Міністерство освіти і науки України

Житомирський державний технологічний університет

Факультет інформаційно-комп'ютерних технологій

Кафедра інженерії програмного забезпечення

ПОЯСНЮВАЛЬНА ЗАПИСКА

до магістерської атестаційної роботи

на тему: «Інтерактивний web-сервіс для пошуку

домашніх тварин»

Виконав студент 2-го курсу, групи \_\_\_\_\_\_\_\_  
спеціальності 121 «Інженерія програмного забезпечення»

В.М.Данюк

Керівник к.т.н., доцент, доцент кафедри ІПЗ

О.І.Грабар

Рецензент к.т.н., доцент, доцент кафедри ІПЗ

В.О.Скачков

Житомир – 2018

ЖИТОМИРСЬКИЙ ДЕРЖАВНИЙ ТЕХНОЛОГІЧНИЙ УНІВЕРСИТЕТ

ФАКУЛЬТЕТ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(назва прописними)

КАФЕДРА\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(назва прописними)

СПЕЦІАЛЬНІСТЬ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(назва прописними)

ЗАТВЕРДЖУЮ

Зав. кафедри \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(назва кафедри)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_\_ р.

**ЗАВДАННЯ**

на магістерську атестаційну роботу

Студента \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Тема роботи: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Затверджена Наказом університету від «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_ р. № \_\_\_\_\_\_\_\_\_\_\_

Термін здачі студентом закінченої роботи \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Вихідні дані роботи (зазначається предмет і об’єкт дослідження) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Консультанти з атестаційної роботи магістра із зазначенням розділів, що їх стосуються

|  |  |  |  |
| --- | --- | --- | --- |
| Розділ | Консультант | Підпис, дата | |
| Завдання видав | Завдання прийняв |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис)

**Календарний план**

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назва етапів магістерської атестаційної роботи | Термін виконання етапів роботи | Примітка |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис)

Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис)

РЕФЕРАТ

Магістерська атестаційна робота складається з вступу, 3-х розділів, висновків, списку використаних джерел із \_\_ найменувань (\_\_ стор.), \_\_ додатків (\_\_ стор.), \_\_ таблиць та \_\_ ілюстрацій по тексту. Повний обсяг роботи становить \_\_ сторінок, з них \_\_\_ сторінок основного тексту.

Метою роботи є розробка інтерактивного web-сервісу для пошуку домашніх тварин завдяки якій користувачі зможуть розміщувати оголошення про втрачених/знайдених домашніх тварин, що містить модулі локалізації, фільтрації за основними критеріями та модерації.

В роботі визначено основні завдання на розробку системи, проаналізовано аналоги розробленої системи, обґрунтовано вибір архітектурного патерна для реалізації системи – MVVM. Наведено сценарії роботи програмного комплексу, загальна структура програмного комплексу, опис бази даних системи (Firebase Realtime NoSQL), опис алгоритмів взаємодії окремих модулів системи, опис об’єктної структури системи (діаграма компонентів), алгоритми роботи основних модулів, реалізованих з використанням мови програмування JavaScript та фреймворку Vue. Також наводиться результати тестування системи.

Ключові слова: БЮРО ЗНАХІДОК ДОМАШНІХ ТВАРИН, ВТРАТИВ, ЗНАЙШОВ, MVVM, FIREBASE, VUE.

ABSTRACT

Master's attestation work consists of introduction, 3 chapters, conclusions, list of used sources from \_\_ names (\_\_ pages), \_\_ applications (\_\_ pages), \_\_ tables and \_\_ illustrations in the text. The full volume of the work is \_\_ pages, including \_\_\_ pages of the main text.

The goal of the work is to develop an interactive web-based search service for pets, through which users will be able to place ads for lost / found pets containing localization modules, filtering by main criteria and moderation.

In the work the main tasks for the development of the system are determined, analogs of the developed system are analyzed, the choice of the architectural model for the system implementation - MVVM is substantiated. The scenarios of the program complex work, the general structure of the program complex, the description of the database of the system (Firebase Realtime NoSQL), the description of the algorithms of interaction of the individual modules of the system, the description of the object structure of the system (the diagram of the components), algorithms of the main modules implemented with the use of the programming language JavaScript and the Vue framework. The results of the system testing are also given.

KEYWORDS: LOST AND FOUND PETS, LOST, FOUND, MVVM, FIREBASE, VUE.

ЗМІСТ

[РЕФЕРАТ 4](#_Toc531539732)

[ABSTRACT 5](#_Toc531539733)

[ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ 8](#_Toc531539734)

[ВСТУП 9](#_Toc531539735)

[РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ ІНТЕРАКТИВНОГО WEB-СЕРВІСУ ПО ПОШУКУ ДОМАШНІХ ТВАРИН 11](#_Toc531539736)

[1.1. Постановка задачі 11](#_Toc531539737)

[1.2. Аналіз аналогів програмного продукту 13](#_Toc531539738)

[1.3. Вибір архітектури інтерактивного web-сервісу для пошуку домашніх тварин 15](#_Toc531539739)

[1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення 19](#_Toc531539740)

[Висновки до розділу 1 25](#_Toc531539741)

[РОЗДІЛ 2. МОДЕЛЮВАННЯ ІНТЕРАКТИВНОГО WEB-СЕРВІСУ ДЛЯ ПОШУКУ ДОМАШНІХ ТВАРИН 26](#_Toc531539742)

[2.1. Принцип роботи Virtual DOM 26](#_Toc531539743)

[2.2. Реактивність у Vue 28](#_Toc531539744)

[2.3. Розробка алгоритму створення оголошень с типом “Побачено” 29](#_Toc531539745)

[Висновки до розділу 2 32](#_Toc531539746)

[РОЗДІЛ 3. ПРОЕКТУВАННЯ ІНТЕРАКТИВНОГО WEB-СЕРВІСУ ДЛЯ ПОШУКУ ДОМАШНІХ ТВАРИН 33](#_Toc531539747)

[3.1. Визначення варіантів використання та об’єктно-орієнтованої структури системи 33](#_Toc531539748)

[3.2. Розробка бази даних системи 39](#_Toc531539749)

[3.3. Проектування та реалізація алгоритмів роботи системи 41](#_Toc531539750)

[3.4 Реалізація інтерактивного web-сервісу для пошуку домашніх тварин 45](#_Toc531539751)

[Висновки до розділу 3 49](#_Toc531539752)

[ВИСНОВКИ 50](#_Toc531539753)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 52](#_Toc531539754)

[ДОДАТКИ 54](#_Toc531539755)

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

|  |  |
| --- | --- |
| CRUD | – create, read, update, delete (*з англ.* створити, прочитати, оновити, видалити) |
| MVVM | – Model-View-ViewModel (*з англ*. Модель–Представлення–Модель представлення) |
| БД | – база даних |
| DOM | – об'єктна модель документа |
| ПЗ | * програмне забезпечення |

ВСТУП

**Актуальність теми.** Щодня десятки людей втрачають і знаходять домашніх улюбленців. Як правило, загублена тварина для її власника має набагато вищу цінність, ніж для іншої людини. Швидкий і зручний пошук та повернення знайдених/загублених домашніх тварин на сьогоднішній день дуже важливий.

Ми можемо по вулиці і раптово натрапити на загублену домашню тварину. На даний час у кожного із нас є мобільний телефон с доступом до інтернету. Таким чином ми можемо сфотографувати загублену тварину і створити оголошення. За допомогою веб-сервісу створити оголошення дуже зручно и швидко.

**Метою випускної роботи** є проектування архітектури, розробка алгоритмів роботи та реалізація ПЗ для знаходження та розміщення об’яв про знайдену/загублену домашню тварину та модулів які поліпшить їх умови.

Для досягнення поставленої мети необхідно вирішити такі основні завдання:

* проведення системного аналізу предмету дослідження;
* визначення основних функцій та особливостей системи;
* вибір технології та засобів реалізації програмного продукту;
* проектування структурних складових та алгоритмів роботи системи.

**Об’єктом дослідження** є автоматизація управління процесами CRUD, що полегшають користувачам їх використовувати.

**Предметом дослідження** є використання веб-орієнтованих технології для автоматизації роботи інтерактивного web-сервісу для пошуку домашніх тварин.

В роботі було використано методи об’єктно орієнтованого проектування, теорії NoSQL баз даних, веб-програмування, об’єктно-орієнтованого програмування та дизайну інтерфейсів.

**Наукова новизна.** В даній роботі запропоновано новий технічний підхід до вирішення завдань інтерактивного web-сервісу для пошуку домашніх тварин. Суть даного підходу полягає у використані прогресивного фреймворку який найшвидший серед своїх аналогів, відображення оголошень на карті Google та використання Firebase Realtime бази яка надає данні в режимі реального часу та бекенд як службу.

**Практичне значення одержаних результатів.** Запропонований технічний підхід дозволяє вирішувати такі завдання, як інтерактивне відображення оголошень на карті Google, відображення інформації про оголошення в реальному часі. За допомогою таких технологій користувач може швидко та зручно створювати або знаходити оголошення про втрачених/знайдених домашніх тварин.

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ ІНТЕРАКТИВНОГО WEB-СЕРВІСУ ПО ПОШУКУ ДОМАШНІХ ТВАРИН

## Постановка задачі

Основною метою створення даного веб-сервісу в тому, що кожен користувач зможе створювати або знаходити оголошення не виходячи з дому і тим самим зможе повернути знайдену/загублену домашню тварину. Одна з особливостей веб-сервісу є те, що всі оголошення позначаються на карті.

В випускній роботі необхідно розробити інтерактивний web-сервіс для пошуку домашніх тварин, засоби управління інформаційним контентом та функціональні модулі, створення, редагування, видалення та пошук у відповідності із заданими даними. Для реалізації поставленого завдання, основними етапами є:

1. Розробити дизайн сайту.
   * Спроектувати загальну концепцію дизайну для головної та внутрішніх сторінок сайту: головна сторінка з картою оголошень, маркери на карті з інформацією про оголошення, спливаюче вікно с формою створення та редагування оголошень, меню; сторінка авторизації; сторінка реєстрації; сторінка профілю користувача; сторінка модерації оголошень.
   * За допомогою мови розмітки HTML 5 і таблиць стилів CSS 3 створити шаблони головної та внутрішніх сторінок.
   * Використання бібліотеки компонентів Vuetify для побудови інтерфейсів сторінки з використанням принципів Google Material Design.
2. Реалізувати основну роботу програми.

2.1. Написати програмні модулі для:

* модуль авторизації користувачів в системі. Розподіл прав доступу користувачів повинен визначатися за наступними ролями: користувач, модератор та адміністратор;
* модуль реєстрації користувачів на сайті – при заповненні обов’язкових полів та правильному введенні користувачем дані він може зареєструватися на сайті;
* модуль створення нового оголошення – після авторизації і заповнені обов’язкових полів користувач зможе створити оголошення;
* модуль фільтрації оголошень типом оголошення, типом, породою, коліром, забарвлення, віком тварини;
* модуль редагування оголошення;
* модуль модерації.

2.2. Реалізувати CRUD операції для всіх колекцій бази даних системи.

1. Реалізувати модулі виводу оголошення та статистики на сайті:
   * Статичний звіт по створених оголошеннях за місяць, кількості користувачів та інше.
   * Модуль виводу карти з оголошеннями за допомогою “Google Maps API”.
   * Контролювання нових оголошень.
2. Реалізувати структуру збереження даних наступного виду:
   * Дані про користувача (Електронна пошта, пароль, роль користувача);
   * Дані про об’яву (тип, статус, дата створення, тип тварини, порода тварини, колір тварини, забарвлення тварини, вік тварини, контактна інформація, координати місця знахідки/пропажі);
3. Створити демонстраційний та тестовий контент сайту.
4. Розробити план тестування системи управління контентом. Зокрема передбачити наступні види тестування:
   * Тестування на функціональність сайту та помилки в його роботі.
   * Кросбраузерне та адаптивне тестування.
   * Тестування безпеки даних.
   * Тестування графічного інтерфейсу та “usability” тестування.
   * Тестування навантаження.

Результатом виконання даних завдань ми отримаємо інтерактивний web-сервіс для пошуку домашніх тварин, який має клієнт-серверну архітектуру, безпечну базу даних, різнопланові функціональні можливості, що забезпечують коректну роботу сервера та зрозумілий клієнтський інтерфейс, а також панель модерації та адміністрації з обмеженим доступом.

## Аналіз аналогів програмного продукту

На сучасному ринку програмних засобів представлено велика кількість, але в основному вони прив’язані до одного міста. Переглянувши їх я зміг відібрати тільки 4 веб-сервіса для аналізу. Для розгляду аналогів, за критеріями популярності та позитивного рейтингу вибрані веб-сервіси наведені в табл. 1.1.

Опишемо кожну із розглянутих програмних систем.

Всеукраинского онлайн-бюро находок “Верни”. Перевагами цього сервісу є те, що він має фільтр по регіонам, для створення об’яви не потрібно реєструватись та пошук по об’явам. Недоліками сервісу є те, що він не підтримує локалізацію інших мов, в об’явах немає карти на якій відображається місце знахідки/пропажі також сервіс немає фільтра по іншим критеріям об’яви.

PETSI. Сервіс має сучасний адаптивний дизайн та пошук по об’явам. Особливість цього сервісу є те, що він має магазин та довідник порід тварин. Недоліками сервісу є те, що він не має фільтрації оголошень та локалізації для інших мов.

LOSTPETS. Переваги сервісу в тому, що він надає різні послуги, відображає місце знахідки/втрати на карті та має фільтрацію по оголошенням. Недоліки сервісу в тому, що він дуже повільно працює, застарілий дизайн, не коректна верстка та не має пошуку.

LUCKFIND. Переваги сервісу в тому, що він має сучасний дизайн, локалізацію трьома мовами, пошук по об’ява, відображає місце знахідки/пропажі на карті, розроблений на сучасній мові програмування AngularJS. Недоліки його в тому, що він не має фільтра, меню розміщено тільки внизу сторінки та авторизацію яку важко знайти.

Для того, щоб зробити якісь висновки потрібно порівняти їх на функціональні можливості, тому порівняння будуть наведені в табл. 1.2.1.

Таблиця 1.2.1

Функціональні можливості

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Назва | Всеукраинское виртуальное онлайн-бюро находок "Верни" | PETSI | LOSTPETS | Громадська організація «LUCKFIND» |
| Електронний вхід/вихід | - | + | + | + |
| Пошук | + | + | - | + |
| Фільтр | + | - | + | - |
| Служба підтримки | - | - | - | + |
| Відображення місця втрати та знахідки на карті | - | - | + | + |
| Адміністративні функції | - | - | + | + |
| Можливість залишити коментарій | + | - | + | + |
| Мобільна версія сайту | + | + | + | + |

Переглянувши список аналогів інтерактивних web-сервісів для пошуку домашніх тварин можна зробити висновки, що майже всі програми однотипні і тільки один сервіс відрізняється від інших і це “LUCKFIND”.

Таким чином, основним я зробив висновок, що мій сервіс повинен мати такі функціональні можливості:

− Фільтр за різними критеріями.

− Дані повинні оновлюватись оперативно швидко.

− Простота у використанні.

− Електронний вхід/вихід.

− Пошук.

− Локалізація.

− Відображення місця втрати та знахідки на карті.

− Модерація об’яв.

## Вибір архітектури інтерактивного web-сервісу для пошуку домашніх тварин

Для реалізації даного проекту було використано клієнт-серверну архітектуру.



Рис. 1.3.1. Клієнт-серверна архітектура

Архітектура клієнт-сервер є одним із архітектурних шаблонів програмного забезпечення та є домінуючою концепцією у створенні розподілених мережевих застосувань і передбачає взаємодію та обмін даними між ними. Вона передбачає такі основні компоненти:

* набір серверів, які надають інформацію або інші послуги програмам, які звертаються до них;
* набір клієнтів, які використовують сервіси, що надаються даними серверами;
* мережа, яка забезпечує безперебійну взаємодію між клієнтами та серверами.

***Сервер* –** це набір програм, які контролюють виконання різних процесів. Відповідно, цей набір програм встановлений на будь-якому комп'ютері. Часто комп'ютер, на якому встановлено сервер, і називають сервером. Основна функція комп'ютера-сервера - по запиту клієнта запустити який-небудь певний процес і відправити клієнту результати його роботи.

***Клієнтом*** називають будь-який процес, який користується послугами сервера. Клієнтом може бути як користувач, так і програма. Основне завдання клієнта - виконання програми та здійснення зв'язку з сервером, коли цього потребує програма. Тобто клієнт повинен надавати користувачеві інтерфейс для роботи з додатком, реалізовувати логіку його роботи і при необхідності відправляти завдання серверу.

Для реалізації веб-сервісу по пошуку домашніх тварин був обраний фреймворк Vue який використовує архітектурний патерн MVVM.

В класичному варіанті, MVVM складається з трьох частин, які і дали йому назву (рис.1.3.2). Розглянемо їх особливості більш детально:

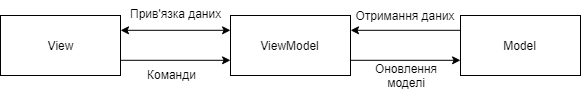


Рис.1.3.2. Узагальнена архітектура MVVM

MVVM полегшує відокремлення розробки графічного інтерфейсу від розробки бізнес логіки (бек-енд логіки), відомої як модель (можна також сказати, що це відокремлення представлення від моделі). Модель представлення є частиною, яка відповідає за перетворення даних для їх подальшої підтримки і використання. З цієї точки зору, модель представлення більше схожа на модель, ніж на представлення і оброблює більшість, якщо не всю, логіку відображення даних. Модель представлення може також реалізовувати патерн медіатор, організовуючи доступ до бек-енд логіки навколо множини правил використання, які підтримуються представленням.

MVVM використовується для відокремлення моделі та її відображення. Необхідністю цього є надання можливості змінювати їх незалежно одну від одної. Наприклад, розробник працює над логікою роботи з даними, а дизайнер — з користувацьким інтерфейсом.

MVVM була створена з метою поділу праці дизайнера і програміста, що є неможливим, коли Java-розробник намагається побудувати GUI в Swing або розробник на Visual C++ намагається створити користувальницький інтерфейс в MFC. Розробники — кмітливі хлопці і мають безліч навичок, але створення зручних і привабливих інтерфейсів вимагає абсолютно інших талантів, ніж ті, якими вони володіють. Ця робота більше підходить для дизайнерів інтерфейсів. Хороші дизайнери інтерфейсів краще знають, чого бажають користувачі, ніж експерти в області проектування і написання коду. Зрозуміло, буде краще, якщо дизайнер інтерфейсів створить інтерфейс, а розробник напише код, який реалізує логіку цього інтерфейсу, але технології типу Swing або MFC просто-напросто не дозволяють чинити таким чином.

MVVM зручно використовувати замість класичного MVC та йому подібних у тих випадках, коли на платформі, де ведеться розробка, присутнє «зв'язування даних».

В MVC/MVP зміни у користувацькому інтерфейсі не впливають безпосередньо на модель, а йдуть через Контролер/Presenter. У таких технологіях, як WPF та Silverlight, присутня концепція «зв'язування даних», що дозволяє зв'язувати дані із візуальними елементами в обидві сторони.

Архітектура MVVM вирішує цю проблему ясним поділом відповідальності:

* Розробка користувацького інтерфейсу здійснюється дизайнером інтерфейсів за допомогою технології, більш-менш природної для такої роботи.
* Логіка користувацького інтерфейсу реалізується розробником як компонент ViewModel.
* Функціональні зв'язки між користувацьким інтерфейсом та ViewModel реалізуються через біндинги (bindings), які, по суті, є правилами типу «якщо кнопка A була натиснута, повинен бути викликаний метод onButtonAClick() з ViewModel». Біндинги можуть бути написані в коді або визначені декларативним шляхом.

Веб-сервіс складається із декількох піддодатків, таких як:

* front end: та частина сайту, яку бачать користувачі;
* back end: адміністративна частина сайту, яка дозволяє керувати додатком. Доступ до неї зазвичай обмежений;
* консоль: додаток, який складається із набору консольних команд та запускається у вікні терміналу або як завдання cron-у;
* API: надає стороннім додаткам інтерфейси для інтеграції з вашим додатком.

Шаблон MVVM ділиться на три частини:

1. Модель (Model), як і в класичному шаблоні MVC, Модель являє собою фундаментальні дані, що необхідні для роботи застосунку.
2. Вид/(Вигляд) (View) як і в класичному шаблоні MVC, Вигляд — це графічний інтерфейс, тобто вікно, кнопки тощо.
3. Модель вигляду (ViewModel, що означає «Model of View») з одного боку є абстракцією Вигляду, а з іншого надає обгортку даних з Моделі, які мають зв'язуватись. Тобто вона містить Модель, яка перетворена до Вигляду, а також містить у собі команди, якими може скористатися Вигляд для впливу на Модель. Фактично ViewModel призначена для того, щоб:

* Здійснювати зв'язок між моделлю та вікном.
* Відслідковувати зміни в даних, що зроблені користувачем.
* Відпрацьовувати логіку роботи View (механізм команд).

Таким чином, можна зробити висновок, що даний програмний продукт використовує архітектурні шаблони, що забезпечує технологічні можливості для побудови гнучкого додатку із дотриманням принципів ООП. Такий підхід надає можливість на успішну реалізацію проекту та подальшу його модифікацію та розвиток.

## Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Для створення інтерактивного web-сервісу по пошуку домашніх тварин було обрано найпоширеніші інструментальні засоби такі як: мову розмітки гіпертекстових документів HTML5, каскадні таблиці стилів CSS3, браузерну мову програмування JavaScript, прогресивний фреймворк для створення користувацьких інтерфейсів Vue, веб-сервер Apache, графічний редактор Adobe Photoshop CC 2017 та середовище розробки Visual Studio Code 1.28.2.

***HTML*** ***(HyperText Markup Language, мова розмітки гіпертексту)*** − це стандарт верстки веб-сторінок, який визначає, які елементи і як повинні розташовуватися в документі.

Технологія HTML5 дозволяє не тільки спростити розмітку сторінок, зменшивши їх розмір, а й істотно полегшити завдання для творців макетів сторінок сайтів. У HTML5 введено безліч нових можливостей, які в розмітці коду сторінок можуть використовувати дизайнери, наприклад, нові елементи форм, мультимедіа, API та інше.

***CSS*** ***(Cascading Style Sheets, каскадні таблиці стилів)*** – формальна мова опису зовнішнього вигляду документа, написаного з використанням мови розмітки.

Переважно використовується як засіб опису, оформлення зовнішнього вигляду веб-сторінок, написаних за допомогою мов розмітки HTML і XHTML, але може також застосовуватися до будь-яких XML-документах, наприклад, до SVG або XUL.

CSS використовується творцями веб-сторінок для задання кольорів, шрифтів, розташування окремих блоків і інших аспектів представлення зовнішнього вигляду цих веб-сторінок. Основною метою розробки CSS було розділення опису логічної структури веб-сторінки (яке проводиться за допомогою HTML або інших мов розмітки) від опису зовнішнього вигляду цієї веб-сторінки (яке тепер проводиться за допомогою формальної мови CSS). Такий поділ може збільшити доступність документа, надати велику гнучкість і можливість управляти його представленням, а також зменшити складність і повторюваність в структурному вмісті. Крім того, CSS дозволяє представляти один і той же документ в різних стилях або методах виведення, таких як екранне уявлення, друковане подання, читання голосом (спеціальним голосовим браузером або програмою читання з екрану), або при виведенні пристроями, що використовують шрифт Брайля.

***JavaScript*** − динамічна мова програмування. Підтримує об'єктно-орієнтовані, імперативні і функціональні стилі. Є реалізацією мови ECMAScript (стандарт ECMA-262).

JavaScript зазвичай використовується як вбудована мова для програмного доступу до об'єктів додатків. Найбільш широке застосування знаходить в браузерах як мова сценаріїв для додання інтерактивності веб-сторінок.

Основні архітектурні риси: динамічна типізація, слабка типізація, автоматичне керування пам'яттю, прототипне програмування, функції як об'єкти першого класу.

***Vue*** - це прогресивний фреймворк для створення користувацьких інтерфейсів. На відміну від фреймворків-монолітів, Vue створений придатним для поступового впровадження. Його ядро в першу чергу вирішує завдання рівня уявлення (view), що спрощує інтеграцію з іншими бібліотеками та існуючими проектами. З іншого боку, Vue повністю підходить і для створення складних односторінкових додатків (SPA, Single-Page Applications), якщо використовувати його спільно з сучасними інструментами та додатковими бібліотеками.

Існує багато прогресивних фреймворків які підходять для реалізації такого веб-сервісу, серед них найпопулярніші це Angular 2, React та Vue. Щоб визначитись який з них краще, потрібно ї порівняти.

Таблиця 1.4.1

Порівняння прогресивних фреймворків

|  |  |  |  |
| --- | --- | --- | --- |
| Пункти вирішення | Angular 2 | React | Vue |
| Стабільність | Так | Так | Так |
| Підтримка сильною спільнотою | Так, за ним стоїть величезна спільнота та компанія Google. | Так, величезна спільнота та Facebook підтримує його. | Не такака величезна, але досить велика, і його підтримують Laravle та Alibaba. |
| Хороша документація | Так | Так | Так |

Продовження таблиці 1.4.1

|  |  |  |  |
| --- | --- | --- | --- |
| Легко вивчити | Ні, потрібно знати Typescript | Частково | Так |
| Інтеграція з Bootstrap | Так | Так | Так |
| Розмір | 566К | 139К | 58.8К |
| Можливість повторно використовувати код | Так | Ні, тільки CSS | Так, HTML та CSS |
| Швидкість програмування | Повільно | Нормально | Швидко |
| Реактивність | Частково | Так | Так |
| Компонента основа | Так | Так | Так |
| CLI | Так | Так | Так |

Проаналізувавши цю таблицю можна зробити висновок, що фреймворки хороші але один з них являється фаворитом і це Vue. Він швидкий у вивчені, на ньому дуже швидко програмувати, він займає дуже мало розміру, що дуже важливо на даний момент, так як більшість користувачів інтернету використовують свій смартфон для доступу до інтернету. В наш час не дуже швидкий швидкий мобільний інтернет, тому мала розмірність є дуже важливою перевагою.

***Vuex*** - патерн управління станом + бібліотека для додатків на Vue.js. Він служить централізованим сховищем даних для всіх компонентів програми з правилами, що гарантують, що стан може бути змінено тільки передбачуваним чином. Vuex інтегрується з офіційним розширенням vue-devtools, надаючи «з коробки» такі передові можливості, як «машину часу» для налагодження і експорт/імпорт зліпків стану даних.

У центрі будь-якого Vuex-додатка знаходиться сховище. «Сховище» - це контейнер, в якому зберігається стан вашого застосування. Два моменти відрізняють сховище Vuex від простого глобального об'єкта:

* Сховище Vuex реактивне. Коли компоненти Vue покладаються на його стан, то вони будуть реактивні і ефективно оновлюватися, якщо стан сховища змінюється.
* Не можна безпосередньо змінювати стан сховища. Єдиний спосіб внести зміни - явно викликати мутацію. Це гарантує, що будь-яка зміна стану залишає слід і дозволяє використовувати інструментарій, щоб краще розуміти хід роботи програми.

***Apache*** – це найбільш розповсюджений веб-сервер, який являється безкоштовним та призначений для використання під управлінням більшості відомих операційних систем. Apahce у якості протоколу передачі даних використовує протокол ***HTTP***.

Порівняльний аналіз СКБД дозволяє раціонально вибрати систему керування базами даних для проекту. У якості альтернатив будуть розглянуті наступні СКБД: Firebase Realtime Database, MongoDB, MySQL. Всі бази безкоштовні, Firebase Realtime Database (частково).

Усі обрані системи керування базами даних підходять для проведення аналізу й порівняння тому що їх можна використовувати с фреймворком Vue.

Таблиця 1.4.2

Аналіз загальних показників СКБД

|  |  |  |  |
| --- | --- | --- | --- |
| Показники | Firebase Realtime Database | MongoDB | MySQL |
| Модель первинної бази даних | Сховище документів | Сховище документів | Реляційна |
| Вторинна модель баз даних | - | Сховище ключів | Сховище документів та ключів |
| Доступна тільки як хмарна служба | Так | Ні | Ні |
| Серверні операційні системи | Хостинг | Linux, OS X, Solaris, Windows | FreeBSD, Linux, OS X, Solaris, Windows |
| Схема данных | Без схеми | Без схеми | Так |
| Зумовлені типи даних | Так | Так | Так |
| Вторинні індекси | Так | Так | Так |
| Підтримка SQL | Ні | Запити тільки для читання SQL-запитів через коннектор MongoDB для BI | Так |
| API і інші методи доступу | Android, IOS, JavaScript API, RESTful HTTP API | Пропріетарний протокол з використанням JSON | Власний API, ADO.NET, JDBC, ODBC |
| Мови програмування | Java, JavaScript, Objective-C | C, C#, C++, Java, JavaScript, PHP, Lua, Python та ін. | C, C#, C++, Java, JavaScript, PHP, Perl, Python та ін. |

Продовження таблиці 1.4.2

|  |  |  |  |
| --- | --- | --- | --- |
| Збережені процедури | обмежена функціональність з використанням «правил» | JavaScript | Так |
| Тригери | Зворотні виклики запускаються при зміні даних | Ні | Так |
| Транзакції | Так | Багатодомені транзакції ACID з ізоляцією моментальних знімків | ACID |

Проаналізувавши табл.1.4.2 можна зробити висновок, що даного веб-сервісу підходить Firebase Realtime Database.

***База даних Firebase Realtime*** є хмарною базою даних. Дані зберігаються як JSON і синхронізуються в реальному часі з кожним підключеним користувачем. Коли створюється крос-платформені додатки з iOS, Android і JavaScript SDK, всі користувачі діляться одним екземпляром бази даних Realtime і автоматично отримують оновлення з новітніми даними.

Основні можливості:

* Режим реального часу. Замість звичайних HTTP-запитів база даних Firebase Realtime використовує синхронізацію даних - кожен раз, коли дані змінюються, будь підключений пристрій отримує це оновлення в мілісекундах.
* Не в мережі. Додатки Firebase залишаються чуйними навіть в автономному режимі, оскільки SDK бази даних Firebase Realtime зберігає ваші дані на диск. Після відновлення зв'язку клієнтський пристрій отримує будь-які припущення зміни, синхронізуючи їх з поточним станом сервера.
* Доступність. Доступ до бази даних Firebase Realtime можна отримати безпосередньо з мобільного пристрою або веб-браузера; немає необхідності в сервері додатків. Безпека і перевірка даних доступні через правила безпеки бази даних Firebase Realtime, правила на основі виразів, які виконуються при читанні або запису даних.

***Webpack*** - це модуль з відкритим вихідним кодом для JavaScript модулів. Його основна мета - об'єднати файли JavaScript для використання в браузері, але він також здатний перетворювати, пов'язувати або упаковувати практично будь-які ресурси або активи. Webpack приймає модулі з залежностями і генерує статичні активи, що представляють ці модулі. Це модульний модуль в основному для JavaScript, але він може перетворювати інтерфейсні активи, такі як HTML, CSS, навіть зображення, якщо включені відповідні плагіни.

Він приймає залежності та генерує графік залежностей, що дозволяє веб-розробникам використовувати модульний підхід для цілей розробки веб-додатків. Компоновщик може використовуватися з командного рядка або може бути налаштований з використанням файлу конфігурації, який називається webpack.config.js.

***Babel*** - безкоштовний компілятор JavaScript з відкритим вихідним кодом і настроюваним транспілятором, який використовується в веб-розробці. Babel дозволяє розробникам програмного забезпечення писати вихідний код на вашій мові програмування або мовою розмітки і переводити його на Babel на JavaScript, мову, зрозумілу сучасним веб-браузерів.

Babel являється популярним інструмент для використання новітніх функцій мови програмування JavaScript. Як транспілятор або вихідного джерела, розробники можуть програмувати нові мовні функції ECMAScript 6 (ES6), використовуючи Babel для перетворення вихідного коду в версії JavaScript, які можуть обробляти еволюціонуючи браузери.

## Висновки до розділу 1

Проведений аналіз предметної області дозволив визначити основні напрямки у розробці інтерактивного web-сервісу для пошуку домашніх тварин. Необхідно реалізувати збереження інформації про оголошення та користувачів. Розробити механізм локалізації та фільтрації за різними критеріями.

Огляд наявних аналогів показав, що основними функціями таких систем є: наявність модулів розміщення оголошень, модерація, фільтрація за різними критеріями, відображення місця знахідки/втрати на карті, локалізація, авторизація і реєстрація.

В роботі обґрунтовано вибір архітектури для реалізації системи:

* мову розмітки гіпертекстових документів HTML5;
* каскадні таблиці стилів CSS3;
* браузерну мову програмування JavaScript;
* фреймворк Vue;
* патерн управління станом Vuex;
* середовище керування базами даних Firebase Realtime;
* веб-сервер Apache 2.4;
* статичний модульний збиральник Webpack 3.6.0;
* транспілятор кода Babel 6.2.1.

Використано такі програмні продукти та інтегровані середовища розробки: графічний редактор Adobe Photoshop CC 2017; середовище розробки Visual Studio Code 1.28.2.

[РОЗДІЛ 2. МОДЕЛЮВАННЯ](file:///C:\Users\Anna\AppData\Roaming\Skype\бакалаврат\fragment_bakalavr.docx#_Toc212894718) ІНТЕРАКТИВНОГО WEB-СЕРВІСУ ДЛЯ ПОШУКУ ДОМАШНІХ ТВАРИН

## [2.1. Принцип роботи Virtual DOM](file:///C:\Users\Anna\AppData\Roaming\Skype\бакалаврат\fragment_bakalavr.docx#_Toc212894719)

DOM (абревіатура від Document Object Model) - специфікація прикладного програмного інтерфейсу для роботи зі структурованими документами (як правило, документи HTML, XML). Визначається ця специфікація консорціумом W3C.

Веб-браузери обробляють складові DOM, і ми можемо взаємодіяти з ними, використовуючи JavaScript і CSS. Ми можемо працювати з вузлами документа, змінювати їхні дані, видаляти і вставляти нові вузли. В наші дні DOM API є практично кросплатформним і кросбраузерним.

Головна проблема DOM - він ніколи не був розрахований для створення динамічного призначеного для користувача інтерфейсу (UI). Ми можемо працювати з ним, використовуючи JavaScript і бібліотеки на зразок jQuery, але їх використання не вирішує проблем з продуктивністю.

Для прикладу, спробуйте перемістити 1000 div-блоків на 10 пікселів вправо.

Це може зайняти більше секунди - це занадто багато для сучасного інтернету. Ви можете оптимізувати скрипт і використовувати деякі прийоми, але в підсумку це викличе лише головний біль при роботі з величезними сторінками і динамічним UI.

В даний час W3C працює над новим стандартом Shadow DOM.

Shadow DOM - це робоча чернетка стандарту W3C. Специфікація, що описує метод об'єднання декількох DOM-дерев в одну ієрархію і як ці дерева взаємодіють один з одним в межах документа, що дозволяє краще скомпонувати DOM.

Virtual DOM не є стандартом і в кінцевому підсумку ми як і раніше взаємодіємо з DOM, але робимо це якомога рідше і більш ефективно. Замість того, щоб взаємодіяти з DOM безпосередньо, ми працюємо з його копією. Ми можемо вносити зміни в копію, виходячи з наших потреб, а після цього застосовувати зміни до реального DOM.

При цьому відбувається порівняння DOM-дерева з його віртуальної копією, визначається різниця і запускається перерисовка того, що було змінено.

Такий підхід працює швидше, бо не включає в себе всі великі частини реального DOM.

Але тільки якщо ми робимо це правильно. Є дві проблеми: коли саме робити повторну перерисовку DOM і як це зробити ефективно.

Перерисовку DOM потрібно робити тоді, коли дані змінились і потребують оновлення. Є два варіанти дізнатися, що дані змінилися:

* Перший з них - «dirty checking» (брудна перевірка) полягає в тому, щоб опитувати дані через регулярні проміжки часу і рекурсивно перевіряти всі значення в структурі даних.
* Другий варіант - «observable» (спостережувач) полягає в спостереженні за зміною стану. Якщо нічого не змінилося, ми нічого не робимо. Якщо змінилося, ми точно знаємо, що потрібно оновити.

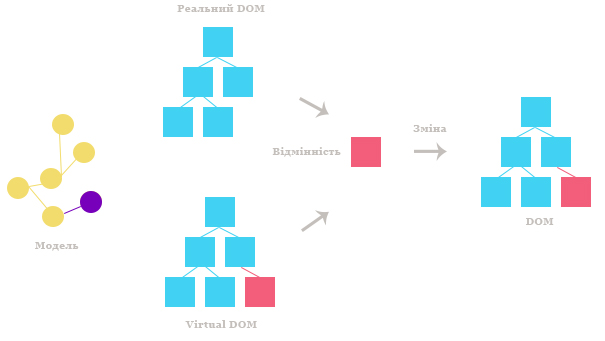


Рис.2.1.1. Принцип роботи Virtual DOM

Що робить цей підхід дійсно швидким:

* Ефективні алгоритми порівняння;
* Угруповання операцій читання/запису при роботі з DOM;
* Ефективне оновлення тільки під-дерев.

Virtual DOM - це техніка і набір бібліотек/алгоритмів, які дозволяють нам поліпшити продуктивність на клієнтській стороні, уникаючи прямої роботи з DOM шляхом роботи з легким JavaScript-об'єктом, що імітує DOM-дерево.

## [2.2. Реактивність у Vue](file:///C:\Users\Anna\AppData\Roaming\Skype\бакалаврат\fragment_bakalavr.docx#_Toc212894719)

Реактивне програмування — це парадигма програмування, побудована на потоках даних і розповсюдженні змін. Це означає, що у мовах програмування має бути можливість легко виразити статичні чи динамічні потоки даних, а реалізована модель виконання буде автоматично розсилати зміни через потік даних.

Коли простий JavaScript-об'єкт передається в екземпляр Vue в якості опції data, Vue обходить всі його поля і перетворює їх в пари геттер/сетер, використовуючи Object.defineProperty. Ця можливість з'явилася в JavaScript тільки починаючи з версії ES5, і в більш ранніх версіях її емулювати не вийде - з цієї причини Vue і не підтримує IE8 і нижче.

Геттери і сеттери невидимі користувачеві, але саме вони є тим внутрішнім механізмом, який дозволяє Vue відстежувати залежності і зміни даних.

До кожного екземпляра компонента приставлений пов'язаний з ним екземпляр спостерігача, який позначає всі поля, порушені при відображенні компонента, як залежності. Надалі, коли викликається сетер поля, позначеного як залежність, цей сетер повідомляє спостерігача, який, в свою чергу, ініціює повторну відрисовку компонента.

В силу обмежень сучасного JavaScript (і відмови від Object.observe), Vue не може відстежити додавання або видалення властивості об'єкта. Щоб поле стало реактивним, Vue перетворює його в пару геттер/сетер в ході ініціалізації екземпляра. Тому всі поля повинні спочатку бути задані в об'єкті data.

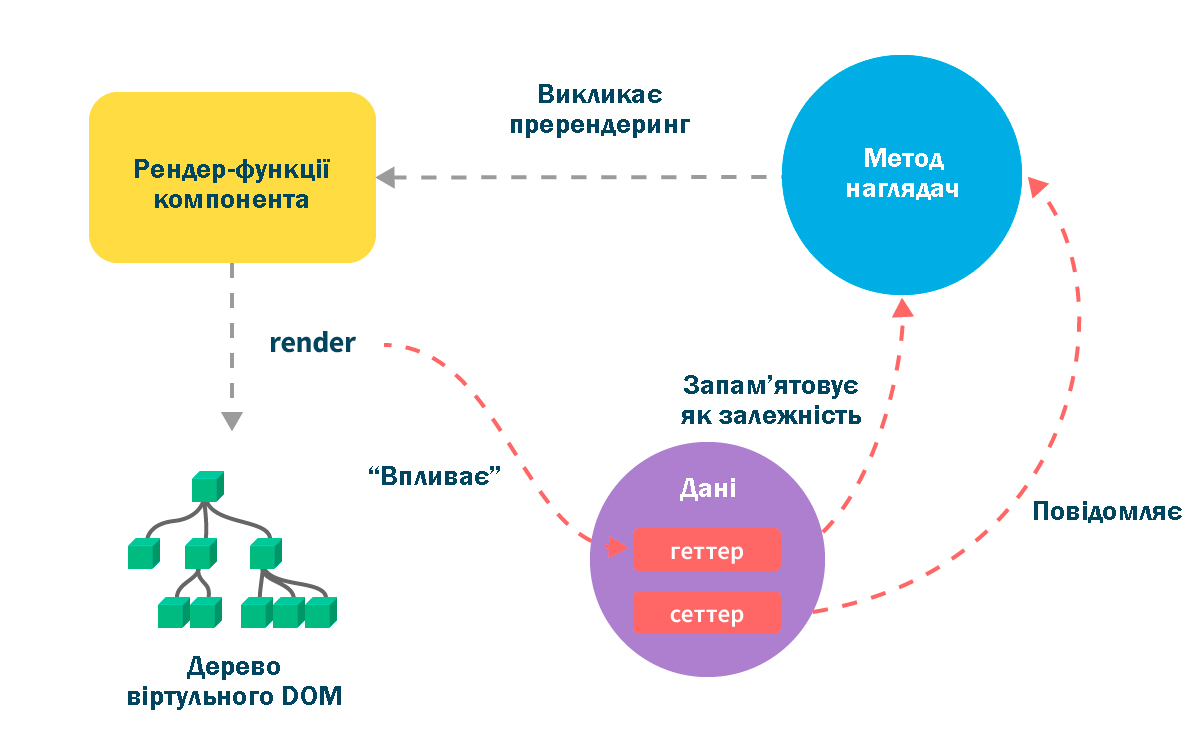


Рис.2.2.1. Принцип роботи реактивності у Vue

У Vue не можна динамічно додавати нові кореневі реактивні властивості в уже існуючий екземпляр. Проте, можна додати реактивні властивості у вкладені об'єкти, використовуючи метод Vue.set(object, key, value).

Також можна використовувати метод примірника vm.$set, який представляє собою псевдонім до глобального Vue.set.

Іноді потрібно додати кілька властивостей в існуючий об'єкт, наприклад, за допомогою Object.assign() або \_.extend(). Якщо так вчинити, додані властивості не стануть реактивними. Для вирішення цього завдання доведеться створити новий об'єкт, що містить поля як оригінальний об'єкт, так і об'єкт-домішки.

## [2.3. Розробка алгоритму створення оголошень с типом “Побачено”](file:///C:\Users\Anna\AppData\Roaming\Skype\бакалаврат\fragment_bakalavr.docx#_Toc212894719)

Особливість оголошень даного типу в тому, що при створенні оголошення ведеться пошук подібного оголошення і якщо є збіги то малюється стрілка с того оголошення до ново створеного. Ця стрілка буде давати користувачам зрозуміти яка рухається їхня домашня тварина і в якому місці потрібно її шукати.

При створенні оголошення, ми за допомогою метода getAdvertsInRadius(), відбираємо оголошення в радіусі 5 км.

getAdvertsInRadius(newAdvert) {

const radius = 5;

return this.adverts.filter(advert =>

utils.checkIncludesMarkerInRadius(

this.google,

newAdvert.position,

advert.position,

radius,

),

);

}

Після чого за допомогою методів compareAdvertsColors() та getSuitableAdverts() шукаємо оголошення які співпадають з новоствореним оголошенням.

compareAdvertsColors(newAdvert, advert) {

return newAdvert.petColor.every(

petColor => advert.id\_pet\_color.indexOf(petColor) >= 0,

);

},

getSuitableAdverts(newAdvert) {

return this.getAdvertsInRadius(newAdvert).filter(

advert =>

advert.id\_pet\_type === newAdvert.petType &&

advert.id\_pet\_breed === newAdvert.petBreed &&

advert.id\_pet\_coloring === newAdvert.petColoring &&

advert.id\_pet\_age === newAdvert.petAge &&

this.compareAdvertsColors(newAdvert, advert),

);

}

За допомогою методу checkSawAdvert(), ми відбираємо оголошення тільки с типом “Побачено”.

checkSawAdvert(newAdvert) {

if (newAdvert.typeMarker === 'saw') {

return this.getSuitableAdverts(newAdvert);

}

return false;

}

Метод getPreviousPosition(), повертає позицію знайденого оголошення, або повертає null, якщо такого оголошення не знайшлось. Якщо ж знайшлось декілька подібних оголошень, то повертається позиція оголошення яке створено недавно.

getPreviousPosition(newAdvert) {

const resultCheckSawAdvert = this.checkSawAdvert(newAdvert);

if (resultCheckSawAdvert && resultCheckSawAdvert.length > 0) {

if (resultCheckSawAdvert.length > 1) {

resultCheckSawAdvert.sort(utils.compareDateCreate);

}

return resultCheckSawAdvert[0].position;

}

return null;

}

Тепер в компоненті AdvertMarker при відображенні маркера ми перевіряємо поле previousPosition, якщо там є координати, то використовуючи Google Maps API малюємо стрілку.

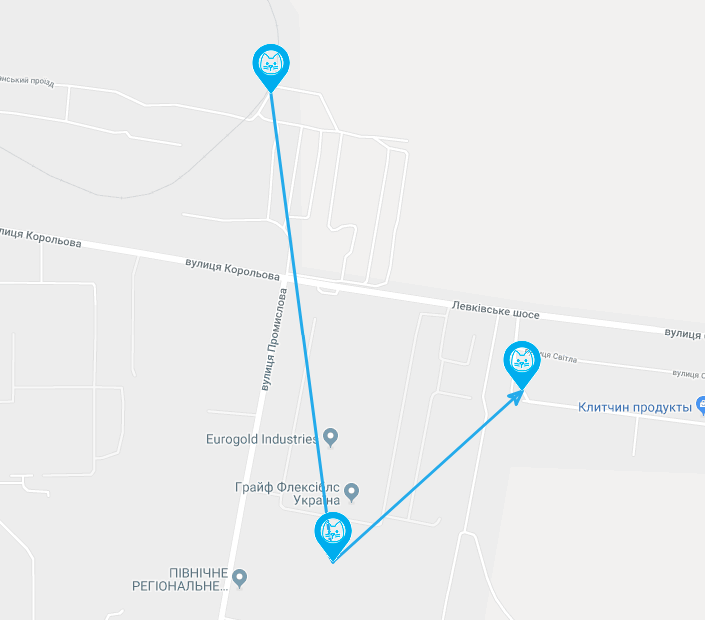


Рис.2.3.1. Приклад відображення оголошення с типом “Побачено”

## Висновки до розділу 2

В даному розділі було описано принцип робото Virtual DOM, з якого можна було зрозуміти, що це техніка і набір бібліотек/алгоритмів, які дозволяють нам поліпшити продуктивність на клієнтській стороні, уникаючи прямої роботи з DOM шляхом роботи з легким JavaScript-об'єктом, що імітує DOM-дерево.

Також було описано одну із особливостей фреймворка Vue, реактивність. Тобто за нашими даними слідкує наглядач і при змінні даних він реагує на них і перерендерить віртуальне DOM дерево, якщо ці дані впливають на нього.

Був докладно описаний алгоритм створення оголошення с типом “Побачено”. Де для такого оголошення виконується пошук схожих оголошень і малюється стрілка від знайденого до новоствореного. Ця стрілка допомагає користувач зрозуміти як рухалась їх домашня тварина і в якому місці потрібно робити пошук.

[РОЗДІЛ 3. ПРОЕКТУВАННЯ](file:///C:\Users\Anna\AppData\Roaming\Skype\бакалаврат\fragment_bakalavr.docx#_Toc212894718) ІНТЕРАКТИВНОГО WEB-СЕРВІСУ ДЛЯ ПОШУКУ ДОМАШНІХ ТВАРИН

## [3.1. Визначення](file:///C:\Users\Anna\AppData\Roaming\Skype\бакалаврат\fragment_bakalavr.docx#_Toc212894719) варіантів використання та об’єктно-орієнтованої структури системи

Автоматизований веб-сервіс для пошуку домашніх тварин має основні цілі: проект створюється з метою пошуку втрачених домашніх тварин або знаходженні господаря тварини.

**Вимоги користувачів**

Зовнішні користувачі мають доступ до наступних функцій:

1. Перегляд основної інформації про сайт, його основні сторінки.

2. Реєстрація на сайті.

3. Аутентифікація на сайті.

4. Створення, редагування та видалення об’яв.

5. Фільтрація об’яв за різними критеріями.

6. Перегляд огошень на карті.

7. Локалізація трьома мовами: англійська, російська та українська.

Внутрішні користувачі – адміністратори сервісу мають доступ до додаткових функцій, таких як:

1. Редагування БД.

2. Редагування дизайна сервісу.

3. Модерація оголошень.

4. Перегляд статистики сервісу.

5. Монетизація сервісу.

З діаграми прецендентів яка зображена на рис. А.1 ми бачимо, що для звичайного користувача даний програмний продукт буде доступний не повністю. Йому буде доступно лише перегляд оголошень та фільтрація. Для того щоб він був доступний повністю користувачу потрібно зареєструватись.

Після реєстрація він стає зареєстрованим користувачем і вже зможе створювати, редагувати та видаляти свої оголошення про знайдену/загублену домашню тварину. Також якщо у зареєстрованого користувача щось не виходить він може звернутися за допомогою до модератора, який в свою чергу відповість йому на поставлені запитання.

В даному програмному продукті є модератор який перевіряє добавлені оголошення користувачів. Він може прийняти або відхилити об’яви користувачів які не є коректні.

Також є адміністратор який слідкує щоб програмний продукт працював швидко та стабільно. Він планує стратегію по просуванню веб-сервісу в мережі інтернет. Аналізує відвідуваність, залучення відвідуваності, конверсію. Розробляє, підтримує концепцію веб-сервісу, покращує його роботу. Удосконалює та виправляє поточні структури веб-сервісу. При появі помилок він швидко виправляв їх, щоб вони не впливали на процес роботи користувачів з програмним продуктом.

***Функціональні вимоги:***

1. Авторизація користувачів в системі: У веб-сервісі для пошуку домашніх тварин має бути можливість реєстрації та авторизації користувача, при повторному відвідуванню сайту та присвоєння йому відповідної ролі (зареєстрований користувач, модератор, адміністратор).

2. Збереження нових даних введених користувачами: Система повинна зберігати нові оголошення при створені, редагуванні та при редагуванні інформації про користувача. Також надавати можливість користувачу шукати та переглядати об’яви.

3. Модерація: Модератор має можливість переглянути, схвалити або відхилити оголошення.

4. Реалізація фільтрації за різними критеріями оголошень на сайті.

5. Валідація введених даних.

***Нефункціональні вимоги:***

1. Навчання роботи з системою:

* час, потрібний для навчання з інструментами роботи з інформаційною системою для звичайних користувачів – 10-20 хвилин, а для досвідчених – 10 хвилин;
* час відповіді системи для звичайних запитів не повен перевищувати 2 секунди, а для більш складних запитів — 3 секунд;
* інтерфейс представлення програми повинен бути інтуїтивно зручним для користувача;
* надійність;
* доступність – час, потрібний для обслуговування системи не повинен перевищувати 2% від загального часу роботи;
* середній час безперервної роботи – 365 робочих днів;
* максимальна норма помилок та дефектів в роботі системи – 1 помилка на 1000 запитів користувача.

1. Продуктивність:

* швидке отримання результатів при здійсненні функцій програми;
* cистема повинна підтримувати мінімум 100 одночасно працюючих користувачів, пов’язаних з спільною базою даних.

1. Можливість експлуатації:

* даний програмний продукт повинен збільшувати споживання ресурсів системи при збільшенні навантаження на нього.
* оновлення версій компонентів програми повинно автоматично встановлюватися на клієнтську частину даного програмного продукту.

Для реалізації проекту була вибрана технологія MVVM, та браузерна мова програмування JavaScript з використанням фреймворку Vue.

Аналіз функціональних вимог дозволив виділити наступні сутності, що забезпечать реалізацію програмного комплексу системи. На рис. Б.1 наведено діаграму компонентів та класів. В розділі 3.2. буде наведена модель бази даних системи у відповідності до якої організований рівень моделі.

Для даного програмного продукту було реалізовано 13 компонентів та 3 класи.

Компонент Navigation – відноситься до навігації по сайту, він має наступні методи:

* onLogout(), викликається коли користувач припиняє сеанс роботи в якості зареєстрованого користувача;
* addModerationTab(navigationMenu), викликається щоб добавити новий пункт меню “Модерація” для модератора.

Компонент Signup – призначений для реєстріції користувачів, він має наступні методи:

* onSignup(), викликається для створення нового користувача;
* onDismissed(), викликається щоб закрити помилку, яка з’явилась при реєстрації користувача.

Компонент Signin – призначений для авторизації користувач, він має наступні методи:

* onSignin(), викликається для авторизації користувача;
* onDismissed(), викликається щоб закрити помилку, яка з’явилась при авторизації користувача.

Компонент Home – призначений для відображення головної сторінки, він має наступні методи:

* setLatLng(event), потрібен для відображення спливаючого вікна для створення оголошення;
* setNewCenter(), потрібен для задання нових координат для центру карти;
* getDistanceBetween(markerPosition), потрібен для розрахунку дистанції між маркером на карті і центром карти;
* getCurrentLocation(), потрібен для визначення поточного місця геолокації;
* setNewCenterPosition(position), потрібен для задання початкового центру на карті;
* getAdverts(), потрібен для отримання всіх оголошень, які підійшли під фільтр;
* filterRadius(adverts), фільтрує оголошення за введеним радіусом.

Класс Store – призначений для зберігання стану програми, також він служить центральним сховищем даних для всі компонентів програми з правилами, що гарантують, що стан може бути змінено тільки передбачуваним чином, він має наступні методи:

* loadPetTypes(), потрібен для завантаження видів тварин з БД;
* loadPetBreeds(), потрібен для завантаження порід тварин з БД;
* loadPetColors(), потрібен для завантаження кольорів тварин з БД;
* loadPetColorings(), потрібен для завантаження забарвлень тварин з БД;
* loadAdverts(), призначений для завантаження всіх оголошень з БД;
* createAdvert(advertData), викликається для створення оголошення;
* updateAdvert(payload), викликається для оновлення даних про оголошення;
* deleteAdvert(id), викликається для видалення оголошення з відповідним id;
* updateInfoWindow(payload), викликається для оновлення даних в спливаючому вікні з інформацією про оголошення;
* editAdvert(payload), викликається для зміни даних в спливаючому вікні при оновлені даних про оголошення;
* updateAdvertFilter(payload), викликається для оновлення даних фільтра;
* autoSign(payload), викликається для автоматичної авторизації користувача;
* signUserUp(payload), викликається для реєстрації користувача;
* signUserIn(payload), викликається для авторизації користувача;
* loadUserData(userId), потрібен для завантаження інформації про користувача з БД;
* updateLocale(payload), викликається для оновлення локалізації в програмі;
* logout(), викликається для припинення сеансу роботи в якості зареєстрованого користувача.

Компонент AdvertsList – призначений для відображення списку оголошень, він має наступні методи:

* onClosePopup(), викликається для закриття спливаючого вікна редагування оголошення;
* onOpenEditPopup(advert), викликається для відкриття спливаючого вікна редагування оголошення;
* onDeleteAdvert(id), викликається для видалення оголошення з відповідним id;
* onApproveAdvert(id), викликається для схвалення оголошення модератором;
* onRejectAdvert(id), викликається для відхилення оголошення модератором.

Компонент AdvertPopup – призначений для відображення спливаючого вікна створення/редагування оголошення, він має наступні методи:

* createMarker(), викликається для створення оголошення;
* onUpdateAdvertInfo(), викликається для оновлення інформації про оголошення;
* hidePopup(), викликається для закриття спливаючого вікна;
* loadedPetBreeds(petType), викликається для завантаження інформації порід тварини певного типу;
* onChangePetType(value), викликається при змінні типу тварину;
* onCancelEdit(), викликається для відхилення змін в оголошенні.

Таким чином, дана система реалізує функціонал, що передбачає реєстрацію та авторизацію користувачів, створення оголошень. Також містить методи редагування та видалення цих даних та модерування оголошень модератором. Даний функціонал передбачає фільтрацію оголошень за різними критеріями.

## [3.2. Розробка бази](file:///C:\Users\Anna\AppData\Roaming\Skype\бакалаврат\fragment_bakalavr.docx#_Toc212894723) даних системи

Реалізована база даних Firebase Realtime складається з 7 документів, які містять у собі всі дані для роботи веб-сервісу.

Структура бази даних наведена на рис. В.1.

Документ «users» призначений для зберігання інформації про користувачів, він має наступні поля:

1. id – унікальний ідентифікаційний номер користувача;
2. email – електрона пошта;
3. password – пароль;
4. role – роль користувача.

Документ «adverts» призначений для зберігання інформації про оголошення, він має наступні поля:

1. id – унікальний ідентифікаційний номер оголошення;
2. id\_pet\_type – унікальний ідентифікаційний номер виду тварини;
3. id\_pet\_breed – унікальний ідентифікаційний номер породи тварини;
4. id\_pet\_color – унікальний ідентифікаційний номер кольору тварини;
5. id\_pet\_coloring - унікальний ідентифікаційний номер забарвлення тварини;
6. id\_user - унікальний ідентифікаційний номер користувача;
7. typeMarker – тип оголошення;
8. id\_pet\_age – унікальний ідентифікаційний номер віку тварини;
9. status – статус оголошення;
10. position – позиція оголошення на карті;
11. contactInfo – контактна інформація;
12. dateCreate – дата створення оголошення;
13. previousPosition – попередня позиція оголошення.

Документ «pet\_breeds» призначений для зберігання інформації про породи тварини, він має наступні поля:

1. id – унікальний ідентифікаційний номер породи тварини;
2. id\_pet\_type - унікальний ідентифікаційний номер виду тварини;
3. en – назва на англійській мові;
4. ru – назва на російській мові;
5. ua – назва на українській мові.

Документ «pet\_colorings» призначений для зберігання інформації про забарвлення тварини, він має наступні поля:

1. id – унікальний ідентифікаційний номер забарвлення тварини;
2. en – назва на англійській мові;
3. ru – назва на російській мові;
4. ua – назва на українській мові.

Документ «pet\_colors» призначений для зберігання інформації про кольори тварин, він має наступні поля:

1. id – унікальний ідентифікаційний номер кольору тварини;
2. en – назва на англійській мові;
3. ru – назва на російській мові;
4. ua – назва на українській мові.

Документ «pet\_types» призначений для зберігання інформації про види тварин, він має наступні поля:

1. id – унікальний ідентифікаційний номер виду тварини;
2. en – назва на англійській мові;
3. ru – назва на російській мові;
4. ua – назва на українській мові.

Документ «pet\_ages» призначений для зберігання інформації про вік тварин, він має наступні поля:

* 1. id – унікальний ідентифікаційний номер віку тварини;
  2. en – назва на англійській мові;
  3. ru – назва на російській мові;
  4. ua – назва на українській мові.

## 3.3. Проектування та реалізація алгоритмів роботи системи

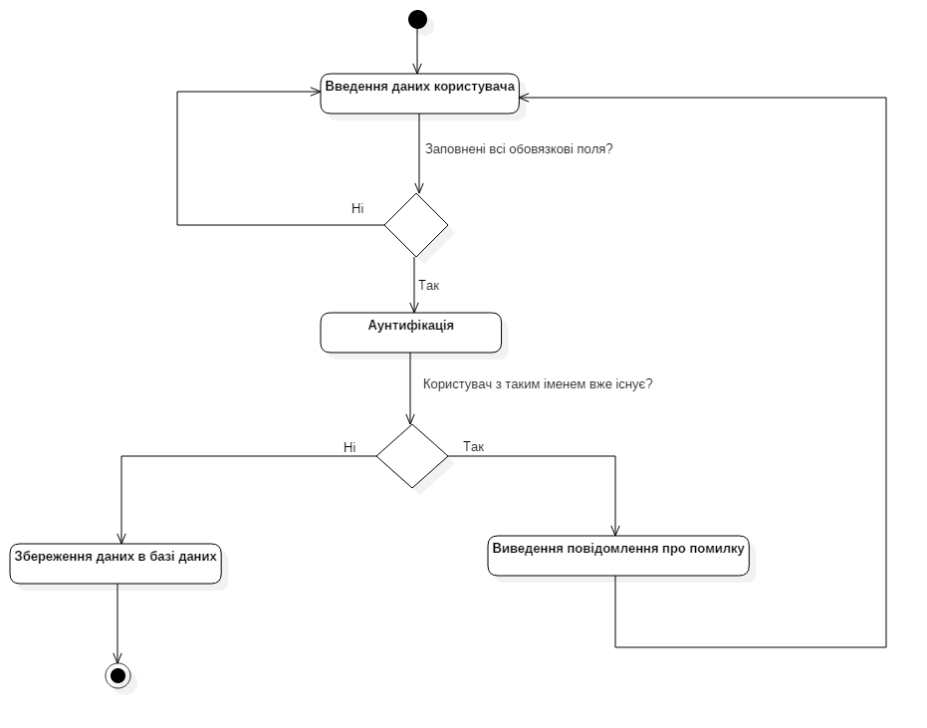


Рис.3.3.1. Діаграма активності “Реєстрація користувача”

Як показано на рис. 3.3.1 ми бачимо, після того як користувач перейшов на сторінку реєстрації йому потрібно буде ввести данні: електрону пошту, пароль, підтвердження паролю. Після того перевіряється чи заповнені обов’язкові поля і чи вони заповнені коректно, якщо ні то йому виводиться повідомлення які поля не були заповнені або були заповнені не коректно. Якщо ж всі поля заповнені відбувається аунтифікація – перевіряється чи користувач с такою електронною поштою зареєстрований, якщо є збіги то виводиться помилка. Якщо ж так, то користувач зберігається в базі даних і стає зареєстрованим користувачем.

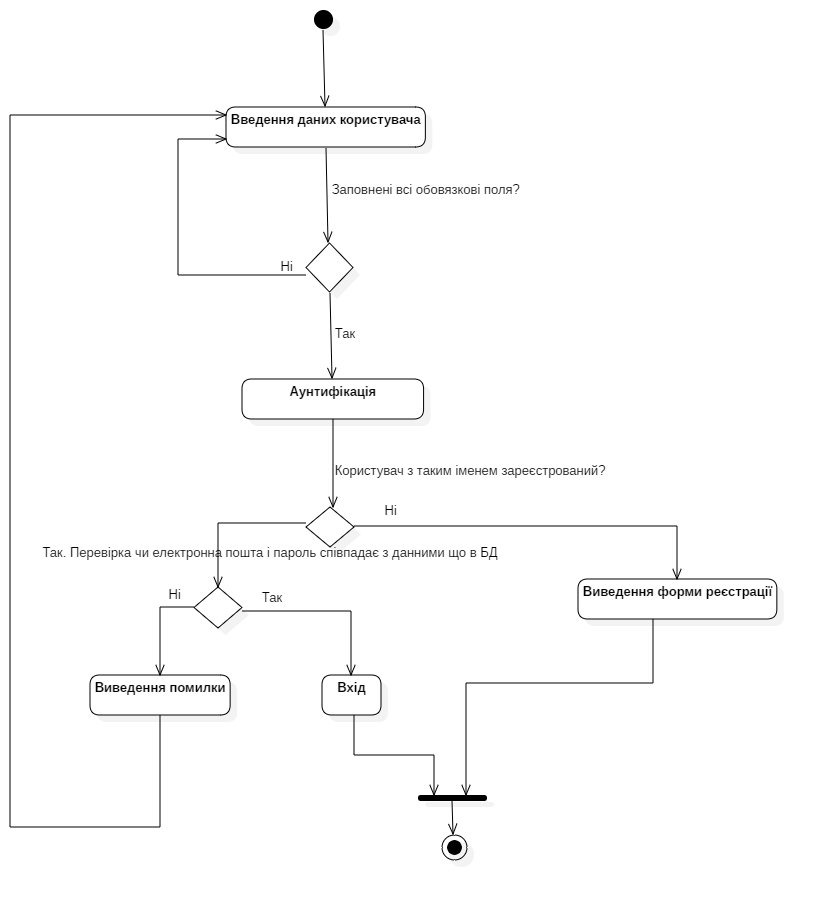


Рис.3.3.2. Діаграма активності “Авторизація користувача”

Щоб користувач перейшов на свій профіль йому потрібно авторизуватись. Для цього йому потрібно ввести такі дані: електронну пошту та пароль, після чого перевіряється чи ці поля заповнені. Якщо так, то відбувається аунтифікація, де перевіряється чи така електронна пошта існує, якщо ні то перенаправляє на сторінку реєстрації, якщо так то перевіряє чи електронна пошта співпадає з паролем. Якщо ні, то виводить помилку, якщо так, то авторизовує користувача на сайті.

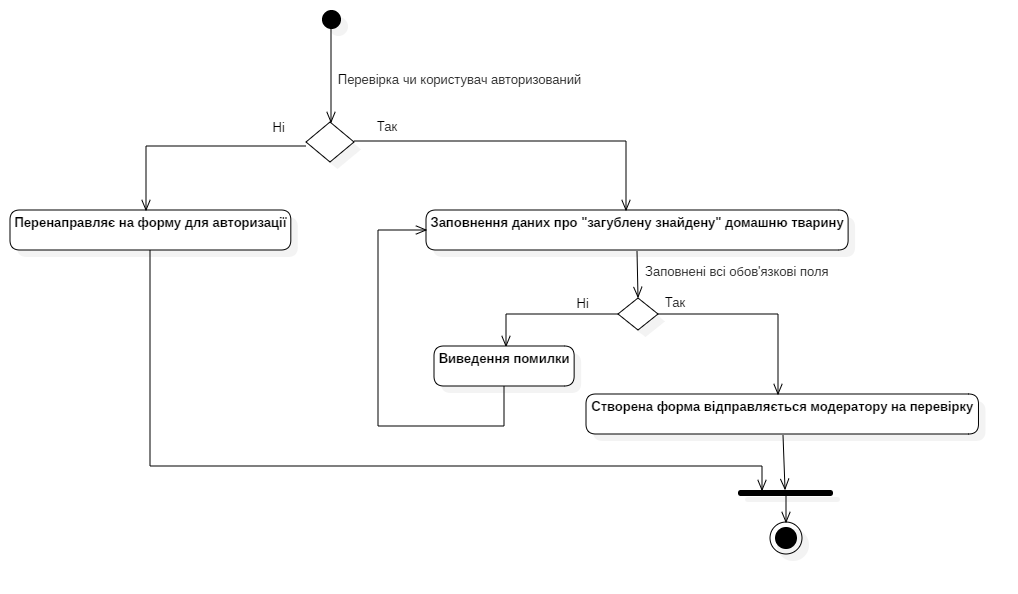


Рис.3.3.3. Діаграма активності “Додавання оголошення”

При додаванні оголошення про знайдену/загублену домашню тварину в першу чергу перевіряється чи користувач авторизований, якщо ні то його перенаправляє на сторінку, де він зможе зареєструватись. Якщо так, то він починає заповнювати поля. Після чого йде перевірка чи всі обов’язкові поля заповнені, якщо так, то створене оголошення відправляється модератору на перевірку.

Панель модерації доступна лише модераторам, інші користувачі її не бачать. При переходу на сторінку перевірки оголошень перевіряється чи оголошення існують, якщо ні, то виводиться повідомлення що об’яв немає, якщо ж об’яви є то виводить їх. Після чого модератор починає перевіряти їх на коректність, якщо ж об’ява коректна то він схвалює її і вона автоматично додається на сайт, якщо ні, то відхиляє і користувачу в профілі буде відображено, що ця об’ява не схвалена. Сам процес відображений на рис. 3.3.4.

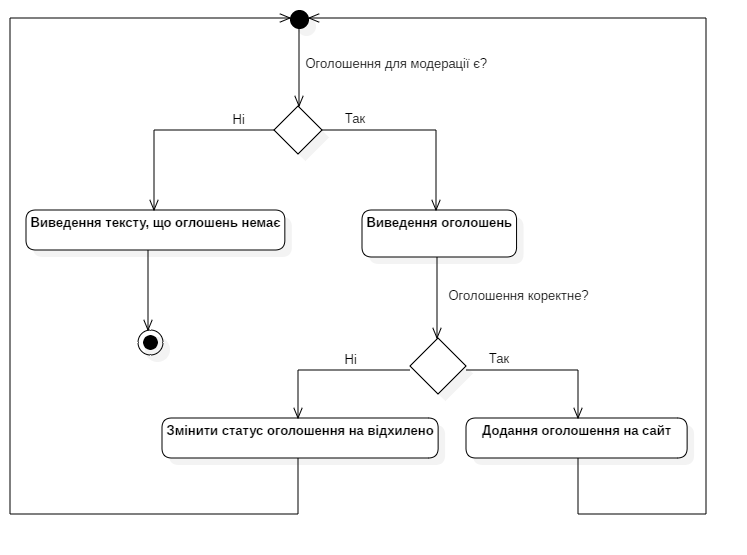


Рис.3.3.4. Діаграма активності “Модерація”

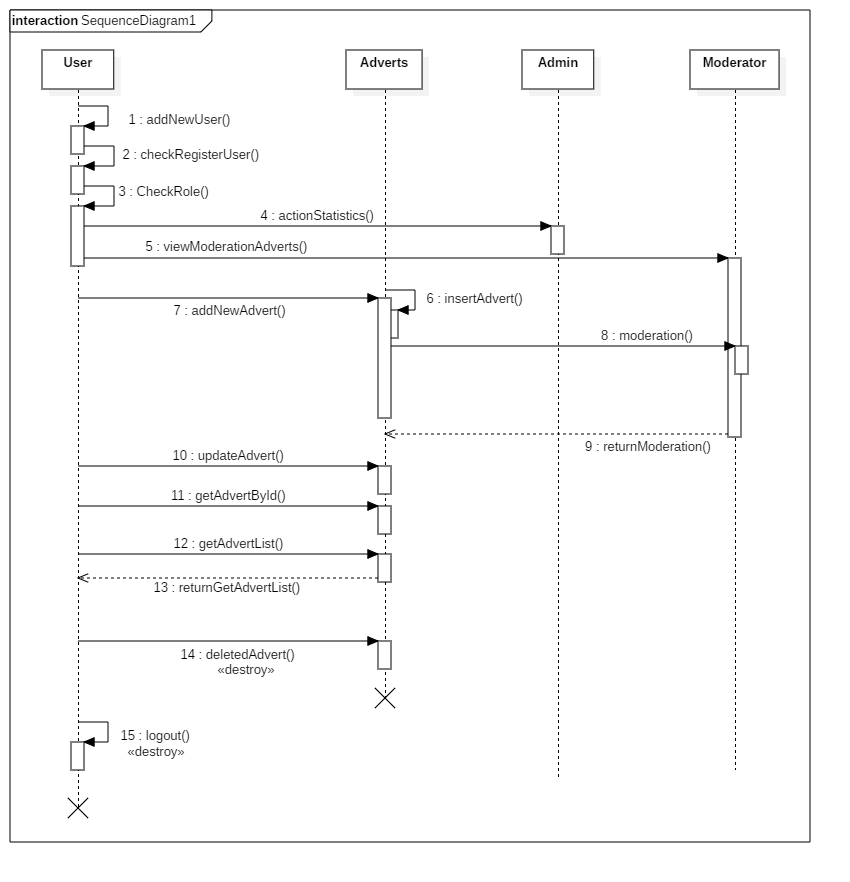


Рис. 3.3.5. Діаграма послідовності

З рис. 3.3.5 ми бачимо, спочатку створюється користувач, після чого перевіряється яка роль у користувача. В даному програмному продукті існує три ролі:

− Користувач;

− Модератор;

− Адміністратор.

Модератор має такий самий функціонал як користувач, але він ще має панель модерації завдяки якій він перевіряє створені оголошення користувачів.

Адміністратор має такий самий функціонал як і модератор але в добавок він має доступ до бази даних і файлів програмного продукту.

Після чого користувач добавляє оголошення, сама об’ява має три статуси знахідка, втрата та побачено. Він вибирає до якого типу належить об’ява і створює, сама об’ява відправляється на модерацію. Після перевірки модератора, об’ява або добавляється до інших записів, або змінює статус на відхилено. Сам користувач може переглянути об’яву, редагувати або видалити.

Отже нами було розглянуто діаграми та алгоритми основних процесів інтерактивного web-сервісу для пошуку домашніх тварин та проаналізовано взаємодію компонентів та класів системи під час процесів створення, редагування та фільтрація оголошення, а також реєстрації та авторизації на сайті.

## 3.4 Реалізація інтерактивного web-сервісу для пошуку домашніх тварин

Для реалізації системи було використано такі модулі:

1. vuetify – бібліотека готових компонентів, яка дає можливість створювати користувальницькі інтерфейси з використанням принципів Google Material Design;
2. vue-i18n – модуль для локалізації;
3. vuex – бібліотека для управлінням станом для додатків на Vue.js;
4. vue-router – бібліотека маршрутизації для Vue.js;
5. vue2-google-maps – бібліотека компонентів для взаємодії з Google Maps;
6. firebase – API для взаємодії з базою Firebase Realtime.

Всі взаємодії з БД відбуваються в Store (сховище для управління станом усього додатку), він реалізований за допомогою Vuex.

Для додавання нового оголошення в БД в компоненті AdvertPopup викликається метод createMarker(), який перевіряється валідність всіх полів та збирає дані в один великий об’єкт і відправляє його в Store, де викликається метод createAdvert().

createMarker() {

if (this.$refs.form.validate()) {

if (!this.image) {

this.photoUrlRules = false;

return;

}

const newMarker = {

typeMarker: this.typeMarker,

petType: this.petType,

petBreed: this.petBreed,

petAge: this.petAge,

petColor: this.petColor,

petColoring: this.petColoring,

contactInfo: this.contactInfo,

image: this.image,

position: this.latLng,

};

newMarker.previousPosition = this.getPreviousPosition(newMarker);

this.$store.dispatch('createAdvert', newMarker);

this.hidePopup();

}

}

createAdvert({ commit, getters, dispatch }, advertData) {

const advert = {

id\_user: getters.user.id,

typeMarker: advertData.typeMarker,

id\_pet\_type: advertData.petType,

id\_pet\_breed: advertData.petBreed,

id\_pet\_color: advertData.petColor,

id\_pet\_coloring: advertData.petColoring,

dateCreate: new Date().toISOString(),

status: 'moderation',

id\_pet\_age: advertData.petAge,

contactInfo: advertData.contactInfo,

position: {

lat: advertData.position.lat(),

lng: advertData.position.lng(),

},

previousPosition: advertData.previousPosition,

};

let key;

let photoUrl;

firebase

.database()

.ref('adverts')

.push(advert)

.then(data => data.key)

.then(generatedKey => {

key = generatedKey;

const filename = advertData.image.name;

const ext = filename.slice(filename.lastIndexOf('.'));

return firebase

.storage()

.ref(`adverts/${key}.${ext}`)

.put(advertData.image);

})

.then(fileData =>

firebase

.storage()

.ref(fileData.metadata.fullPath)

.getDownloadURL(),

)

.then(url => {

photoUrl = url;

return firebase

.database()

.ref('adverts')

.child(key)

.update({ photoUrl, id: key });

})

.then(() => {

commit('createAdvert', { ...advert, id: key, photoUrl });

})

.catch(err => {

// eslint-disable-next-line

console.log(err);

});

}

Фільтрація оголошень за різними критеріями, відбувається в компоненті Home в методі getAdverts(), де по властивості filterAdvert ми фільтруємо оголошення.

getAdverts() {

let result = this.adverts;

Object.keys(this.filterAdvert).forEach(key => {

if (this.filterAdvert[key] && this.filterAdvert[key] !== 'all') {

switch (key) {

case 'id\_pet\_color':

result = result.filter(

advert =>

this.filterAdvert[key].every(

color => advert[key].indexOf(color) >= 0,

)

? advert

: false,

);

break;

case 'radius':

result = this.filterRadius(result);

break;

default:

result = result.filter(

advert => advert[key] === this.filterAdvert[key],

);

break;

}

}

});

return result;

}

Для фільтрації оголошень за радіусом використовується метод filterRadius(adverts), а в ньому метод checkIncludesMarkerInRadius(gmapApi, positionA, positionB, radius).

filterRadius(adverts) {

return adverts.filter(advert =>

utils.checkIncludesMarkerInRadius(

this.google,

this.mapConfig.dynamicCenter,

advert.position,

this.filterAdvert.radius,

),

);

}

function checkIncludesMarkerInRadius(gmapApi, positionA, positionB, radius) {

return (

getDistanceBetween(gmapApi, positionA, positionB) / 1000 <= Number(radius)

);

}

function getDistanceBetween(gmapApi, positionA, positionB) {

return gmapApi.maps.geometry.spherical.computeDistanceBetween(

checkTypePositionMarker(positionA)

? positionA

: getNewPositionMarker(positionA),

checkTypePositionMarker(positionB)

? positionB

: getNewPositionMarker(positionB),

);

}

Модерація оголошень відбувається в компоненті Moderation, для дочірнього компонента AdvertList передається властивість isModeration = true та самі оголошення, які були отримані із Store. Оголошення із Store отримуються за допомогою метода moderationAdverts(state).

moderationAdverts(state) {

return state.adverts.filter(advert => advert.status === 'moderation');

}

## Висновки до розділу 3

Інтерактивний web-сервіс для пошуку домашніх тварин реалізує функціонал за допомогою 12 компонентів та 3 класів, що виконують основні функції сервісу, які передбачають реєстрацію та авторизацію користувачів, створення, редагування та видалення оголошення. Також містять модерацію, фільтрацію за різними критеріями та зміну локалізації.

В розділі реалізована база даних у відповідності до кращих практик для структури даних, вона забезпечує збереження, отримання даних в режимі реального часу та колективний доступ до web-сервісу. База даних складається з 7 документів, основними документами є: users та adverts.

Було розглянуто алгоритми основних процесів створення оголошень, фільтрація, модерація та проаналізовано взаємодію компонентів та класів системи під час їх процесів.

# ВИСНОВКИ

Під час виконання випускної роботи бакалавра було спроектовано та реалізовано web-сервіс для пошуку домашніх тварин, а також виконано наступні завдання:

1. Здійснено аналіз напрямків використання інформаційних технологій для автоматизації роботи програмного продукту. Проведений аналіз предметної області дозволив правильно поставити основну задачу веб-сервісу для пошуку домашніх тварин та необхідність реалізації засобів управління контентом сайту, написання функціональних модулів, створення, редагування, видалення, фільтрація оголошень.
2. Проаналізовано аналоги програмного продукту, такі як: «Всеукраинское виртуальное онлайн-бюро находок «Верни»», PETSI, LOSTPETS, громадська організація «LUCKFIND» та визначено, що основним рисами нової системи мають бути наявний веб-інтерфейс, модуль створення, редагування та видалення оголошень, модуль модерації, модуль локалізації, модуль фільтрації та реєстрації з наступною авторизацією.
3. Обрано архітектурні шаблони – клієнт-серверний та паттерн MVVM. Встановлено, що даний програмний продукт використовує архітектурні шаблони, які забезпечують технологічні можливості для побудови гнучкого додатку із дотриманням принципів ООП.
4. Для реалізації було вибрано наступні засоби: мову розмітки гіпертекстових документів HTML5; каскадні таблиці стилів CSS3; браузерну мову програмування JavaScript; фреймворк Vue; патерн управління станом Vuex; середовище керування базами даних Firebase Realtime; статичний модульний збиральник Webpack 3.6.0; транспілятор кода Babel 6.2.1; графічний редактор Adobe Photoshop CC 2017; середа розробки Visual Studio Code 1.28.2.
5. Описано принцип роботи Virtual DOM, який дав зрозуміти, що для розробки такого типу сервісів він являється дуже ефективним.
6. Розглянуто одну із особливостей фреймворка Vue, реактивність.
7. Докладний опис алгоритму створення оголошень с типом “Побачено”. Даний функціонал являється дуже ефективним для користувачів сервісу із-за відображення маршруту переміщення їхньої домашньої тварини на карті.
8. Визначено функціонал, який передбачає реєстрацію та авторизацію користувачів, створення, редагування та видалення оголошень. Веб-сервіс містить методи модерації та фільтрації оголошень.
9. Створено базу даних у відповідності до правил структури даних. База даних складається з 7 документів.

Із-за своїх особливостей, даний веб-сервіс являється кращим серед своїх аналогів. В подальшому даний веб-сервіс можна покращити шляхом оптимізації та додавання нових модулів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

**Книги:**

1. Петюшкин А. В. «HTML. Экспресс-курс», – СПб.: БХВ – Петербург, 2003,– 256 с.
2. Шеллі Пауерс, Вивчаючи Javascript, М.: МетаТехнология, 2006, 240 с.
3. С. Стефанов, Шаблони Javascript. СПб.: БХВ, Петербург, 2013, 124 с.
4. Д. Герман, Ефективний Javascript. М.: БИНОМ, 2016, 236 с.
5. М. Фаулер, К. Скотт. UML в кратком изложении. Применение стандартного языка объектного моделирования. М: «Мир», 1999, 191 с.
6. М. Гленфорд, Т. Баджетт, С. Кори, Искусство тестирования программ, 3-е изд. Москва: Дилектика, 2012, 345 с.
7. Мартин Фаулер. Шаблоны корпоративных приложений (Signature Series) = Patterns of Enterprise Application Architecture (Addison-Wesley Signature Series). — М.: «Вильямс», 2012. — 544 с.
8. Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес. Приемы объектно-ориентированного проектирования. Паттерны проектирования = Design Patterns: Elements of Reusable Object-Oriented Software. — СПб: «Питер», 2007. — С. 366.
9. Стив Макконнелл. Совершенный код = Code complete. — СПб.: Питер, 2005. — С. 896.
10. Джозеф Шмуллер. Освой самостоятельно UML 2 за 24 часа. Практическое руководство = Sams Teach Yourself UML in 24 Hours, Complete Starter Kit. — М.: Вильямс, 2005. — 416 с.
11. Filipova, Olga (2016). Learning Vue.js 2: learn how to build amazing and complex reactive web applications easily with Vue.js. ISBN 978-1-78646-113-1. Процитовано 2017-02-10.

**Електронні ресурси:**

1. Довідниково-інформаційний портал htmlbook [Електронний ресурс]. Режим доступу до ресурсу: http://htmlbook.ru/.
2. Современный учебник Javascript [Електронний ресурс]. Режим доступу до ресурсу: http://learn.javascript.ru.
3. Vue прогресивний JavaScript-фреймворк [Електронний ресурс] – Режим доступу до ресурсу: https://vuejs.org.
4. Що таке Vuex? [Електронний ресурс] – Режим доступу до ресурсу: https://vuex.vuejs.org.
5. Material Design Component Framework [Електронний ресурс] – Режим доступу до ресурсу: https://vuetifyjs.com.
6. Vue Router: Введення [Електронний ресурс] – Режим доступу до ресурсу: https://router.vuejs.org.
7. Vue I18n [Електронний ресурс] – Режим доступу до ресурсу: https://kazupon.github.io/vue-i18n.
8. Документація webpack [Електронний ресурс] – Режим доступу до ресурсу: https://webpack.js.org.
9. Babel - The compiler for next generation JavaScript [Електронний ресурс] – Режим доступу до ресурсу: https://babeljs.io.
10. Документація firebase [Електронний ресурс] – Режим доступу до ресурсу: https://firebase.google.com/docs.
11. Google Maps API - Google Developers [Електронний ресурс] – Режим доступу до ресурсу: https://developers.google.com/maps/documentation.
12. DB-Engines - Knowledge Base of Relational and NoSQL Database Management Systems [Електронний ресурс] – Режим доступу до ресурсу: https://db-engines.com.

# ДОДАТКИ

# ДОДАТОК А

Варіанти використання веб-сервісу для пошуку домашніх тварин



# ДОДАТОК Б

Діаграма компонентів та класів

# ДОДАТОК В

Структурна схема БД

