильності, а потім погоджуються надіслати гроші.

- **«Липова» банківська картка.** Потенційна жертва отримує нібито вигідну пропозицію: відкрити банківську картку з великим кредитним лімітом та мізерною кредитною ставкою. Все, що потрібно – лише невелика абонентська плата. [5]

За перше півріччя 2020 року в Україні зареєстровано більш ніж 47 тисяч випадків шахрайства з банківськими картками з загальною сумою збитків понад 86 млн грн.

Якщо подивитись на статистику за цей період, то є можливість побудувати діаграму, яка зобразить види шахрайства.

На рисунку зображено види шахрайства та його відсоток в Україні

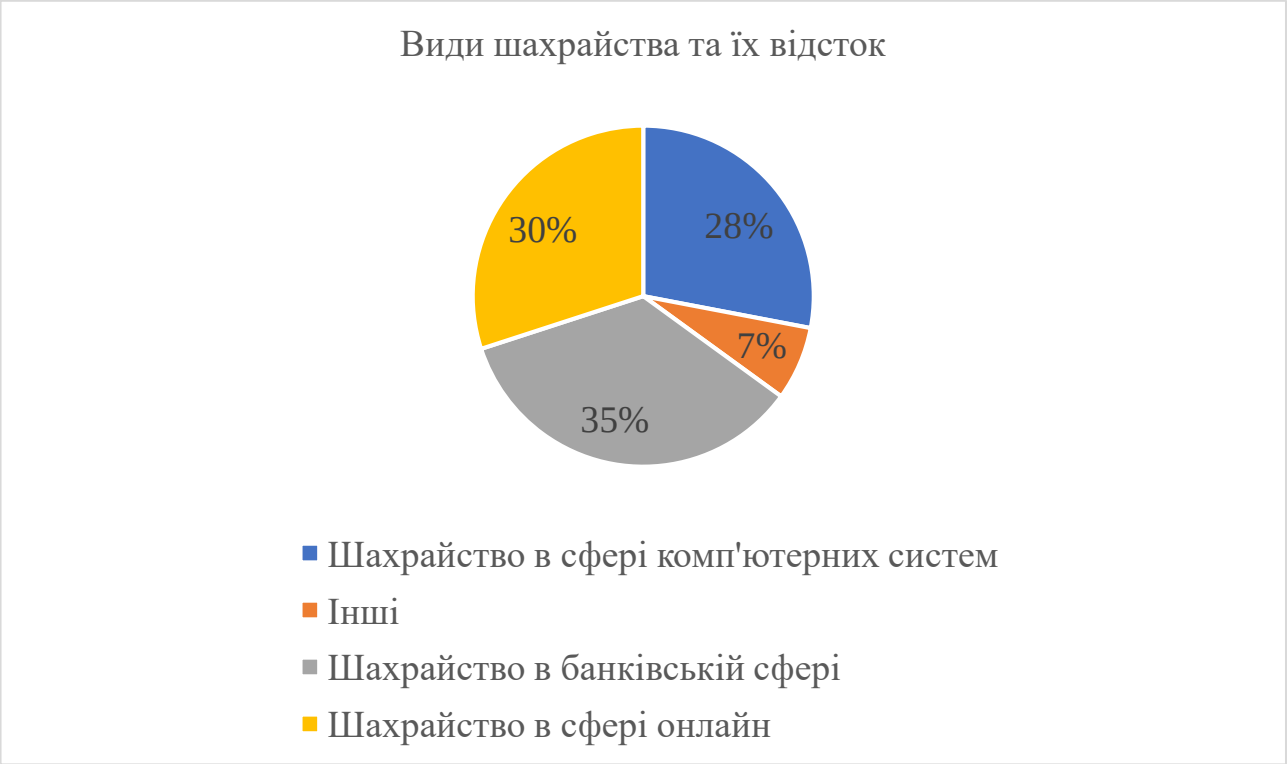

Рис. 1.1 - Види шахрайства та їх відсток в Україні.

Структура шахрайських операцій із платіжними картками в Україні, %

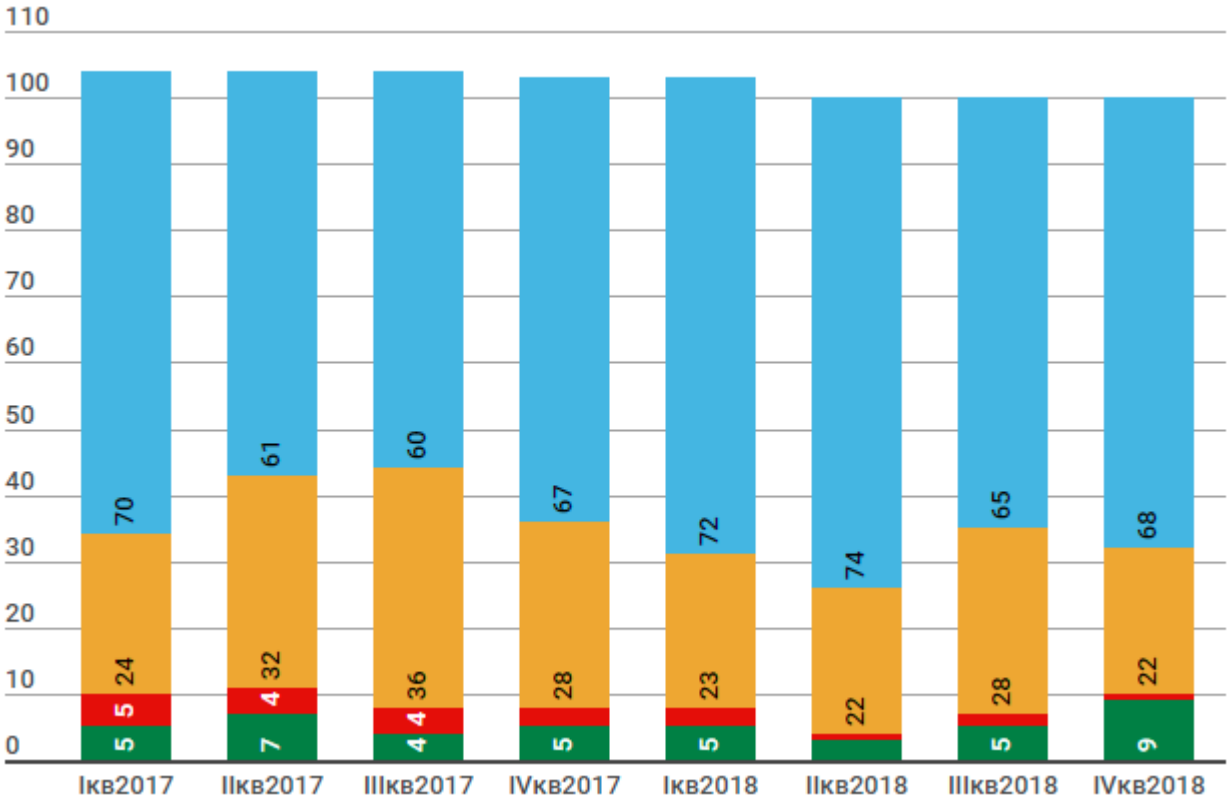

Рис. 1.2 - Структура шахрайських операцій з платіжними картками в Укрііні за певні квартали в період з 2017р по 2018р, %

Де зелений колір - Дистанційне банківське обслуговування; червоний колір - POS-термінали, жовтий – Банкомати, блакитний - Соціальна інженерія та Інтернет.

Якщо подивитись на статистику викриття шахраїв за період з 2015 по 2019рр., то можна помітити, що різниця дуже суттєва: кількість шахрайства, які відбулись (темно-блакитний колір), та шахрайства, які були виявленні та успішно викритті (блакитний колір). Тобто, з кожним роком, кількість випадків вдалих шахрайств значна більше ніж вдало викритті.

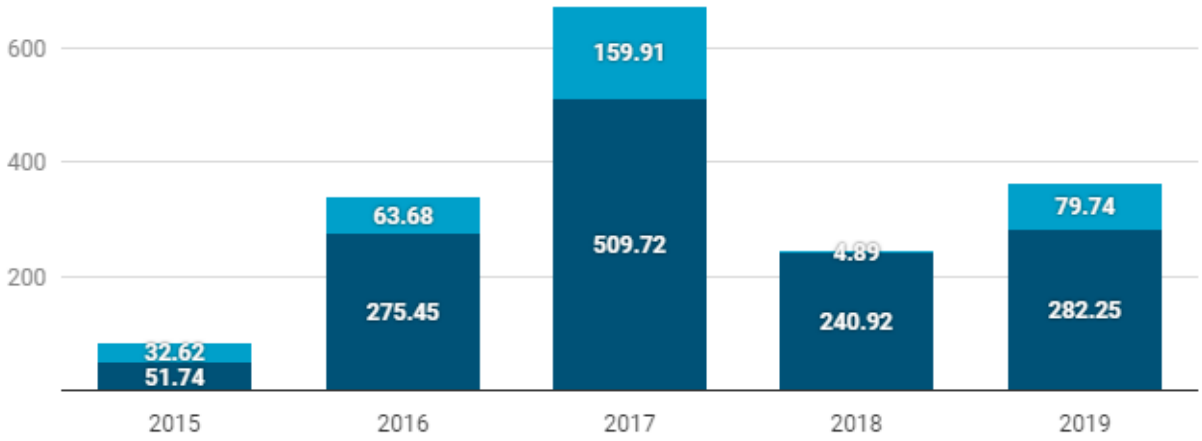

Рис. 1.3 - Різниця між кількістю успішних шахрайств (темно-блакитний колір) та викриті шахрайства (блакитний колір) за період.

На цей момент існує багато інформаційних ресурсів на тематику баз даних шахраїв. Оскільки, їх дуже багато, тому на дослідження буде обрані декілька ресурсів, вони будуть проаналізовані, аналіз буде поділений на переваги та недоліки, і після аналізу всіх інформаційних ресурсів, буде обрані кращі варіанти для реалізації в своїй кваліфікаційній магістерській роботі. [6]

1.2 Постановка задачі

Опираючись на аналіз предметної області поставимо наступне завдання:

1. Знайти й проаналізувати існуючі інформаційні портали зі схожою тематикою. Проаналізувати переваги й недоліки їх;
2. Дослідити й проаналізувати алгоритм НЛП, за допомогою якого можна буде реалізувати виявлення оголошень на шахрайство;
3. Проаналізувати сучасні технології та обрати такий стек технологій, щоб використовуючи його можна було перевершувати конкурентів;
4. Розробити інформаційний портал з функціоналом:
    a. Перегляд оголошень;
    b. Створення оголошень;
    c. Публікація оголошень (зі сторони адміністратора);
    d. Функцію пошуку;
    e. Видалення оголошень;
    f. Детальний перегляд оголошень.

5. Спроєктувати користувацький інтерфейс;
6. Провести тестування створеного інформаційного порталу.

2 ВИБІР АЛГОРИТМІВ РІШЕННЯ ПРОБЛЕМИ

2.1 Аналіз існуючих інформаційних порталів

2.1.1 Cyberpolice [7]

Це офіційний сайт кіберполіції України. Оскільки це офіційний вебресурс, то на ньому є дуже багато інформації, яка ніяк не пов’язана з основною метою роботи. Тобто, цей ресурс містить дуже багато інформації, і щоб знайти функціонал, який відповідає за пошук шахраїв дуже складно.

На порталі можна побачити розділ «Стоп фрад», який з першого важко зрозуміти для людини, яка не знає іншомовну мову, бо stop fraud в перекладі з англійської мови - покласти край шахрайству. На самій вкладці ми бачимо поле для пошуку та напис, який свідчить про «Ви можете перевірити інформацію за такими параметрами: номер банківської картки, телефон або посилання на сайт»

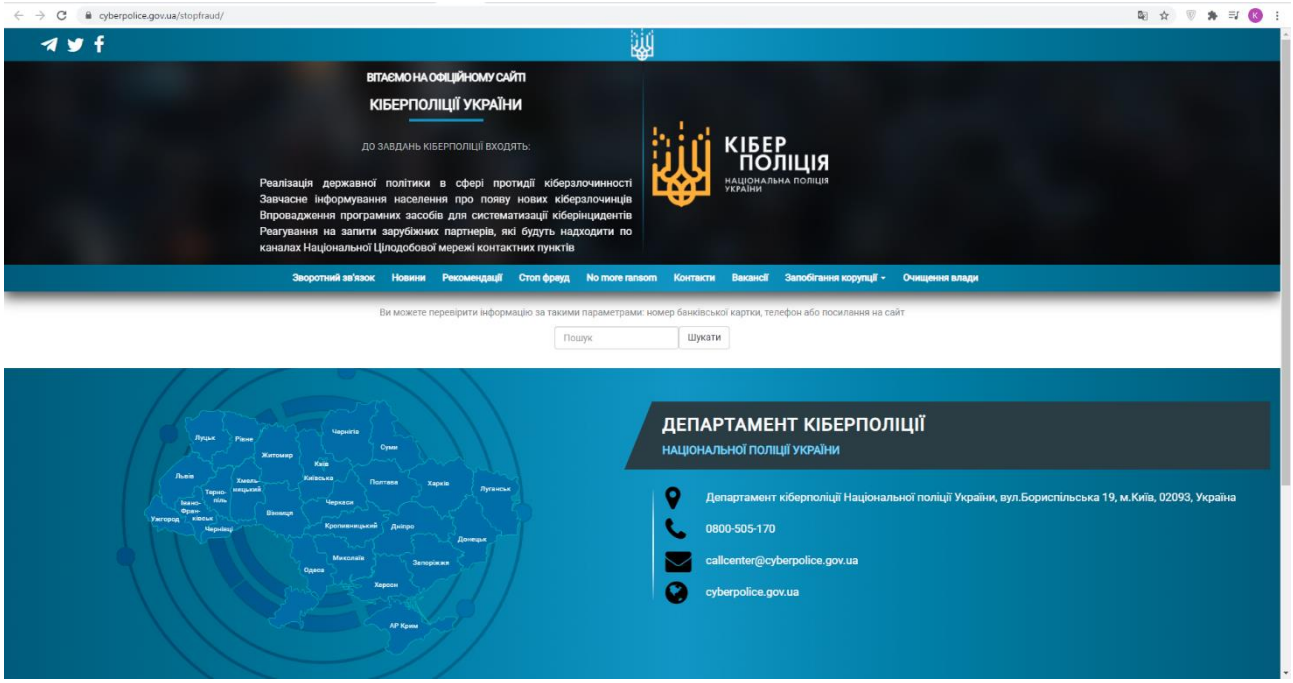

Рисунок 2.1 - Зовнішній вигляд веб-сайту cyberpolice.gov.ua

На ньому немає можливості переглянути даних, якщо у тебе не існує критерію пошуку, який потребує вебсайт.

2.1.2 Foe [8]

Переходячи на ресурс, можна одразу помітити, що мапа сайт зроблена досить не погано. Основні блоки функціоналу сайту винесені в самий вверх, і

можна одразу зрозуміти, що й до чого. Основні блоки: оголошення, додати, шахраї та пошук. Не складно зрозуміти, який блок відповідає за який функціонал.

Одним з цікавим функціоналом, який точно є перевагою над іншими порталами – це система оцінювання оголошень. Тобто, є оголошення, і люди можуть оцінювати його: чи вважають оголошення корисним чи ні.

Так же, якщо обрати будь-яке оголошення, то можна більше детальніше ознайомитись.

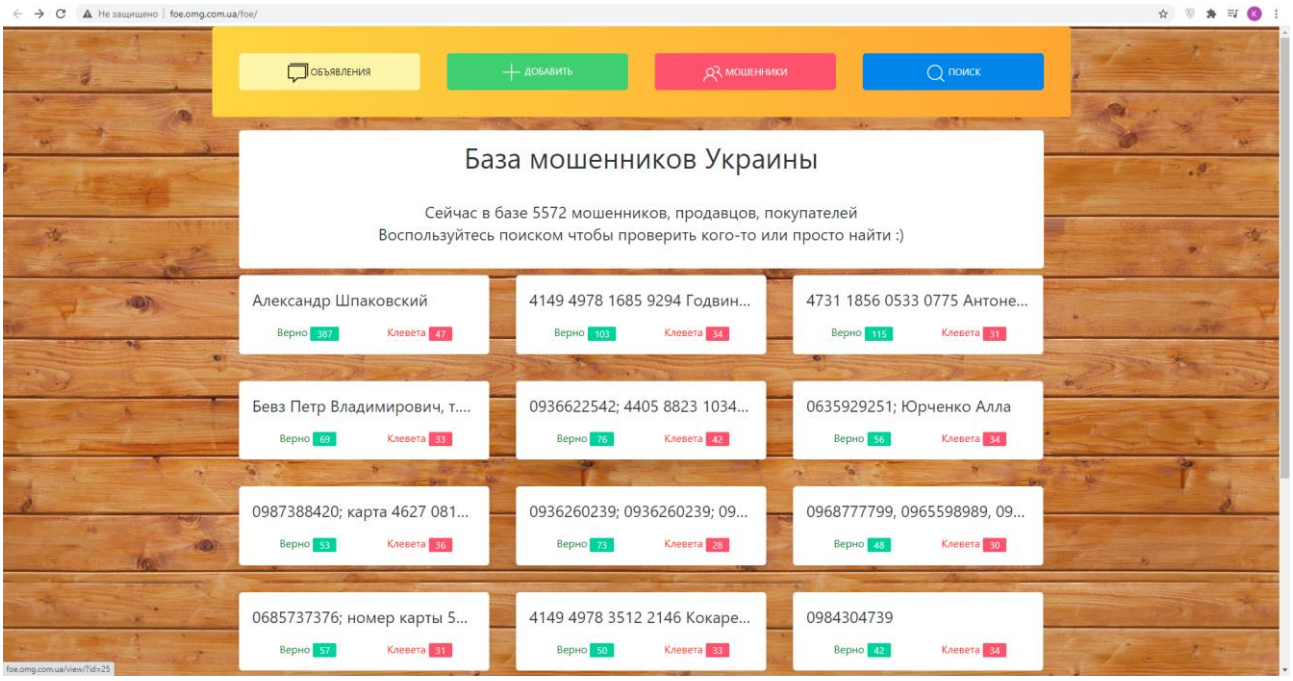

Рисунок 2.2 - Зовнішний вигляд веб-сайту foe.omg.com.ua/foe

2.1.3 Privatbank [9]

Цей ресурс теж, як і перший, офіційний. Інтернет-ресурс належить до офіційного банка України – Приват Банк. При переходжені за посиланням, користувач одразу переходить на сторінку, де може обрати регіон країни, і після того, як юзер обирає регіон – йому відображає всіх шахраїв в цьому регіоні.

Щоб переглянути інформацію, то потрібно завантажити файл. На порталі відсутній функціонал пошуку.

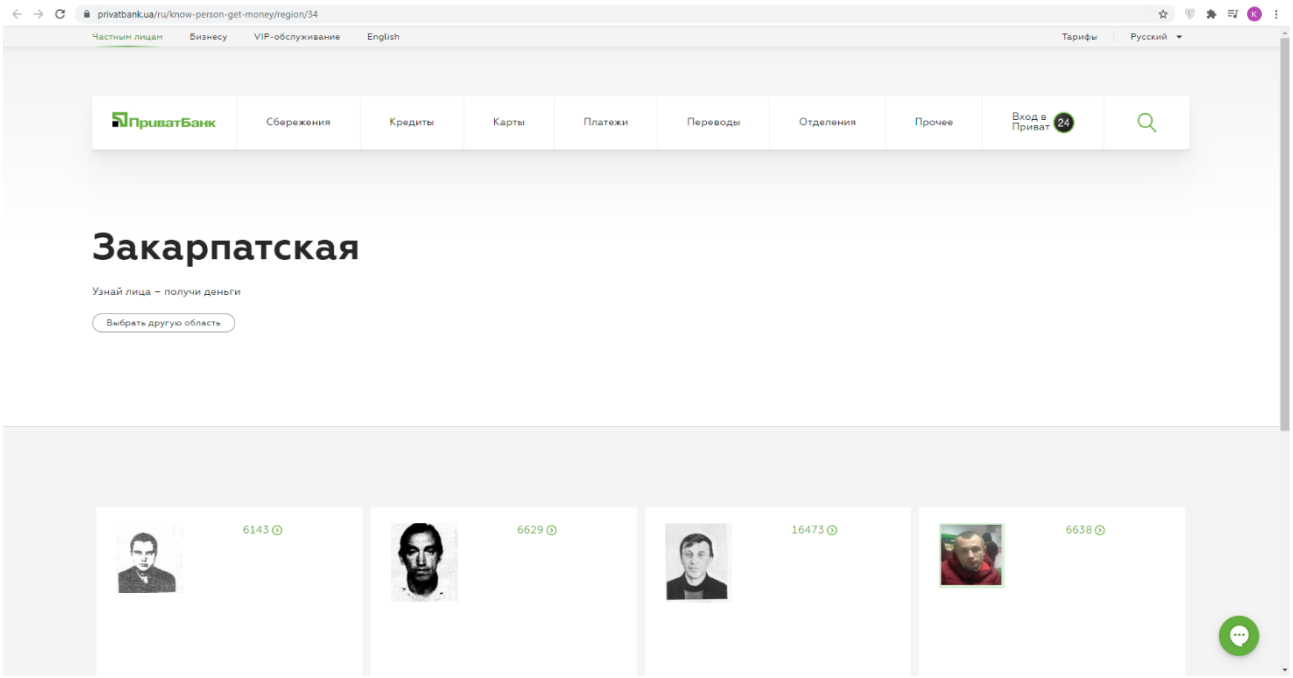

Рисунок 2.3 - Зовнішній вигляд веб-сайту privatbank.ua

2.2 Порівняльна характеристика існуючих інформаційних порталів

Після огляду інформаційних порталів, можна виділити як переваги так й недоліки. Аналізуючи результат, можна виділити такі переваги й недоліки.

Таблиця 2.1

Інформаційний портал	Переваги	Недоліки
Cyberpolice	1) Офіційний портал краї-ни; 2) Присутній функціо-нал пошуку; 3) Приємний дизайн;	1) Не має можливості перегляну-ти оголошення, якщо критерії пошуку відсутні; 2) Складно знайти потрібний функціонал; 3) Інформаційний портал спе-ціалізуються на різноманітному фунціоналі; 4) Відсутня можливість створе-ння оголошень
Foe	1) Присутня можливість створення оголошень;	1) Створення оголошень відбу-вається на сторонньому порталі;

	2) Присутній функціонал пошуку; 3) Система оцінювання оголошень; 3) Можливість перегляду оголошень без критерію пошуку; 4) Можливість більш детальніше ознайомитись з оголошенням	2) Не ергономічний дизайн;
Privatbank	1) Офіційний портал країни; 2) Пошук шахраїв по регіону; 3) Зображення шахрая;	1) Відсутній функціонал пошуку; 2) Перегляд оголошень тільки доступний при завантаженні файлу 3) Відсутня можливість публікації оголошень

2.3 Дослідження алгоритму для виявлення оголошень на шахрайство

Після дослідження та порівняння вже реалізованих вебпортал для зберігання даних про шахраїв, можна було помітити, що вони спрямованні тільки на зберігання та відображення. Не було знайдено інтернет-ресурсу, у якого була можливість виявляти оголошення на шахрайства.

Щоб виявляти оголошення на шахрайство, можна скористуватись алгоритмом НЛП.

НЛП, підмножина штучного інтелекту, надає можливість автоматично читати, розуміти та отримувати значення тексту в різноманітних контекстах.

Технологію можна навчити, щоб бути особливо обізнаними в конкретних областях, що є важливою вимогою для медичних або страхових випадків.

Обробка природної мови, або скорочено НЛП, широко визначається як автоматичне маніпулювання природною мовою, як-от мовлення та текст, за допомогою програмного забезпечення.

Синтаксичний аналіз

Синтаксичний аналіз – або синтаксичний аналіз – аналізує текст за допомогою основних граматичних правил, щоб визначити структуру речень, як слова організовані та як слова пов’язані один з одним.

Деякі з його основних підзадач включають:

- Токенізація полягає в розбиванні тексту на менші частини, які називаються токенами (які можуть бути реченнями або словами), щоб полегшити обробку тексту.
- Позначення частини мови (PoS-тегування) позначає лексеми як дієслово, прислівник, прикметник, іменник тощо. Це допомагає зробити висновок про значення слова.
- Лематизація та вихід на основі полягають у зведенні слів із відхиленнями до їх основної форми, щоб полегшити їх аналіз.
- Видалення стоп-слова видаляє часто зустрічаються слова, які не додають жодного семантичного значення, наприклад I, they, have, like, your тощо.

Семантичний аналіз

Семантичний аналіз зосереджується на охопленні значення тексту. По-перше, вивчається значення кожного окремого слова (лексична семантика). Потім розглядається поєднання слів і те, що вони означають в контексті.

Основними підзавданнями семантичного аналізу є:

- Розміщення значень слова намагається визначити, в якому значенні слово вживається в даному контексті.
- Вилучення зв’язків намагається зрозуміти, як сутності (місця, особи, організації тощо) пов’язані один з одним у тексті.[10]

2.3.1 Види аналіз мов

Алгоритм BM25

BM25 — сімейство функцій ранжирування документів, які оцінюють кількість ключових запитів у кожному документі.

Алгоритм має формулу, яка показує релевантність сторінок залежно від кількості та розташування слів (в усіх блоках тексту, крім посилання) щодо інших документів.

$$w_j(\overline{d},C) = \frac{(k_1+1)tf_j}{k_1((1-b)+b\frac{dl}{avdl})+tf_j}\log\frac{N-df_j+0.5}{df_j+0.5}$$

, де

d – документ;

c – колекція документів;

$w_j(\overline{d},C)$ - вага j-го терма в документі d колекції C;

tf_j - частота j-го терма в документі d колекції C (TF);

df_j - кількість документів колекції, містять j-й терм;

$avdl$ - середня довжина документів в колекції;

k_1, b - кофіцієнти

Суть алгоритму в тому, що якщо на сторінці немає фрази, яка відповідає пошуковому запиту, то не вдасться потрапити до ТОПу порівняно з конкурентами, які її використовують.

Алгоритм TF-IDF

TF-IDF - означає частоту термінів та інверсну частоту документів, є показником оцінки, який широко використовується в пошуку інформації (IR) або узагальненні. TF-IDF призначений для відображення того, наскільки релевантний термін у даному документі.

TF-IDF для слова в документі обчислюється шляхом множення двох різних показників:

- Частота слова в документі. Існує кілька способів обчислення цієї частоти, найпростішим є необроблена кількість випадків, коли слово з’являється в документі. Потім є способи налаштувати частоту, довжину документа або необроблену частоту найпоширенішого слова в документі.

- Обернена документна частота слова в наборі документів. Це означає, наскільки поширеним або рідкісним є слово у всьому наборі документів. Чим воно ближче до 0, тим більш поширене слово. Цю метрику можна обчислити, взявши загальну кількість документів, розділивши її на кількість документів, які містять слово, і обчисливши логарифм.
- Отже, якщо це слово є дуже поширеним і зустрічається в багатьох документах, це число наблизиться до 0. В іншому випадку воно наблизиться до 1.

Це все можна відобразити у вигляді формули:

- TF (term frequency):

$$\mathrm{tf}(t,d) = \frac{n_t}{\sum_k n_k},$$

, де

де n_t є число входжень слова t у документ, а знаменнику - загальна кількість слів у цьому документі.

- IDF (inverse document frequency):

$$\mathrm{idf}(t,D) = \log \frac{|D|}{|\{ d_i \in D \mid t \in d_i \}|}$$

, де

|D| - Число документів у колекції,

$|\{ d_i \in D \mid t \in d_i \}|$ — кількість документів з колекції D, в яких зустрічається t в колекції (коли $n_t \neq 0$) [11]

Таким чином, міра TF-IDF є твором двох співмножників:

$$\text{tf-idf}(t,d,D) = \mathrm{tf}(t,d) \times \mathrm{idf}(t,D)$$

Алгоритм Word2vec

Word2vec – технологія, розроблена Google, для знаходження сематнічних зв'язків між словами.

Word2Vec включає набір алгоритмів для розрахунку векторних уявлень слів, припускаючи, що слова, які використовуються в схожих контекстах, означають схожі речі, тобто. семантично близькі.

$$\frac{(w_v \times w_c)}{\sum(w_{c1} \times w_v)}$$, де

У чисельнику — близькість слів контексту та цільового слова.

У знаменнику — близькість усіх інших контекстів та цільового слова.

Технологія Word2Vec використовує два різні методи:

- CBOW – передбачення слова на підставі прилеглих слів.
- Skip-gram – передбачення довколишніх слів на підставі одного слова.

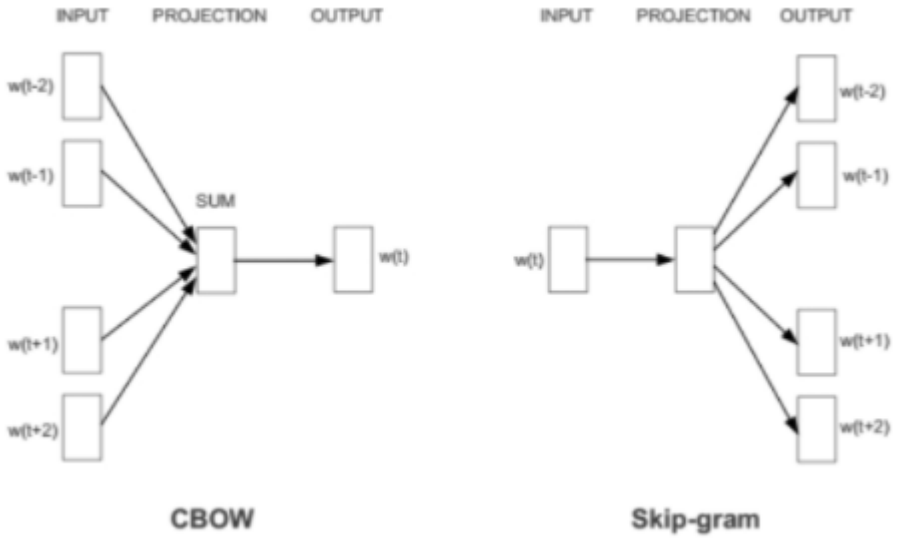

Рис 2.4 – Зображення алгоритмів та їх роботу

w(t) – це слово

w(t-2), w(t-1) і т.д. – довколишні слова [12]

2.3.2 Висновок щодо аналізу мов

Підводячи підсумок, щодо розглянутих алгоритмів для аналізу мов, найбільш доцільний вибір має бути – алгоритм TF-IDF.

TF-IDF — це техніка обробки природної мови для перетворення слів у вектори та з певною семантичною інформацією, яка дає зважені до незвичайних слів, які використовуються в різних програмах НЛП.

У той час як алгоритми машинного навчання традиційно краще працюють з числами, алгоритми TF-IDF допомагають їм розшифровувати слова, призначаючи їм числове значення або вектор. Це було революційним для машинного навчання, особливо в областях, пов’язаних з НЛП, таких як аналіз тексту.

При аналізі тексту за допомогою машинного навчання алгоритми TF-IDF допомагають сортувати дані за категоріями, а також витягувати ключові слова. Це означає, що прості монотонні завдання, як-от додавання тегів у службу підтримки або рядків зворотного зв’язку та введення даних, можна виконати за секунди.

Кожного було цікаво, як Google може надати інформацію, пов’язану з вашим пошуком, за лічені секунди? Ну, тепер ти знаєш. Векторизація тексту перетворює текст всередині документів у числа, тому алгоритми TF-IDF можуть ранжувати статті в порядку релевантності.

3 ВИБІР МЕТОДУ РІШЕННЯ

3.1. Вибір технологій

Платформа Node.js [13]

На сьогодні, існує дуже багато мов програмування. Деякі мови програмування перестають підтримуватись, деякі навпаки – швидко починають набирати свою популярність. Одним з таким є Node.js.

Node.js - це JavaScript–оточення побудоване на JavaScript–рушієві Chrome V8.

Як асинхронне подієве JavaScript–оточення, Node.js спроектований для побудови масштабованих мережевих додатків. Це контрастує з більш загальною моделлю в якій використовуються паралельні OS потоки. Такий підхід є відносно неефективним та дуже важким у використанні. Більше того, користувачі Node.js можуть не турбуватись про блокування процесів, оскільки немає жодних блокувань. Майже жодна з функцій у Node.js не працює напряму з I/O, тому процес не блокується ніколи. Оскільки нічого не блокується на Node.js легко розробляти масштабовані системи.

У нижче наведений приклад "hello world", який може одночасно обробляти багато з’єднань. Для кожного з’єднання викликається функція зворотнього виклику, проте коли з’єднань немає Node.js засинає.

```
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

Рисунок 3.1 - Зображення роботи програми, яка одночасно обробляє багато з'єднань

Node.js створений під впливом таких систем як Event Machine в Ruby або Twisted в Python. Node.js використовує подієву модель значно ширше, він приймає цикл подій (event loop) за основу оточення, замість того, щоб використовувати його в якості бібліотеки. В інших системах завжди стається блокування виклику, щоб запустити цикл подій.

HTTP є об'єктом першого роду в Node.js, розробленим з потоковістю та малою затримкою. Це робить Node.js хорошою основою для веб–бібліотеки або фреймворку.

Основними перевагами Node.js над іншими серверними мовами програмування є:

- **Величезна зовнішніх бібліотек і готових модулів.** Використання пакетного менеджера NPM дозволяє постійно розвивати екосистему Node. Сьогодні кількість опенсорсних пакетів у ньому перевищила цифру 500 тисяч і постійно зростає;
- **Можливість застосовувати одну мову на клієнті та сервері.** Якщо програміст прокачається в JavaScript, йому буде легше вивчити "надбудову", ніж технологію, що кардинально відрізняється. Загальний код. Коду, який використовується і на клієнті, і на серверній стороні, небагато, але він присутній. Головне — розуміти, що часто об'єкти з однаковими назвами можуть виконувати різні функції в браузері і на бекенді. Розроблявся спеціально для web. Вільно взаємодіє з найпопулярнішими базами даних, допомагає отримати низькорівневий доступ (http, udp, https, tcp);
- **Швидкість.** Створення робітника, що справляється з навантаженням, прототипу не забере багато часу. Перший етап, коли програміст формує кістяк майбутнього продукту, проходить дуже швидко. Якщо архітектура добре продумана, то надалі труднощів для того, щоб розширювати сайт на Node JS теж не з'явиться. Велике та бадьоре ком'юніті. Оскільки відкритий код, веб-розробники можуть писати

різні модулі та пакети та ділитися ними. Найчастіше модулі добре комбінуються. [14]

Завдяки перевагам, які були описанні вище, можна зрозуміти, що Node.js – є оптимальним вибором для реалізації порталу.

Фреймворк [15]

Nest (NestJS) — це платформа для створення ефективних, масштабованих додатків Node.js на стороні сервера. Він використовує прогресивний JavaScript, побудований з використанням і повністю підтримує TypeScript (але все ще дозволяє розробникам кодувати на чистому JavaScript) і поєднує елементи OOP (об’єктно-орієнтоване програмування), FP (функціональне програмування) і FRP (функціональне реактивне програмування).

Під капотом Nest використовує надійні фреймворки HTTP-сервера, такі як Express (за замовчуванням), і за бажанням його можна також налаштувати на використання Fastify.

Nest забезпечує рівень абстракції вище цих поширених фреймворків Node.js (Express/Fastify), але також надає їхні API безпосередньо розробнику. Це дає розробникам свободу використовувати безліч модулів сторонніх розробників, які доступні для базової платформи.

Для створення вебсерверу було обрано Fastify. В документації NestJS вказано що Fastify швидше ніж Express. Тому вибір упав на нього. Тести показують що він вдвічі швидше. Fastify — це вебфреймворк, який орієнтований на створення вебсерверів з найменшими витратами та великої кількості плагінів, які розширюють стандартний функціонал основного фреймворку.

Основні переваги Fastify:

- Високопродуктивний - фреймворк може обслуговувати до 30 тисяч запитів в секунду в залежності від складності.
- Розширюваність - можна розширювати за допомогою великої кількості плагінів і декораторів.
- Підтримка TypeScript
- Зручне використання для розробки

- Вбудоване логування
- Є вбудовані схеми для створення перевірки вхідних даних. Ця опція дозволяє контролювати дані, які пришли від клієнта [16]

База даних

MongoDB - документоорієнтована система управління базами даних, яка не вимагає опису схеми таблиць. Вважається одним із класичних прикладів NoSQL-систем, використовує JSON-подібні документи та схему бази даних. Написана мовою C++. Застосовується у веб-розробці, зокрема, у рамках JavaScript-орієнтованого стека MEAN.

Модель даних документів MongoDB, природно, підтримує JSON, а її виразна мова запитів проста для вивчення та використання розробниками. Вбудовані такі функціональні можливості, як автоматичне переключення, горизонтальне масштабування та можливість призначати дані місцеположенню.[17]

PostgreSQL — це потужна об'єктно-реляційна система баз даних з відкритим вихідним кодом, яка використовує та розширює мову SQL у поєднанні з багатьма функціями, які безпечно зберігають і масштабують найскладніші робочі навантаження даних. Витоки PostgreSQL сягають 1986 року в рамках проекту POSTGRES в Каліфорнійському університеті в Берклі і має понад 30 років активної розробки на базовій платформі.

PostgreSQL заслужив міцну репутацію завдяки своїй перевіреній архітектурі, надійності, цілісності даних, надійному набору функцій, розширюваності та відданості спільноти відкритих вихідних кодів, які стоять за програмним забезпеченням, щоб постійно надавати продуктивні та інноваційні рішення. PostgreSQL працює на всіх основних операційних системах, має ACID-сумісність з 2001 року і має потужні доповнення, такі як популярний розширювач геопросторової бази даних PostGIS. Не дивно, що PostgreSQL став реляційною базою даних з відкритим кодом, яку вибирають багато людей та організацій. [18]

Основні переваги MongoDB над PostgreSQL, це:

- MongoDB Collection використовується для зберігання пов'язаної інформації. У таблиці PostgreSQL використовуються для зберігання інформації про пов'язані дані.
- У MongoDB документ використовується для отримання інформації. У PostgreSQL використовується рядок.
- У MongoDB, якщо додається новий стовпець, він називається полем у документі. У PostgreSQL це згадується лише як стовпець.
- У MongoDB набір реплік використовується для підтримки набору даних. У PostgreSQL реплікація є синхронною, що називається 2-безопасной реплікацією.
- MongoDB конвеєр агрегації використовується в запитах. PostgreSQL GROUP_BY використовується для тієї ж мети.
- MongoDB має форуми підтримки спільноти та інші онлайн-сайти, такі як StackOverflow та усуває помилки. PostgreSQL має широкий спектр форумів спільноти та комерційної підтримки.
- MongoDB підтримує документацію, яка допомагає визначити сервери. PostgreSQL підтримує онлайн-документацію.
- MongoDB слідує за розподіленою архітектурою. PostgreSQL слідує монолітній архітектурі.

MongoDB і PostgreSQL – це різні системи управління базами даних. Їх архітектура в основному відрізняється і вони відрізняються у використанні, оскільки MongoDB документується на основі, яка використовує колекції для зберігання пов'язаної інформації. PostgreSQL використовується головним чином, коли використовується статичний JSON та дані структуровані для зберігання SQL. MongoDB в основному використовується, коли дані неструктуровані, і існує необхідність зміни даних JSON всередині сховища.

React.[19]

React — це декларативна, ефективна та гнучка JavaScript-бібліотека для створення інтерфейсів користувача. Вона дозволяє вам збирати складний UI з маленьких ізольованих шматочків коду, які називаються «компонентами».

React має кілька різних видів компонентів, але ми почнемо з підкласів React.Component:

```
class ShoppingList extends React.Component {
  render() {
    return (
      <div className="shopping-list">
        <h1>Список покупок для {this.props.name}</h1>
        <ul>
          <li>Instagram</li>
          <li>WhatsApp</li>
          <li>Oculus</li>
        </ul>
      </div>
    );
  }
}

// Пример использования: <ShoppingList name="Марк" />
```

Рисунок 3.2 - Вигляд компоненти

Переваги ReactJS:

- Легко вивчати та використовувати

ReactJS набагато простіше у вивченні та використанні. Він поставляється з гарним запасом документації, підручників та навчальних ресурсів. Будь-який розробник, який має досвід роботи з JavaScript, може легко зрозуміти і почати створювати веб-програми з використанням React протягом декількох днів. Це V (view part) у моделі MVC (Model-View-Controller), і його називають "одним із JavaScript фреймворків". Він не є повнофункціональним, але має перевагу у вигляді відкритої бібліотеки JavaScript User Interface (UI), яка допомагає виконати завдання.

- Створення динамічних веб-застосунків стає простіше

Створити динамічний веб-додаток за допомогою HTML-рядків було непросто, оскільки це вимагало складного кодування, але React JS вирішує цю проблему і робить її простішою. У ньому менше кодування та більше функціональності. Він використовує JSX (JavaScript Extension), який є спеціальним синтаксисом, що дозволяє використовувати HTML-лапки і синтаксис HTML-тегів для візуалізації певних підкомпонентів. Він також підтримує створення машиночитаних кодів.

- Багаторазово використовувані компоненти

  Веб-додаток ReactJS складається з безлічі компонентів, і кожен компонент має свою власну логіку та елементи керування. Ці компоненти відповідають за виведення невеликого, багаторазово використовуваного фрагмента HTML-коду, який можна повторно використовувати скрізь, де це необхідно. Багаторазовий код допомагає зробити ваші програми більш простими у розробці та обслуговуванні. Ці компоненти можуть бути вкладені в інші компоненти, що дозволяє створювати складні програми з найпростіших будівельних блоків. ReactJS використовує механізм віртуального DOM для заповнення даних у HTML DOM. Віртуальний DOM працює швидко, оскільки він змінює лише окремі елементи DOM замість того, щоб щоразу перезавантажувати весь DOM.

- Підвищення продуктивності

  ReactJS підвищує продуктивність завдяки віртуальному DOM. DOM - це кросплатформовий та програмний API, який працює з HTML, XML або XHTML. Більшість розробників стикалися з проблемою, коли DOM оновлювався, що уповільнювало продуктивність програми. ReactJS вирішив цю проблему, запровадивши віртуальну DOM. React Virtual DOM існує повністю у пам'яті і є уявленням DOM веб-браузера. Завдяки цьому, коли ми пишемо компонент React, ми не пишемо безпосередньо у DOM. Натомість ми пишемо віртуальні компоненти, які реакт буде перетворювати на DOM, що призводить до більш плавної та швидкої роботи.

- Підтримка зручних інструментів

  React JS також набув популярності завдяки зручному набору інструментів. Ці інструменти роблять завдання розробників зрозуміліше та простіше. React Developer Tools був розроблений як розширення для Chrome та Firefox dev і дозволяє переглядати ієрархії компонентів React у віртуальному DOM. Він також дозволяє вибирати

певні компоненти, досліджувати та редагувати їх поточні реквізити та стан.

- Відомий як SEO-дружній

  Традиційні JavaScript-фреймворки мають проблеми із SEO. Пошукові системи зазвичай мають труднощі при читанні додатків, перевантажених JavaScript. Багато веб-розробників часто скаржилися на цю проблему. ReactJS вирішує цю проблему, допомагаючи розробникам легко орієнтуватися у різноманітних пошукових системах. Це відбувається тому, що програми React.js можуть виконуватися на сервері, а віртуальний DOM рендеруватиметься і повертатиметься в браузер як звичайна веб-сторінка.

- Перевага наявності бібліотеки JavaScript

  Сьогодні ReactJS вибирають більшість веб-розробників. Це пов'язано з тим, що він пропонує дуже багату бібліотеку JavaScript. Бібліотека JavaScript забезпечує веб-розробникам більшу гнучкість, дозволяючи їм вибирати те, що вони хочуть.

- Можливість тестування коду

  Програми ReactJS легко тестуються. Розробники можуть тестувати та налагоджувати свої коди за допомогою вбудованих інструментів.

Завдяки перевагам, які були описанні вище, можна зрозуміти, що ReactJS – є оптимальним вибором для реалізації порталу.

3.2. Розробка серверної частини

Для того, що зрозуміти як найкраще реалізувати серверну частину, було розроблено та побудовано UML-діаграми, які спрямовані на розуміння головної мети.

UML – уніфікована мова моделювання (Unified Modeling Language) – це система позначень, яку можна використовувати для об'єктно-орієнтованого аналізу та проектування. Його можна використовувати для візуалізації, специфікації, конструювання та документування програмних систем.

Мінуси:

- необхідність знання різних діаграм та їх нотацій;
- розробка займає достатньо часу.

Плюси:

- можливість подивитися на завдання з різних точок зору;
- іншим програмістам легше зрозуміти суть завдання та спосіб її реалізації;
- діаграми порівняно прості читання після досить швидкого ознайомлення з їх синтаксисом.

UML-діаграма для ролі адміністратора:

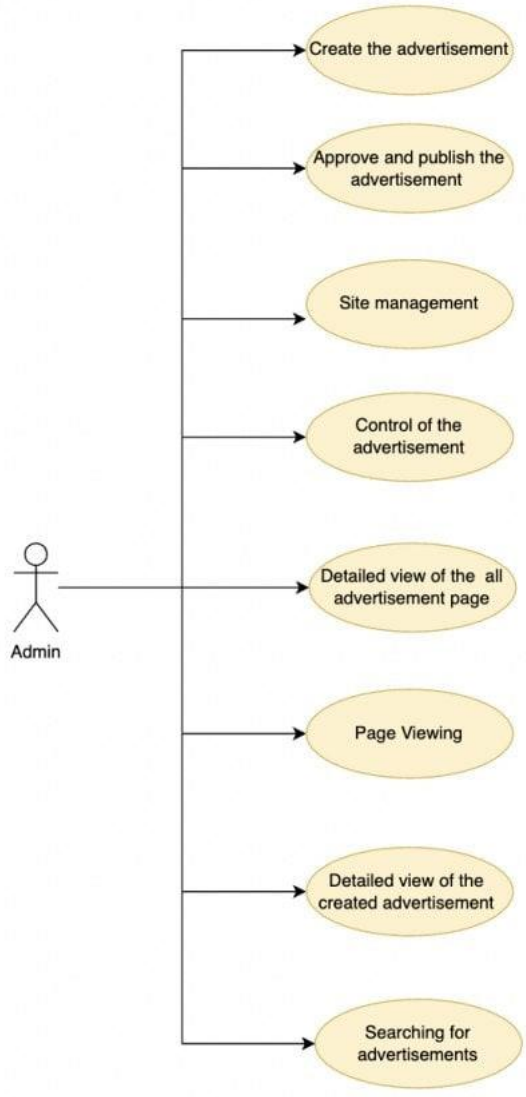

Рисунок 3.3 - UML- діаграму ролі адміністратора

Аналізуючи UML- діаграму для адміністратора, то ми отримуємо уявлення проте, як потрібно реалізувати серверну частину коду. До прав адміністратора відноситься такий функціонал:

- Створення оголошення;
- Перевірка та публікування оголошень;
- Маніпулювати оголошеннями;
- Перегляд усіх існуючих оголошень;
- Детальний перегляд обраного оголошення;
- Пошук оголошень.

UML-діаграма для ролі звичайного користувача:

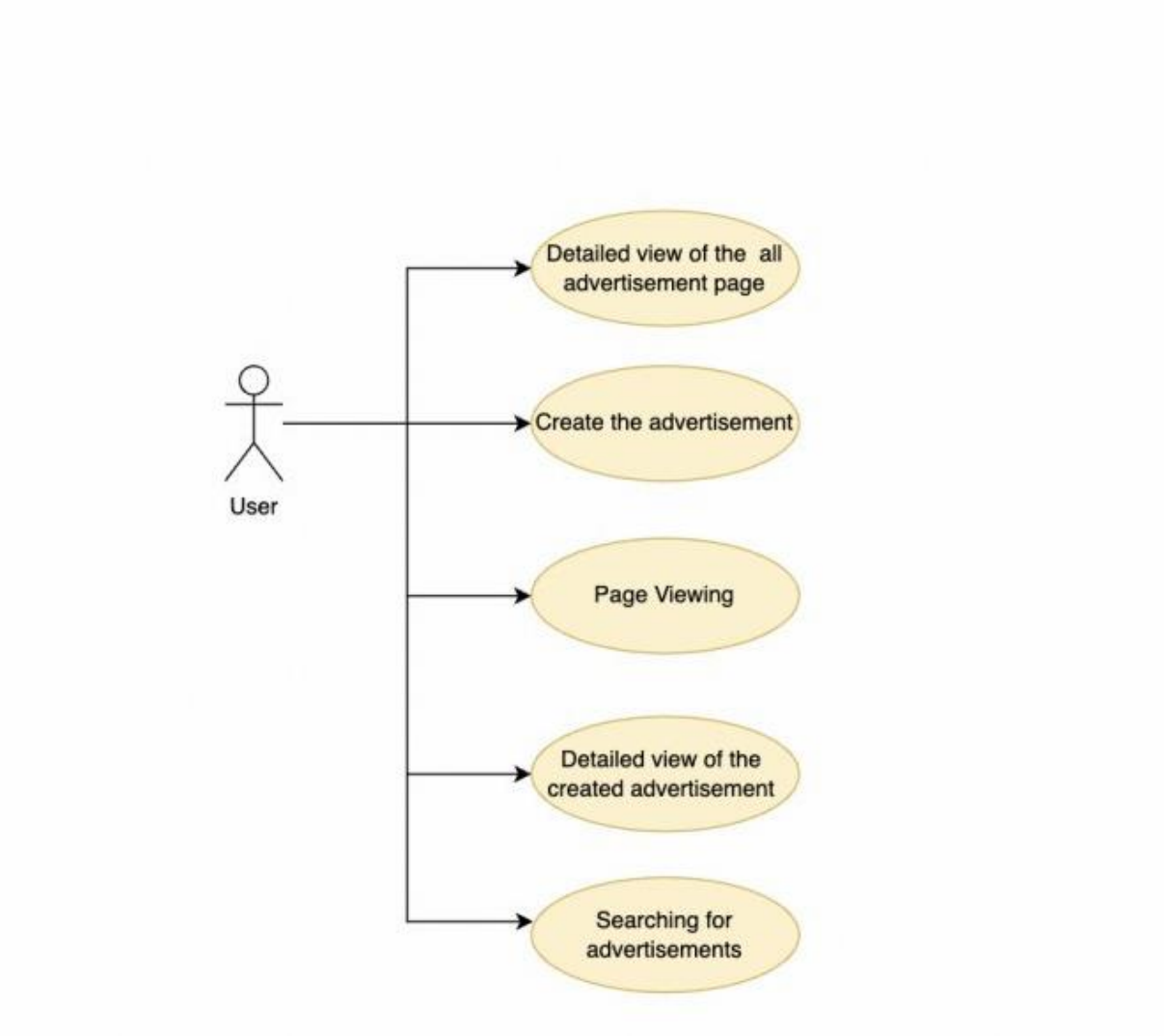

Рисунок 3.4 - UML- діаграму ролі звичайного користувача

Аналізуючи UML- діаграму для звичайного користувача, то ми отримуємо уявлення проте, як потрібно реалізувати серверну частину коду. До прав звичайного користувача відноситься такий функціонал:

- Створення оголошення;
- Перегляд усіх існуючих оголошень;
- Детальний перегляд обраного оголошення;
- Пошук оголошень.

Після обраного стеку технологій, була реалізована частина бек-енд функціоналу.

Код чітко розбитий на шар. Перший шар - це зовнішній вигляд програми, він надсилає запит на шар нижче – шар контролерів. Запит виконується по https. Його обробляє шар контролерів. Шар контролерів знаходить потрібний сервіс. У сервісі виконується вся логіка і йде звернення до бази. В сервісах виконується пошук додання до бази та авторизація. Backend реалізований з використанням DI патерну. Патерн дозволяє не бути залежним від реалізацій, а створювати абстракції й підставляти під них необхідні реалізації. Di реалізований за допомогою фрейморку NestJS. В App Module.ts описані всі залежності.

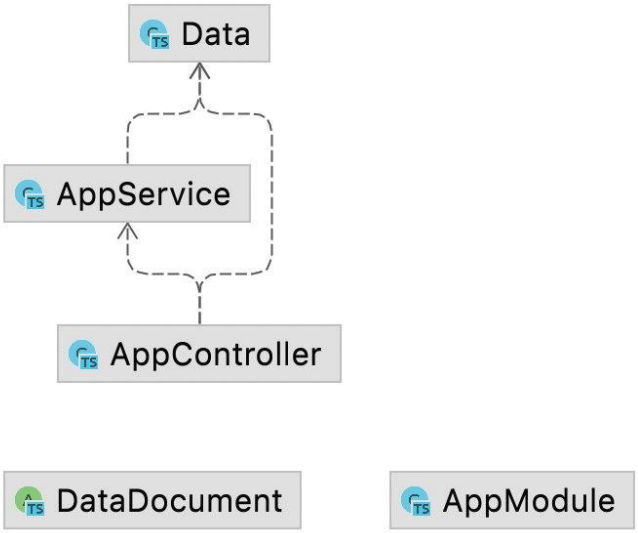

Рисунок 3.5 – Відображення діаграми класів програми

Переваги використання запровадження залежностей:

- Допомагає у модульному тестуванні;
- Кількість шаблонного коду скорочується, оскільки ініціалізація залежностей виконується компонентом інжектора;
- Розширення програми стає ще простішим;
- Допомагає зменшити зв'язок коду, що важливо при розробці додатків.

Впровадження залежностей відповідальне за:

- створення об'єктів;
- уявлення у тому, які класи потрібні цим об'єктам;
- І надання залежностей цим об'єктам.

Datas – це ім’я колекції, яка відповідає за зберіганням даних.

```
import { Prop, Schema, SchemaFactory } from '@nestjs/mongoose';
import { Document } from 'mongoose';

export type DataDocument = Data & Document;

@Schema()
export class Data {
  @Prop({ type: String, text: true })
  data: string;
  @Prop({ type: String, text: true })
  name: string;
  @Prop({ type: String, text: true })
  phone: string;
  @Prop({ type: String, text: true })
  shortName: string;
}

export const DataSchema = SchemaFactory.createForClass(Data);
```

Fulltext search - MongoDB надає текстові індекси для підтримки текстових пошукових запитів за вмістом рядка. текстові індекси можуть включати будь-яке поле, значення якого є рядком або масивом рядкових елементів.

Щоб виконувати текстові пошукові запити, у вашій колекції повинен бути текстовий індекс. Колекція може мати лише один індекс текстового пошуку, але цей індекс може охоплювати кілька полів. В коді Fulltext search реалізован:

```
    async search(text: string) {
      await DataSchema.index({ '$**': 'text' });
      await DataSchema.index({ data: 'text' });

      return this.model.find(
       { $text: { $search: text } },
       { score: { $meta: 'textScore' } },); }
```

В файлі main.ts – головний файл, з якого починається старт нашої програми.

```
import { NestFactory } from '@nestjs/core';
import { AppModule } from './app.module';
import { FastifyAdapter, NestFastifyApplication } from "@nestjs/platform-fastify";
import { DocumentBuilder, SwaggerModule } from '@nestjs/swagger';
async function bootstrap() {
const app = await NestFactory.create<NestFastifyApplication>(
   AppModule,
   new FastifyAdapter(),
 );
const config = new DocumentBuilder()
   .setTitle('Scammer')

   .setVersion('1.0')
   .addBearerAuth()
   .build();
const document = SwaggerModule.createDocument(app, config);
SwaggerModule.setup('api/docs', app, document);
await app.listen(3000);}
bootstrap();
```

Код буде наведений у додатку А.

3.3 Проектування користувацької частини

Оскільки, головною метою розробки – це зберігання та відображення інформації, то користувацький інтерфейс повинен відповідати таким вимогам, як:

- Простим у використанні;
- Відображення головної інформації;
- Інтуїтивним.

Оперуючи вище зазначеними вимогам, було розроблено користувацький інтерфейс.

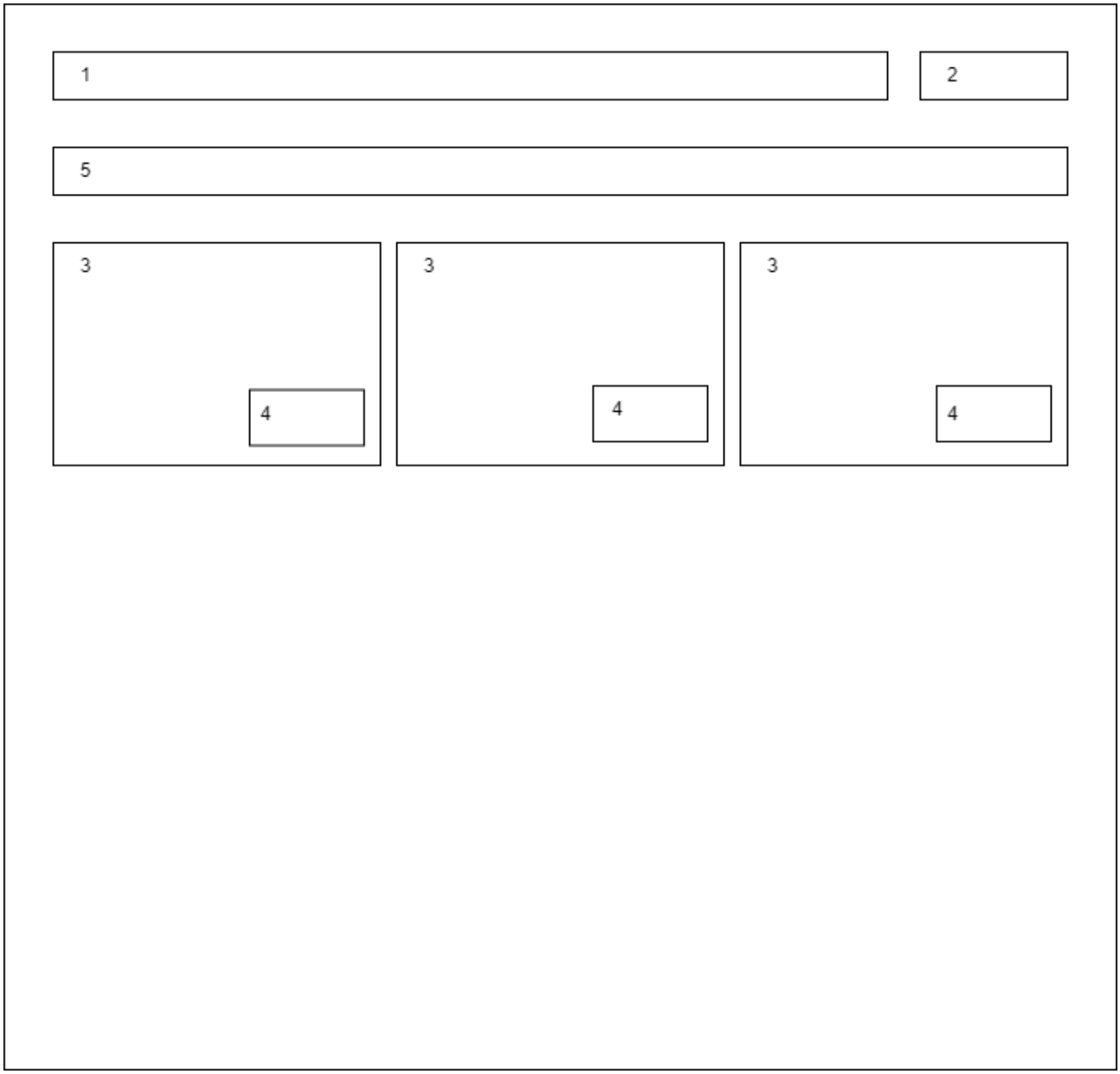

Рис. 3.6 - Проектування стартової сторінки

Дивлячись на зображення початкової сторінки, можна поміти, що на ній відображені найбільше важливіший фунціонал:

- Під цифрою 1 – поле для пошуку;
- Під цифрою 2 – кнопка, яка буде відповідати за пошук. При її натисканні буде спрацьовувати запит;

- Під цифрами 3 – зображення самих оголошень, та кратку інформацію про кожне з них;
- Під цифрами 4 – кнопки, які будуть відкривати оголошення, де буде більше детальне описання його;
- Під цифрою 5 – кнопка, яка буде відкривати форму для створення та надсилання оголошення на перевірку.

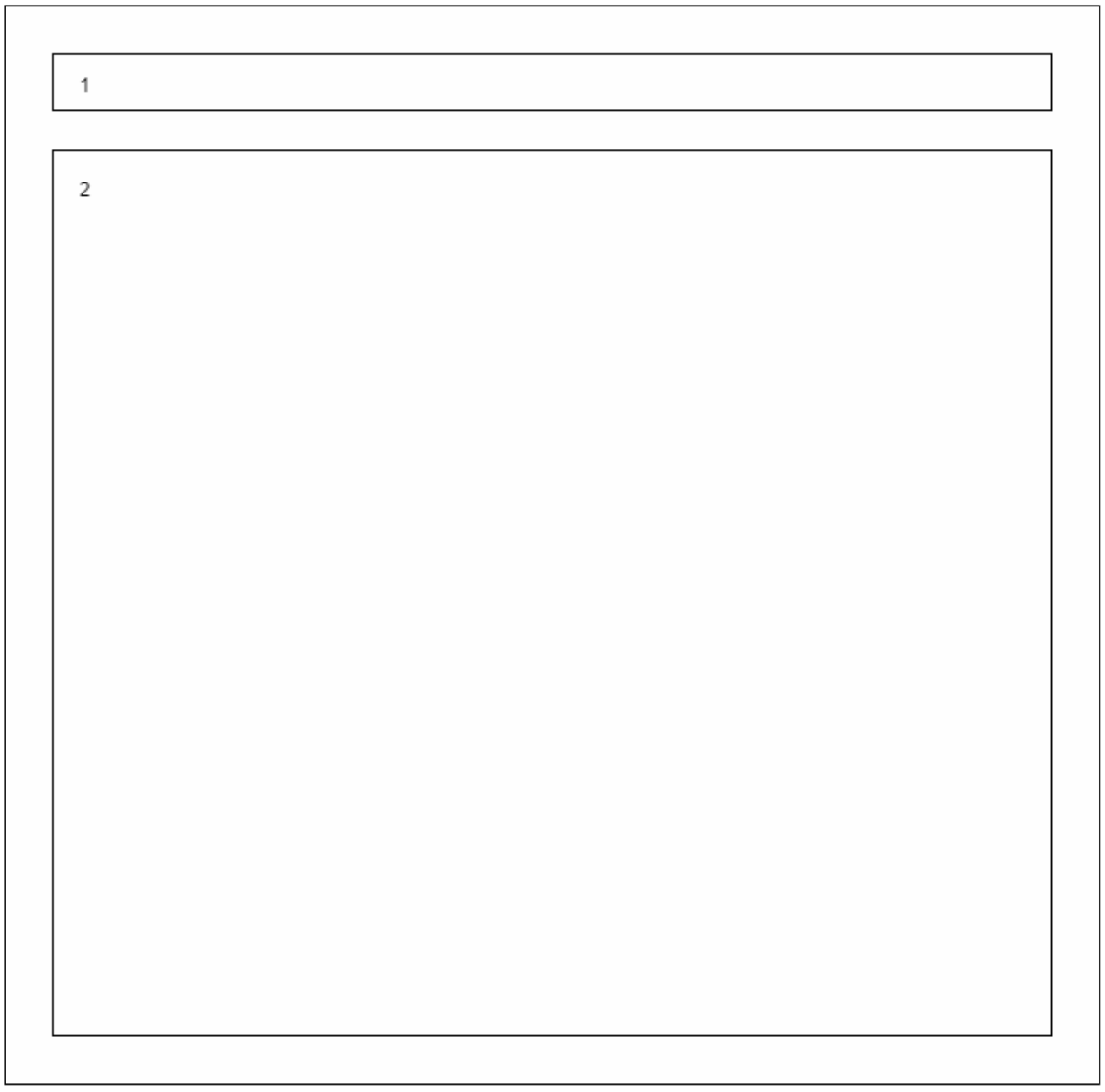

Рис. 3.7 - Проектування сторінки з детальним описом оголошення

Дивлячись на зображення сторінки з детальним описом оголошення, можна помітити, що на ній відображені найбільше важливіший фунціонал:

- Під цифрою 1 – кнопка для повернення на головну сторінку;

- Під цифрою 2 – блок, який буде відображувати детальну інформацію про оголошення.

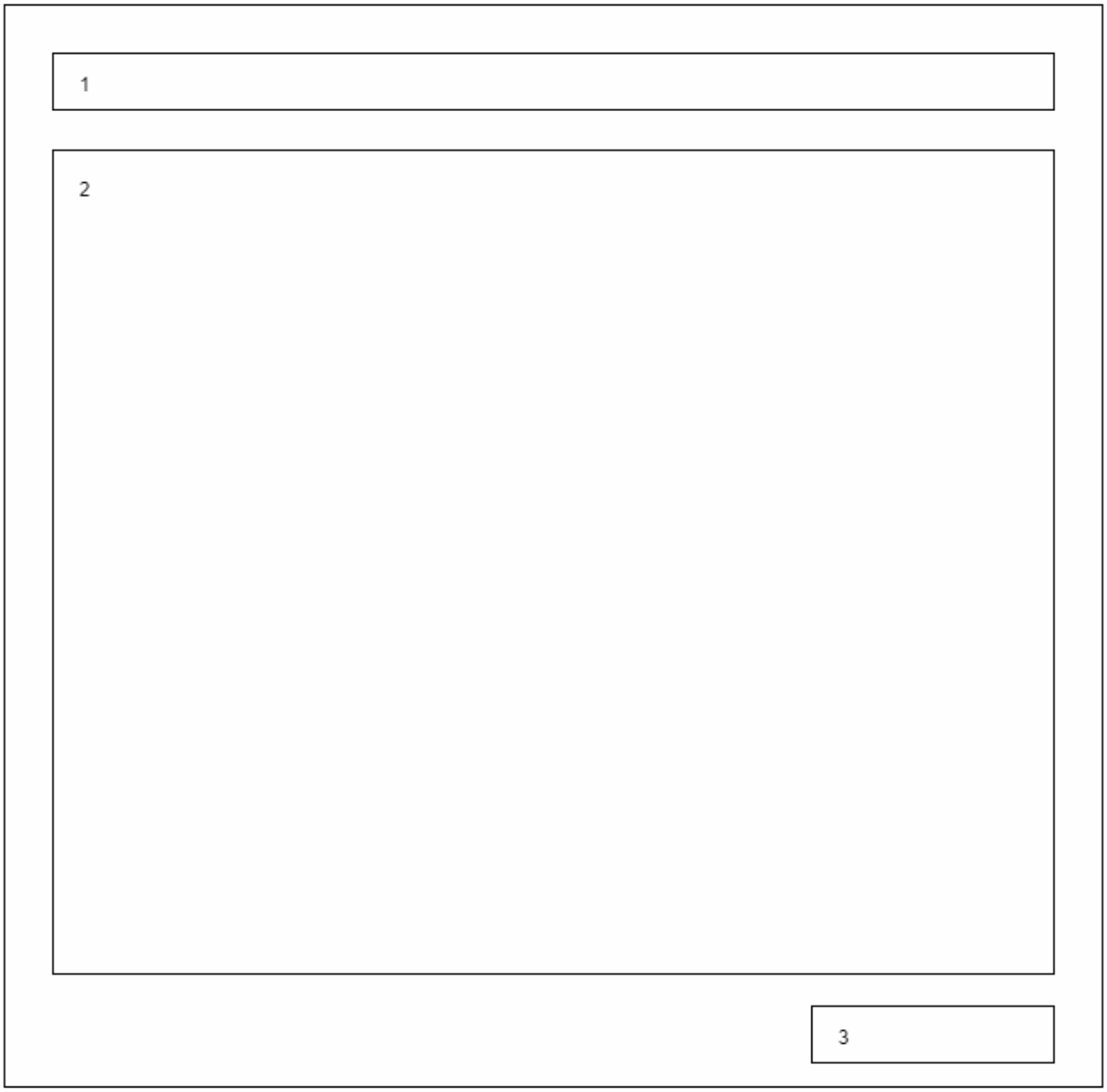

Рис. 3.8 - Проектування сторінки від вигляду адміністратору з детальним описом оголошення

Дивлячись на зображення сторінки з детальним описом оголошення, можна поміти, що на ній відображені найбільше важливіший фунціонал:

- Під цифрою 1 – кнопка для повернення на головну сторінку;
- Під цифрою 2 – блок, який буде відображувати детальну інформацію про оголошення;
- Під цифрою 3 – кнопка для публікування оголошення

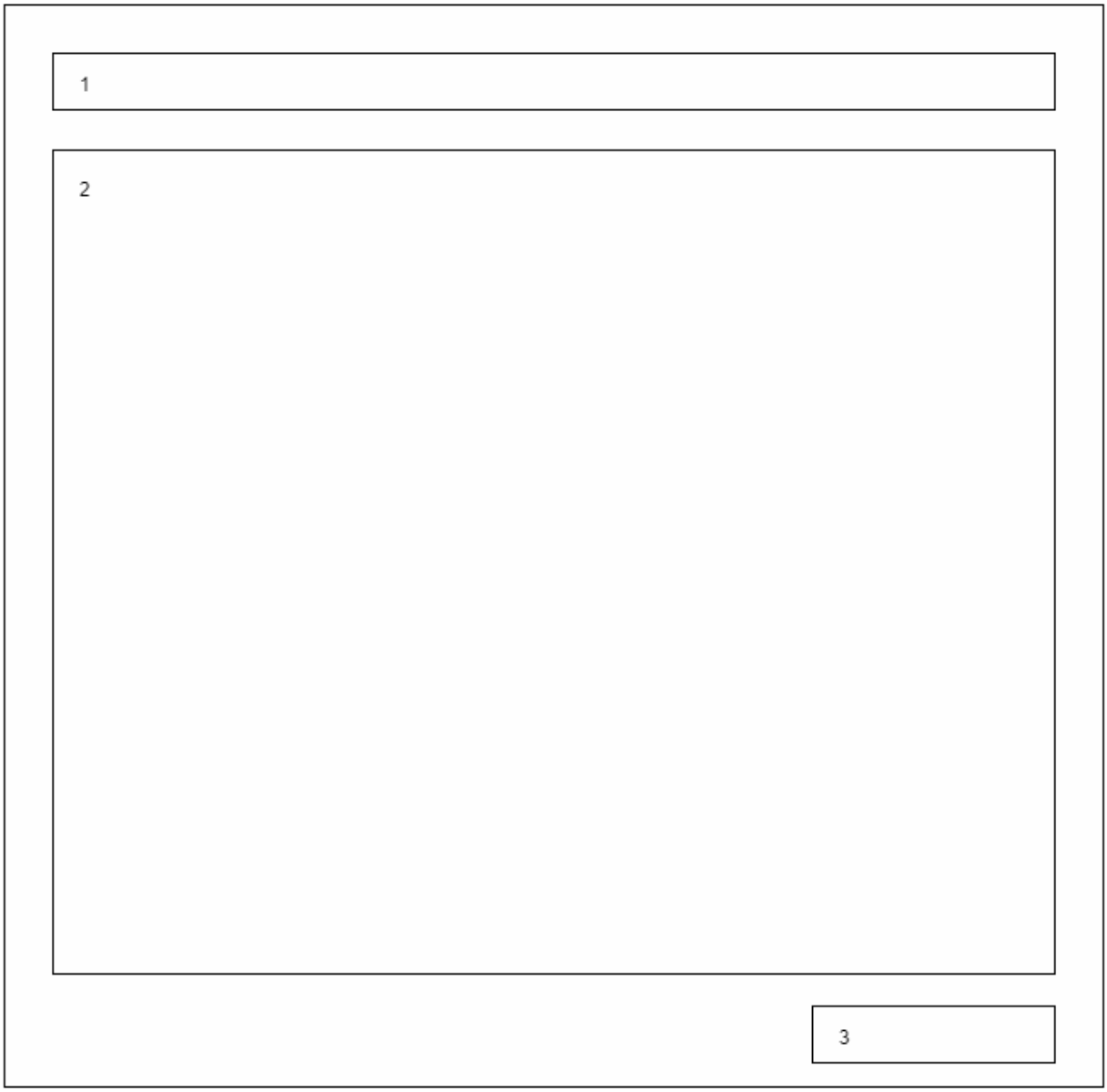

Рис. 3.9 - Проектування сторінки для створення оголошення

Дивлячись на зображення сторінки для створення оголошення, можна помітити, що на ній відображені найбільше важливіший фунціонал:

- Під цифрою 1 – кнопка для повернення на головну сторінку;
- Під цифрою 2 – блок, який буде містити поля для заповнення;
- Під цифрою 3 – кнопка для надсилання оголошення на перевірку

3.4 Тестування реалізованого інформаційного порталу

Головною сторінкою є index.html. Користувач, який заходить до інформаційного ресурсу бачить вже раніше створенні оголошення, поле для пошуку, кнопку на створення оголошення.

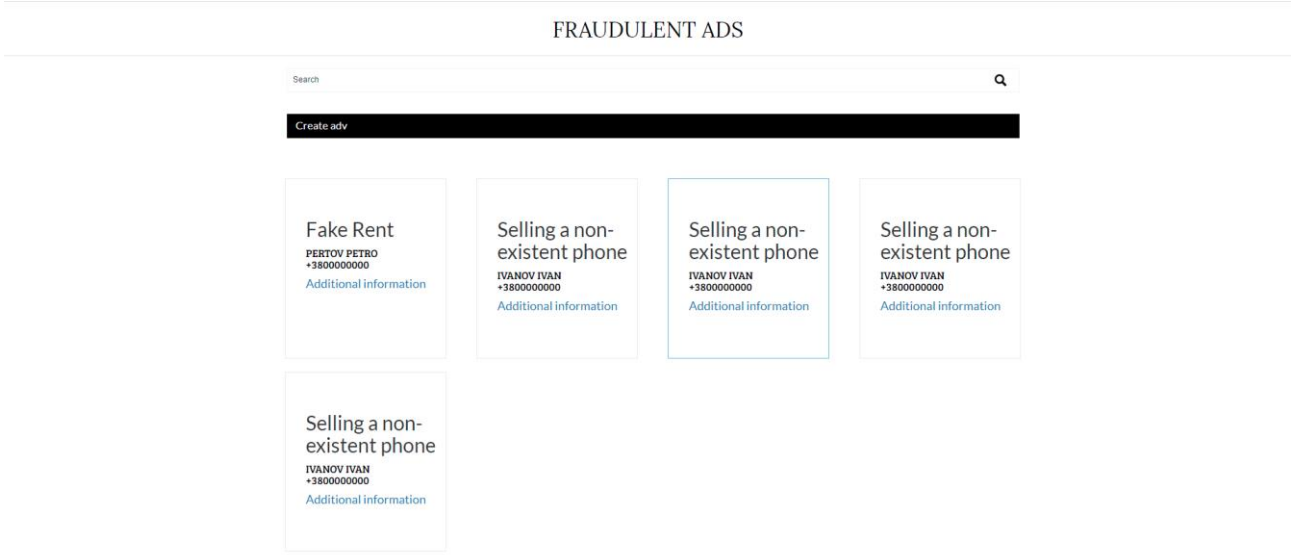

Рисунок 3.10 - Головна сторінка

Якщо користувач натисне на кнопку «Create ads», його перенаправить на сторінку для створення оголошення, на якій будуть поля:

- Поле «Short name» - це поле відповідає за коротке ім’я, яке буде відображатись на головній сторінці;
- Поле «Name or Names» - це поле відповідає за ім’я або імена, які будуть відображатись на головній сторінці;
- Поле «Phone number» - це поле відповідає за відображення номера на головній сторінці;
- Поле «Additional info» - це поле відповідає за відображення повної інформації на сторінці оголошення.

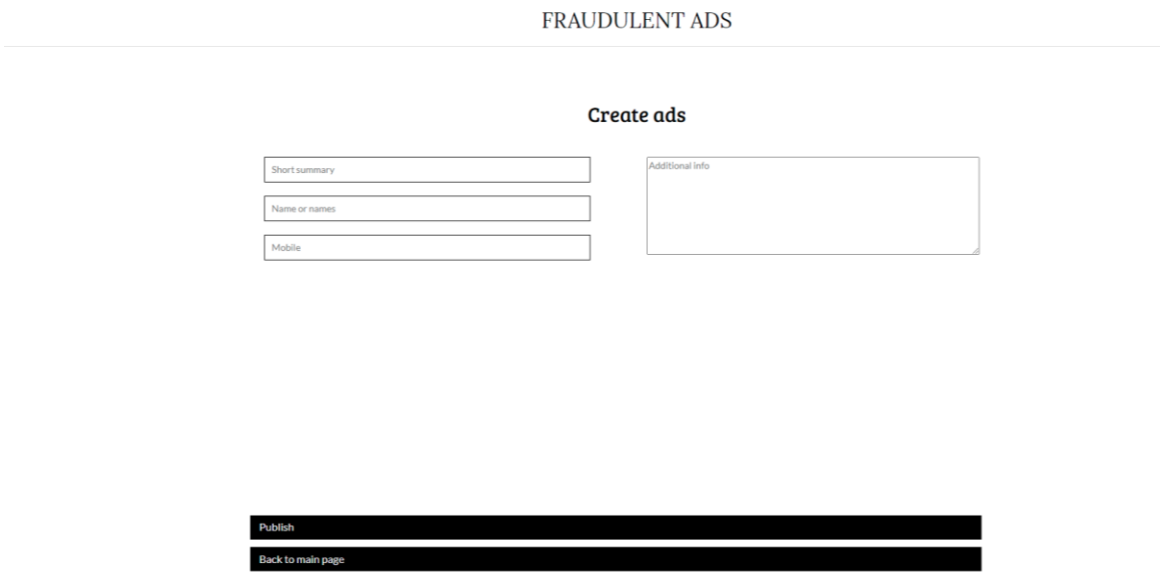

Рисунок 3.11 - Сторінка відображення для створення оголошення

Якщо користувач не хоче публікувати оголошення, то він може повернутись на головну сторінку за допомогою кнопки «Back to main page»

Створенні оголошення можна переглянути для більше детальної інформації натиснучи кнопку «Additional information». На сторінці оголошення відображаються наступна інформація:

- Короткий заголовок;
- Номер телефону;
- Ім’я шахраїв;
- Повна оголошення.

Рисунок 3.12 - Сторінка відображення оголошення.

Під час перегляду оголошення від іменні адміністратора з’являється додаткова кнопка для публікації оголошення, яке пройшло верифікацію.

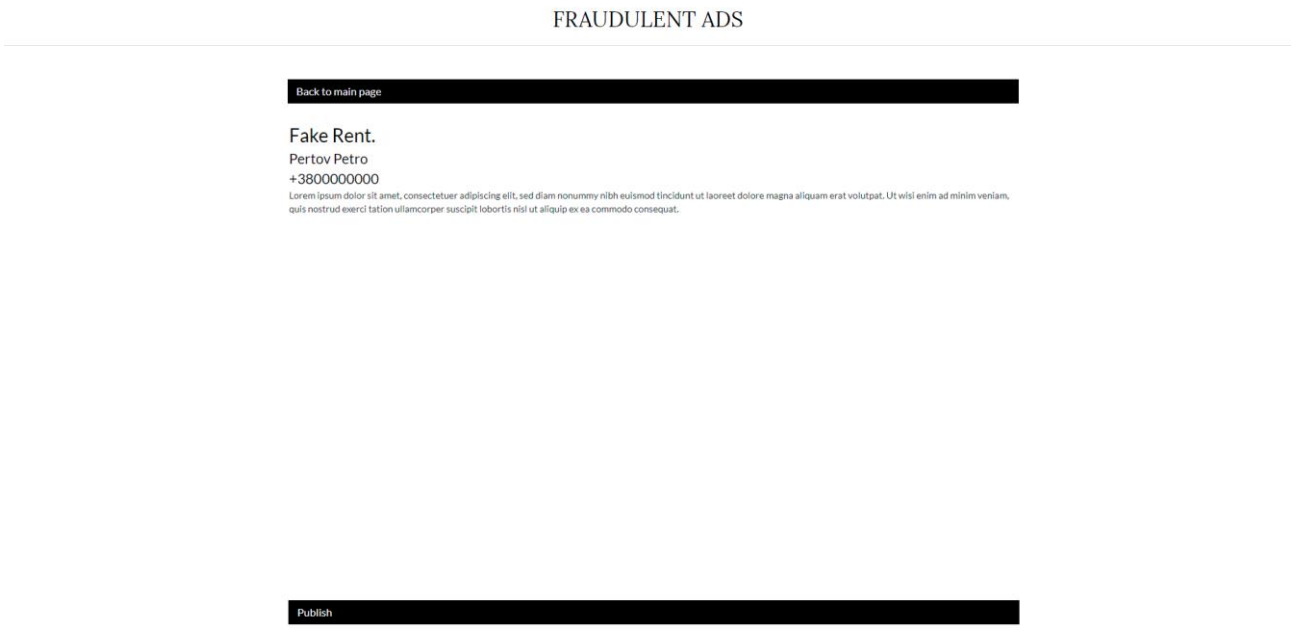

Рисунок 3.13 - Сторінка відображення оголошення від імені адміністратора

ВИСНОВКИ

В квалікаційній магістерській робот проведені дослідження по розробці інформаційної технології для відображення вже створених оголошень, створення оголошень та виявлення шахрайських оголошень. Результати досліджень:

1. Досліджено статистку щодо шахрайства в Україні за певні проміжки часу, проведено відношення між тим, скільки сталось шахрайств та скільки з них було викрито;
2. Проведено аналіз існуючих інформаційних порталів та їх функціональних можливостей. Їх недоліки й переваги;
3. Досліджено алгоритм для розпізнавання оголошень на шахрайство за допомогою нейролінгвістичного програмування. Було проаналізовані аналітичні мови, такі як: Word2vec, BM25, й TF-IDF;
4. Здійснено аналіз сучасних стеків технологій, їх переваги й недоліки. Та оперуючи перевагами було реалізовано комп'ютерну реалізацію:
    a. Серверну частину за допомогою Node.js, Nest.js, Fastify;
    b. Реалізацію для обробки, зберігання даних було використано NoSql – MongoDB;
    c. Клієнтську частину було реалізовано за допомогою ReactJS.
5. Проведене тестування роботи інформаційного порталу та його функціоналу.

Щодо визначення технологій на шахрайство, то надалі буде реалізовано один з існуючих алгоритмів, який буде використовуватись на інформаційному порталі.

Так же в майбутньому, сайт буде встановлений на хостинг та будуть на ньому реалізовуватись додаткові функціоналі можливості, які будуть спрямовані на поліпшення користування для користувачів.

З наступних додаткового функціоналу буде додано:

- Завантаження фотографії;
- Лічильник кількості переглядів оголошення.

Так же планується підтримка порталу на всіх його проміжках існування, виправлення виявлених багів, та розділів на яких будуть описані найчастіші види шахрайств, щоб уникнути їх у майбутньому.

СПИСОК ЛІТЕРАТУРИ

1. Liqpay [Електронний ресурс] – Режим доступу до ресурсу: https://www.liqpay.ua/en.

2. Google Pay [Електронний ресурс] – Режим доступу до ресурсу: https://pay.google.com/intl/uk_ua/about/.

3. Apple Pay [Електронний ресурс] – Режим доступу до ресурсу: https://www.apple.com/ru/apple-pay/.

4. Шахрайство в інтернеті [Електронний ресурс] – Режим доступу до ресурсу: https://minjust.gov.ua/m/yak-ne-stati-jertvoyu-shahraiv-v-interneti-ta-scho-robiti-yakscho-vi-potrapili-u-pastku.

5. Види шахрайств [Електронний ресурс] – Режим доступу до ресурсу: https://thepage.ua/finance/moshennichestvo-v-internete-kakim-byvaet-i-kak-sebya-obezopasit.

6. Статистика шахрайств в Україні [Електронний ресурс] – Режим доступу до ресурсу: https://finclub.net/analytics/moshenniki-lishayut-deneg-doverchivykh-ukraintsev.html.

7. Cyberpolice [Електронний ресурс] – Режим доступу до ресурсу: https://cyberpolice.gov.ua/.

8. База шахраїв України [Електронний ресурс] – Режим доступу до ресурсу: http://foe.omg.com.ua/foe/

9. Privatbank.ua/ru/know-person-get-money [Електронний ресурс] – Режим доступу до ресурсу: https://privatbank.ua/ru/know-person-get-money/region.

10. Ignacio P. M. Natural language processing for scam detection. Classic and alternative analysis techniques / Palacio Marín Ignacio., 2019. – (Msc in BigData and DataScience).

11. Wei Di. Deep Learning Essentials / Wei Di, Anurag Bhardwaj, Jianing Wei., 2019.

12. Jalaj Thanaki. Python Natural Language Processing / Jalaj Thanaki., 2015.

13. NodeJS [Електронний ресурс] – Режим доступу до ресурсу: https://nodejs.org/uk/

14. Node.js в действии / М. Кантелон, М. Хартер, Н. Райлих., 2018.

15. Nestjs [Електронний ресурс] – Режим доступу до ресурсу: https://nestjs.com/

16. Fastify [Електронний ресурс] – Режим доступу до ресурсу: https://www.fastify.io/.

17. mongodb.com [Електронний ресурс] – Режим доступу до ресурсу: https://www.mongodb.com/

18. PosgtreSQL [Електронний ресурс] – Режим доступу до ресурсу: https://ru.reactjs.org/

19. ReactJS [Електронний ресурс] – Режим доступу до ресурсу: https://ru.reactjs.org/

ДОДАТОК А

app.controller.spec.js

```
import { Test, TestingModule } from '@nestjs/testing';
import { AppController } from './app.controller';
import { AppService } from './app.service';

describe('AppController', () => {
  let appController: AppController;

  beforeEach(async () => {
    const app: TestingModule = await Test.createTestingModule({
      controllers: [AppController],
      providers: [AppService],
    }).compile();

    appController = app.get<AppController>(AppController);
  });

  describe('root', () => {
    it('should return "Hello World!"', () => {
      expect(appController.getHello()).toBe('Hello World!');
    });
  });
});
```

app.controller.ts

```
import { Body, Controller, Get, Param, Post, Query } from '@nestjs/common';
import { AppService } from './app.service';
import { Data } from './data';

@Controller()
export class AppController {
  constructor(private readonly appService: AppService) {}

  @Get('/search')
  async search(@Query('text') text: string): Promise<Data[]> {
    return await this.appService.search(text);
  }

  @Post('/publish')
  async insert(@Body() text: string) {
    await this.appService.insert(text);
  }
  @Post('/login')
  async login(@Body() text: LoginDTO) {
```

```
    await this.appService.login(text);
  }
}
```

app.module.ts

```
import { Module } from '@nestjs/common';
import { AppController } from './app.controller';
import { AppService } from './app.service';
import { MongooseModule } from '@nestjs/mongoose';
import { Data, DataSchema } from './data';

@Module({
  imports: [
    MongooseModule.forRoot(

'mongodb+srv://test:test@cluster0.ufnwk.mongodb.net/myFirstDatabase?retryWrites=true&w=majority',
    ),
    MongooseModule.forFeature([{ name: Data.name, schema: DataSchema }]),
  ],
  controllers: [AppController],
  providers: [AppService],
})
export class AppModule {}
```

app.service.ts

```
import { Injectable } from '@nestjs/common';
import { InjectModel } from '@nestjs/mongoose';
import { Data, DataSchema } from "./data";
import { model, Model } from 'mongoose';

@Injectable()
export class AppService {
  constructor(@InjectModel(Data.name) private model: Model<Data>) {

  }
  async insert(text: string) {
    await DataSchema.index({ '$**': 'text' });
    await DataSchema.index({ data: 'text' });
    const doc = new this.model({ data: text });
    return doc.save();
  }

  async login(text: string) {
    await DataSchema.index({ '$**': 'text' });
    await DataSchema.index({ data: 'text' });
```

```
    const doc = new this.model({ data: text });
    return doc.save();
  }

  async search(text: string) {
    await DataSchema.index({ '$**': 'text' });
    await DataSchema.index({ data: 'text' });

    return this.model.find(
      { $text: { $search: text } },
      { score: { $meta: 'textScore' } },
    );
  }
}
```

data.ts

```
import { Prop, Schema, SchemaFactory } from '@nestjs/mongoose';
import { Document } from 'mongoose';

export type DataDocument = Data & Document;

@Schema()
export class Data {
  @Prop({ type: String, text: true })
  data: string;
}

export const DataSchema = SchemaFactory.createForClass(Data);
```

main.ts

```
import { NestFactory } from '@nestjs/core';
import { AppModule } from './app.module';
import { FastifyAdapter, NestFastifyApplication } from "@nestjs/platform-fastify";
import { DocumentBuilder, SwaggerModule } from '@nestjs/swagger';

async function bootstrap() {
  const app = await NestFactory.create<NestFastifyApplication>(
    AppModule,
    new FastifyAdapter(),
  );
  const config = new DocumentBuilder()
    .setTitle('Scammer')
    .setVersion('1.0')
    .addBearerAuth()
    .build();
```

```
  const document = SwaggerModule.createDocument(app, config);
  SwaggerModule.setup('api/docs', app, document);

  await app.listen(3000);
}
bootstrap();
```