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, КІ та КБ

А.А.Єфіменко

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 клмпонентів), 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 clamponents), 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 6](#_Toc529910944)

[ABSTRACT 5](#_Toc529910945)

[ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ 7](#_Toc529910946)

[ВСТУП 8](#_Toc529910947)

[РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ ІНТЕРАКТИВНОГО WEB-СЕРВІСУ ПО ПОШУКУ ДОМАШНІХ ТВАРИН 10](#_Toc529910948)

[1.1. Постановка задачі 10](#_Toc529910949)

[1.2. Аналіз аналогів програмного продукту 12](#_Toc529910950)

[1.3. Вибір архітектури інтерактивного web-сервісу для пошуку домашніх тварин 14](#_Toc529910951)

[1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення 18](#_Toc529910952)

[Висновки до розділу 1 24](#_Toc529910953)

[РОЗДІЛ 2. ПРОЕКТУВАННЯ ІНТЕРАКТИВНОГО WEB-СЕРВІСУ ДЛЯ ПОШУКУ ДОМАШНІХ ТВАРИН 25](#_Toc529910954)

[2.1. Визначення варіантів використання та об’єктно-орієнтованої структури системи 25](#_Toc529910955)

[2.2. Розробка бази даних системи 32](#_Toc529910956)

[2.3. Проектування та реалізація алгоритмів роботи системи 35](#_Toc529910957)

[2.4 Реалізація додатку веб-сервісу бюро знахідок 40](#_Toc529910958)

[Висновки до розділу 2 42](#_Toc529910959)

[РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБ-СЕРВІСОМ БЮРО ЗНАХІДОК 44](#_Toc529910960)

[3.1. Порядок встановлення та налаштування системи 44](#_Toc529910961)

[3.2. Структура інтерфейсу. Інтерфейс та порядок роботи з веб-сервісом бюро знахідок 48](#_Toc529910962)

[3.3.Тестування роботи програмного комплексу 54](#_Toc529910963)

[Висновки до розділу 3 58](#_Toc529910964)

[ВИСНОВКИ 59](#_Toc529910965)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 61](#_Toc529910966)

[ДОДАТКИ 63](#_Toc529910967)

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 оголошеням. 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:

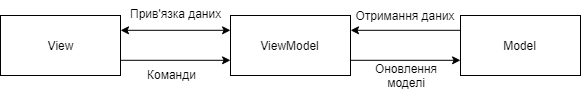


Рис.1.3.2. Узагальнена архітектура MVVM

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

## [2.1. 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. 2.1.1 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.



Рис. 2.1.1. Варіанти використання веб-сервісу для пошуку домашніх тварин

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 фраємворку Vue.

Аналіз функціональних вимог дозволив виділити наступні сутності, що забезпечать реалізацію програмного комплексу системи. На рис.2.2.1 наведено діаграму компонентів та класів. В розділі 2.2. буде наведена модель бази даних системи у відповідності до якої організований рівень моделі. Структура представлень буде наведена у 3 розділі.

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.

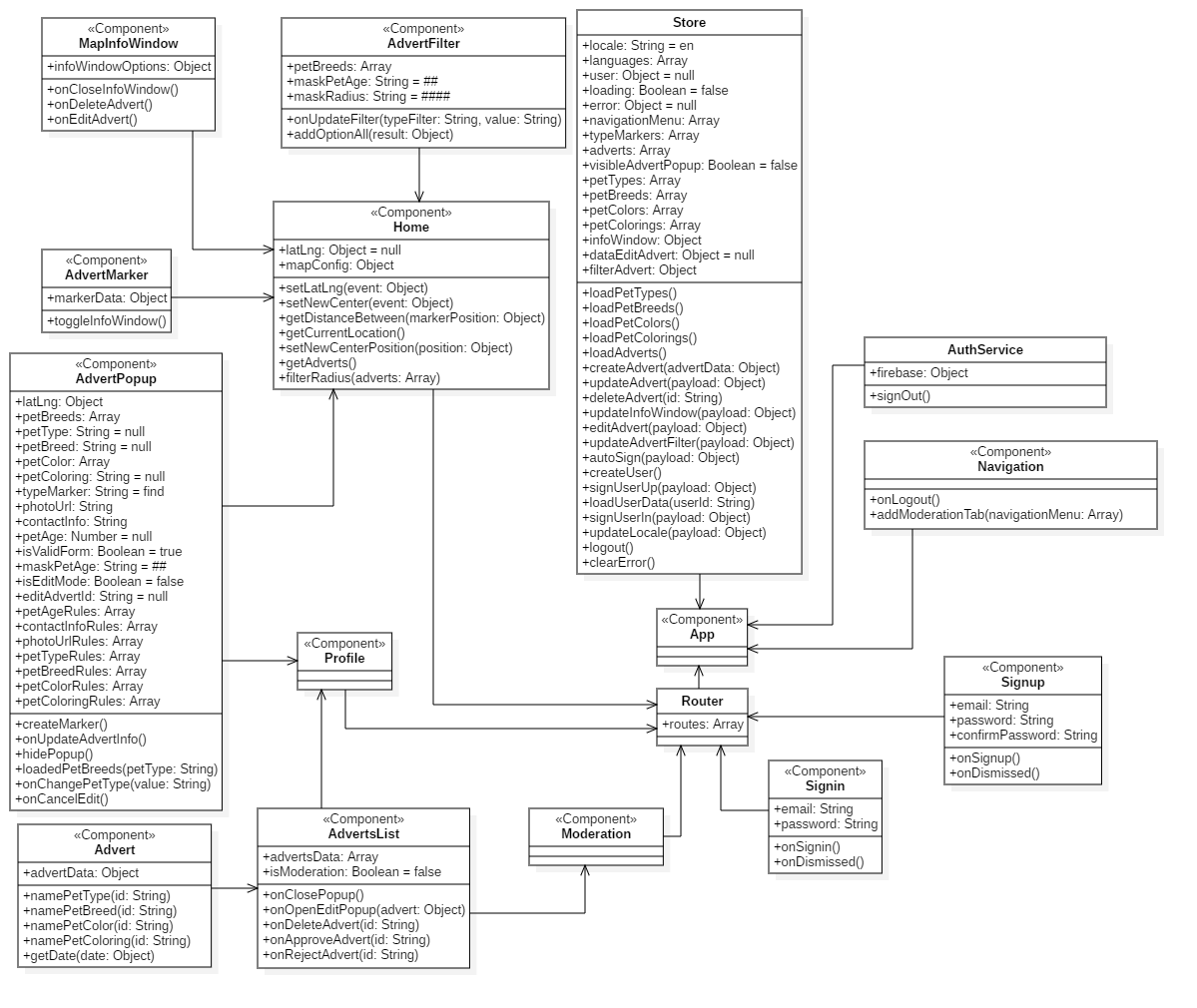


Рис. 2.2.1. Діаграма компонентів та класів

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.

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

Реалізована база даних Firebase Realtime складається з 6 колекцій, які містять у собі всі дані для роботи програми. База даних має назву lostFoundPets.

Структура бази даних наведена на рис. 2.2.1.

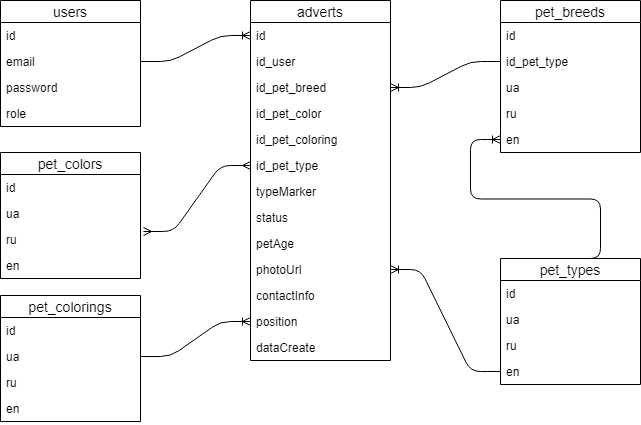


Рис.2.2.1. Структурна схема БД

Таблиця 2.2.1

Інформація про таблиці бази даних lostFoundBase

|  |  |
| --- | --- |
| Назва таблиці | Призначення |
| users | Колекція користувачів |
| adverts | Колекція оголошень |
| pet\_colors | Колекція кольорів домашньої тварини |
| pet\_colorings | Колекція забарвлення домашньої тварини |

Далі наведемо опис основних таблиць бази даних.

Таблиця «users» призначена для користувачів сайту. Структура таблиці наведена нижче:

Таблиця 2.2.2

Опис полів таблиці «users»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Назва | Тип даних | ПК | ЗП | Опис поля |
| id | Int(11) | + | - | Номер користувача |
| login | varchar(12) | - | - | Логін |
| password | varchar(34) | - | - | Пароль |
| email | varchar(36) | - | - | Електрона пошта |
| name | varchar(40) | - | - | ФІО |
| phone | varchar(13) | - | - | Номер телефонна |
| date\_birthday | Date | - | - | Дата народження |
| address | varchar(36) | - | - | Адреса |
| role | Int(11) | - | + | Номер ролі |
| photo\_url | Text | - | - | Посилання на фото |

Таблиця «ads» призначена для збереження оголошень створеними користувачами. Структура таблиці наведена нижче:

Таблиця 2.2.3

Опис полів таблиці «ads»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Назва | Тип даних | ПК | ЗП | Опис поля |
| id | Int(11) | + | - | Номер оголошення |
| name | varchar(26) | - | - | Назва |
| id\_type | Int(11) | - | + | Номер типа |
| description | Text | - | - | Опис |
| id\_type\_category | Int(11) | - | + | Номер типа категорії |
| image\_url | Text | - | - | Посилання на фото |
| date\_lost\_found | Date | - | - | Дата втрати/знахідки |
| date\_create\_ad | Date | - | - | Дата створення об’яви |
| id\_status | Int(11) | - | + | Номер статуса |
| id\_place | Int(11) | - | + | Номео місця |
| id\_user | Int(11) | - | + | Номер користувача |

Таблиця «users\_role» призначена для збереження ролей користувачів. Структура таблиці наведена нижче:

Таблиця 2.2.4

Опис полів таблиці «users\_role»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Назва | Тип даних | ПК | ЗП | Опис поля |
| id | Int(11) | + | - | Номер ролі |
| name\_role | varchar(18) | - | - | Назва ролі |

Таблиця «advert\_status» призначена для збереження статуса оголошення. Структура таблиці наведена нижче:

Таблиця 2.2.5

Опис полів таблиці «advert\_status»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Назва | Тип даних | ПК | ЗП | Опис поля |
| id | Int(11) | + | - | Номер статуса |
| name\_status | Text | - | - | Назва статуса |

Таблиця «advert\_type» призначена для збереження типа оголошення. Структура таблиці наведена нижче:

Таблиця 2.2.6

Опис полів таблиці «advert\_type»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Назва | Тип даних | ПК | ЗП | Опис поля |
| id | Int(11) | + | - | Номер типа |
| name\_type | varchar(20) | - | - | Назва типа |

Таблиця «advert\_type\_category» призначена для збереження типа категорії оголошення. Структура таблиці наведена нижче:

Таблиця 2.2.7

Опис полів таблиці «advert\_type\_category»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Назва | Тип даних | ПК | ЗП | Опис поля |
| id | Int(11) | + | - | Номер типа категорії |
| name\_type\_category | Text | - | - | Назва типа категорії |

Таблиця «advert\_info\_place» призначена для збереження про місце знайденої/загубленої речі. Структура таблиці наведена нижче:

Таблиця 2.2.8

Опис полів таблиці «advert\_info\_place»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Назва | Тип даних | ПК | ЗП | Опис поля |
| id | Int(11) | + | - | Номер місця |
| coordinates\_map | Text | - | - | Координати місця |
| id\_region | Int(11) | - | + | Номер регіона |

Таблиця «regions» призначена для збереження регіонів. Структура таблиці наведена нижче:

Таблиця 2.2.9

Опис полів таблиці «regions»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Назва | Тип даних | ПК | ЗП | Опис поля |
| id | Int(11) | + | - | Номер регіона |
| name\_region | Text | - | - | Назва регіона |

Отже реалізована база даних у відповідності до вимог реляційної моделі забезпечує збереження та колективний доступ до інформації бюро знахідок. База даних складається з 8 таблиць.

## 2.3. Проектування та реалізація алгоритмів роботи системи

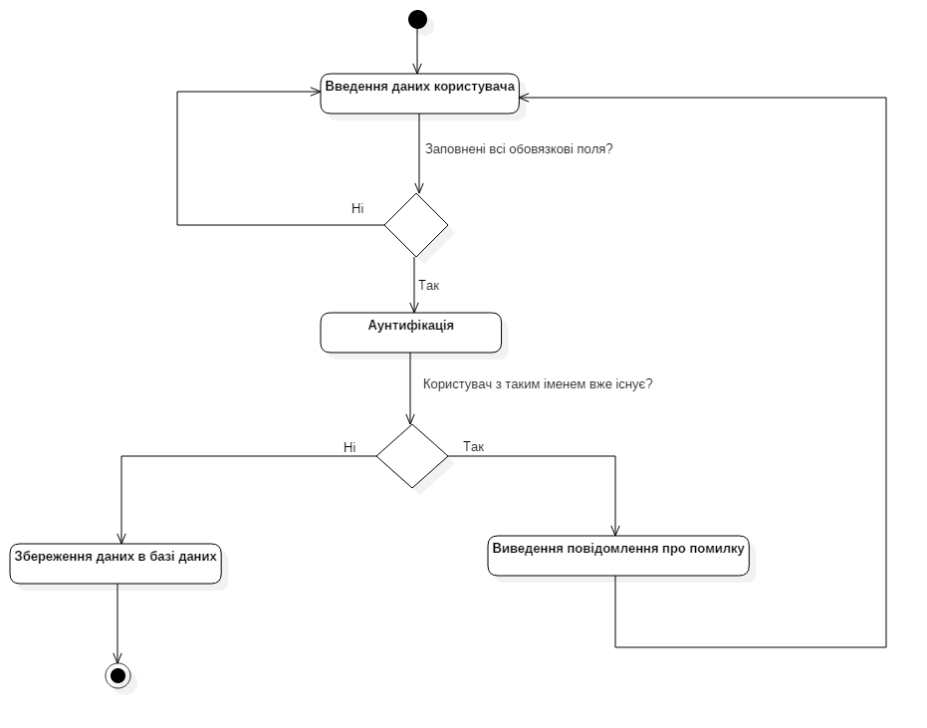


Рис.2.3.1. Діаграма активності “Реєстрація користувача”

Як показано на (рис.2.3.1) ми бачимо, після того як користувач перейшов на сторінку реєстрації йому потрібно буде ввести данні: логін, пароль, підтвердження паролю, електрону пошту. Після того перевіряється чи заповнені обов’язкові поля, якщо ні то йому виводиться повідомлення які поля не були заповнені. Якщо ж всі поля заповнені відбувається аунтифікація – перевіряється чи користувач с такими логіном і електронною поштою зареєстрований, якщо є збіги то виводиться помилка. Якщо ж так то користувач зберігається в базі даних і стає зареєстрованим користувачем.

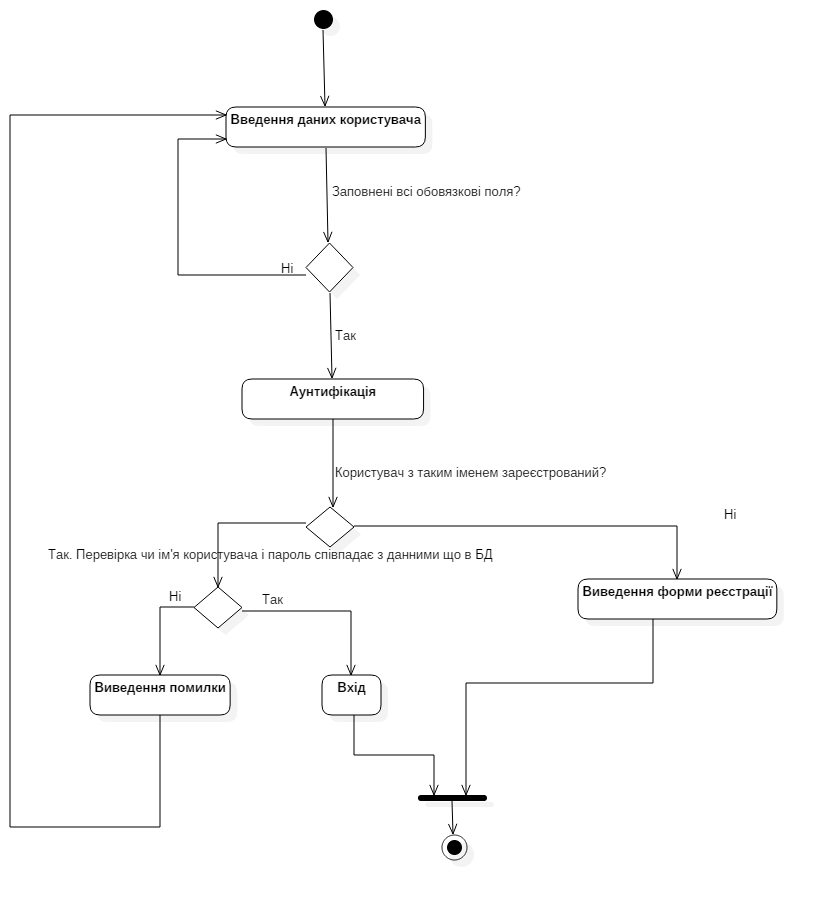


Рис.2.3.2. Діаграма активності “Авторизація користувача”

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

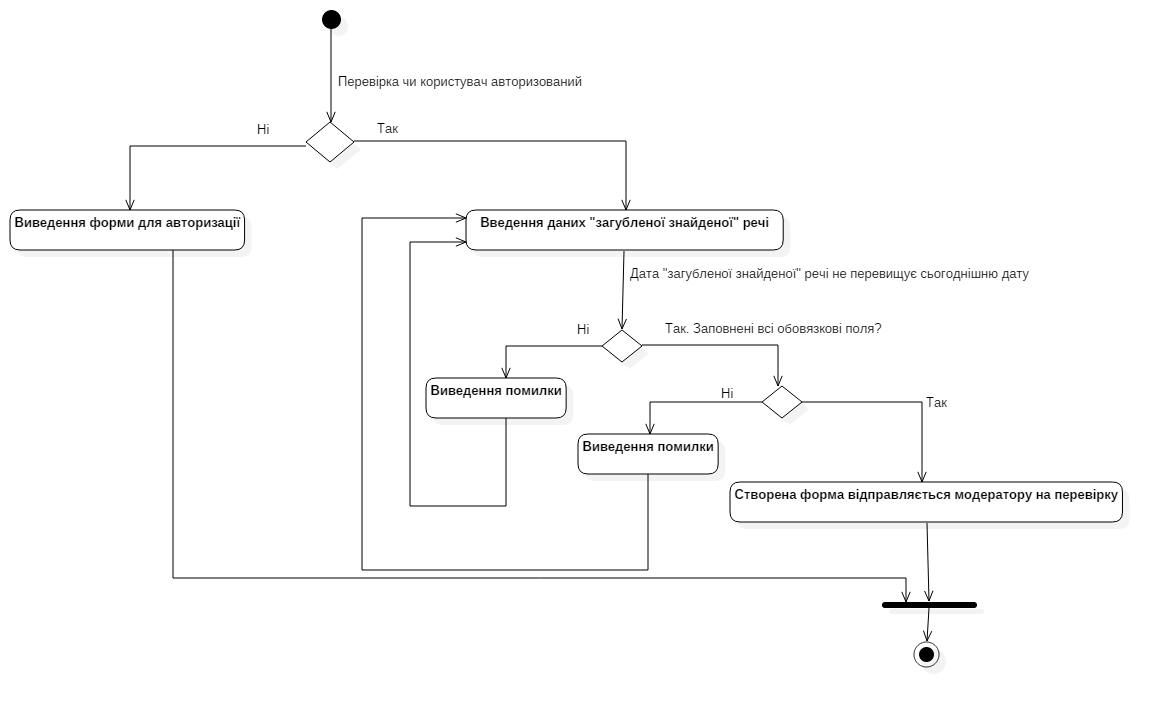


Рис.2.3.3. Діаграма активності “Додавання оголошення”

При додаванні оголошення про знайдену/загублену річ 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 створений запис 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. 2.3.4.

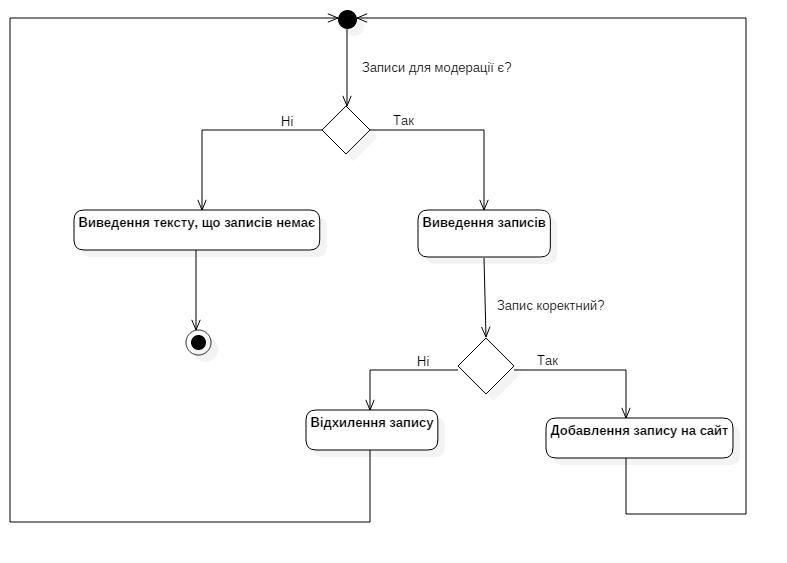


Рис.2.3.4. Діаграма активності “Модерація”

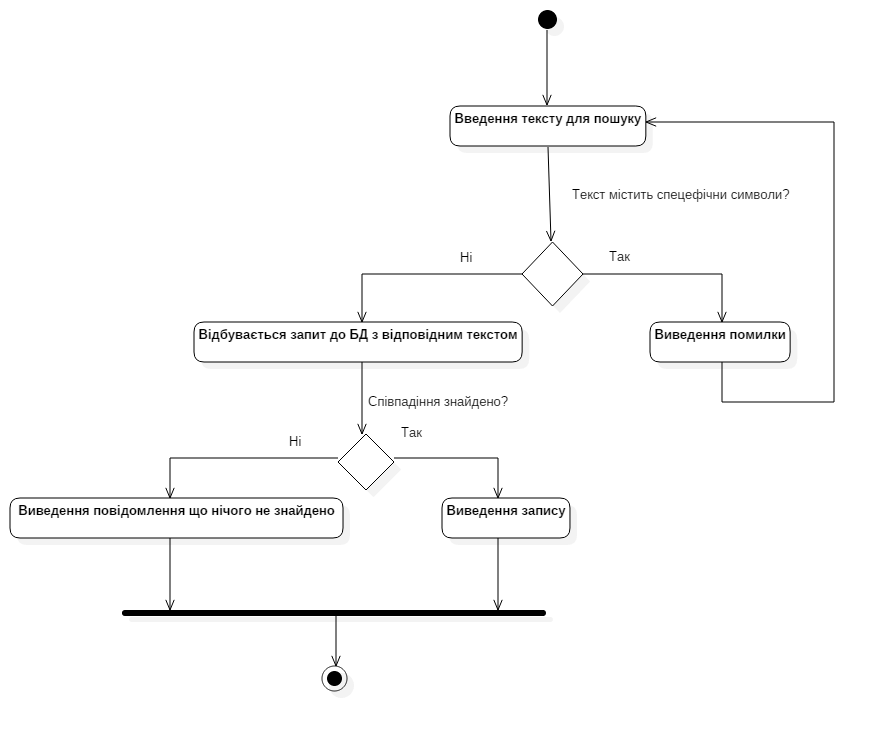


Рис.2.3.5. Діаграма активності “Пошук”

Для пошуку користувачу потрібно ввести текст у відповідне поле, після чого відбувається перевірка чи текст не має специфічних символів, це робиться з метою, щоб зловмисник не зміг отримати доступу до бази даних. Якщо текст має специфічні символи, то йому виводиться помилка, що текст є не коректний, якщо ж ні, то відправляється запит до БД, де перевіряється чи є записи з відповідним текстом. Якщо такого оголошення не знаходить, то виводиться повідомлення, що нічого не знайденого в іншому випадку виводиться оголошення в яких було знайдено відповідний текст.

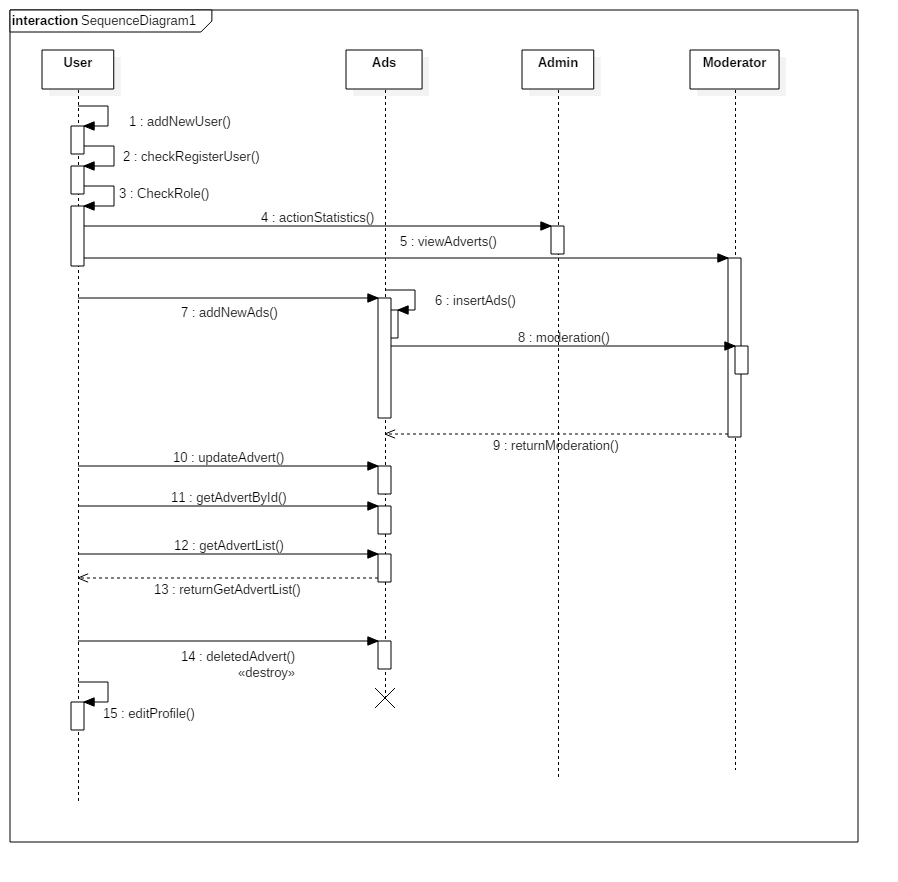


Рис. 2.3.6. Діаграма послідовності

З рис. 2.3.6 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 два статуси знахідка та втрата. 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 веб-сервісу бюро знахідок та проаналізовано взаємодію класів системи під час процесів створення, редагування та пошуку оголошення, а також реєстрації та авторизації на сайті.

## 2.4 Реалізація додатку веб-сервісу бюро знахідок

Для реалізації системи було написано клас DB – для з'єднання з базою даних та налаштуванням параметрів з'єднання. Для з'єднання додатку з базою даних використовуємо технологію PDO.

Для додавання нового оголошення в БД було реалізовано функцію insertAds(dataAd) в класі Ads, яка отримує дані з контролера.

public function addNewAds($dataAd, $userId)

{

$name = $dataAd["adName"];

$type = $dataAd["type"];

$description = $dataAd['adDescription'];

$type\_category = $dataAd["adTypeCategory"];

$date\_lost\_found = $dataAd["adDate"];

$coordinates\_map = $dataAd["coordinates"];

$date\_create\_ad = date("Y-m-d");

$region = $dataAd["region"];

if(empty($dataAd["adUrlImage"])) {

$image\_url = $this->defaultImage;

} else {

$image\_url = $dataAd["adUrlImage"];

}

$query = self::$db->query('SELECT id FROM regions WHERE name\_region="' .$region.'"');

$id\_region = $query->fetchAll();

$id\_region = $id\_region[0]["id"];

$query = "INSERT INTO advert\_info\_place VALUES ('','$coordinates\_map','$id\_region')";

$sth = self::$db->exec($query);

$id\_place = self::$db->lastInsertId();

$query = "INSERT INTO ads VALUES ('','$name','$type','$description','$type\_category','$image\_url','$date\_lost\_found','$date\_create\_ad','$this->statusModeration','$userId', '$id\_place')";

$sth = self::$db->exec($query);

}

Для фільтрації оголошень по регіонам і типом категорії, був реалізований ajax запит на функцію getAdvertList(), в яку аргументом передавали масив даних для фільтра після чого викликалась функція getQueryFilter().

public function getAdvertList($type = null, $status = '1', $filters = null, $search = null)

{

$checkType = Ads::checkTypeAdvert($type);

$filters = Ads::getQueryFilter($filters);

$search = Ads::getQuerySearch($search);

$result = self::$db->query('SELECT ads.\*, advert\_status.name\_status, regions.name\_region, advert\_type\_category.name\_type\_category FROM ads

JOIN advert\_status on ads.id\_status = advert\_status.id

JOIN advert\_info\_place on ads.id\_place = advert\_info\_place.id

JOIN regions on advert\_info\_place.id\_region = regions.id

JOIN advert\_type\_category on ads.id\_type\_category = advert\_type\_category.id

WHERE id\_status = '.$status.''.$checkType.''.$filters.''.$search.' ORDER BY ads.id DESC;');

$result = $result->fetchAll();

return $result;

}

public static function getQueryFilter($filters)

{

$filter = "";

if($filters != null) {

$filters = array\_slice($filters, 1);

foreach ($filters as $key => $value) {

if(!empty($filters[$key])) {

$filters[$key] = explode(",", $filters[$key]);

$filter .= " AND ( ";

foreach ($filters[$key] as $elemFiltr) {

$filter .= $key." = '$elemFiltr' OR ";

}

$filter = substr($filter, 0, -3);

$filter .= ") ";

}

}

}

return $filter;

}

Для пошуку була використана та сама функція getAdvertList(), в яку передавалось аргументом значення з пошуку і після чого викликалась функція getQuerySearch().

public static function getQuerySearch($searchText)

{

$search = "";

if(!empty($searchText)) {

$search = " AND ( name LIKE '%".$searchText['searchText']."%' OR description LIKE '%".$searchText['searchText']."%' ) ";

}

return $search;

}

Для модерації об’яв використовувалась функція moderation(), класа Moder в яку аргументом передавалось статус (схвалено або відхилено) та id об’яви, після чого викликалась функція changeStatusAdvert() класу Ads, яка змінювала статус оголошення.

public function moderation($statusId)

{

$status = "";

$statusId = explode("-", $statusId);

$advertId = $statusId[1];

if($statusId[0] == "accept") {

$status = "1";

} else {

$status = "3";

}

Ads::changeStatusAdvert($advertId, $status);

}

public static function changeStatusAdvert($advertId, $status)

{

$query = "UPDATE ads SET id\_status = '$status' WHERE id = '$advertId'";

$sth = self::$db->exec($query);

}

## Висновки до розділу 2

Веб-сервіс бюро знахідок реалізує функціонал за допомогою 5 контролерів та 4 моделей, що виконують основні функції сайту, які передбачають реєстрацію та авторизацію користувачів, створення, редагування та видалення оголошення. Також містять модерацію, відправлення повідомлення автору створеної об’яви, фільтрацію та пошук по об’явам.

В розділі реалізована база даних у відповідності до вимог реляційної моделі, яка забезпечує збереження та колективний доступ до веб-сервісу бюро знахідок. База даних складається з 8 таблиць, основними таблицями є: Users та Ads.

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

[РОЗДІЛ 3.](file:///C:\планшет\Магістерська%20робота1.docx#_Toc423399389) ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБ-СЕРВІСОМ БЮРО ЗНАХІДОК

## 3.1. [Порядок](file:///C:\планшет\Магістерська%20робота1.docx#_Toc423399390) встановлення та налаштування системи

Веб-сервіс бюро знахідок це сайт створений на PHP – він має конфігураційні файли, в яких прописується ім'я домену або шлях до файлів. Перед перенесенням системи на хостинг необхідно змінити дані в конфігураційних файлах. Сайт використовує MySQL базу, отже потрібно перенести базу даних на сервер і оновити конфігураційні файли. Порядок дій в такому випадку наступний:

1. Створюння копії (бекап) бази даних на старому сервері. Якщо поточний провайдер надає інструмент для створення MySQL копій, тоді можна скористатись ними. В PhpMyAdmin можна вибрати базу даних і обрати команду 'Експорт'. В діалоговому вікні Зберегти як (Зберегти як) слід вказати параметр 'архівована'. Після цього має початися скачування бази.

2. Створення бази даних на сервері. Як тільки скачування закінчиться, визначити назву файл. Перед відновленням, потрібно створити базу на сервері та користувача, якщо назва створеної БД буде відрізнятися від тієї яку ви імпортуєте, то прийдеться в конфігураційних файлах змінити назву для неї.

3. Імпорт даних бази на сервер. Після створення порожньої база даних, в PhpMyAdmin на сервері слід обрати команду 'Імпорт' та вибрати копію бази. Почнеться процес відновлення бази даних. Цей процес може зайняти деякий час, якщо розмір бази занадто великий. Тепер, коли база даних відновлена, потрібно відкрити конфігураційний файл скрипта і змінити значення хоста бази даних, ім'я бази даних, ім'я користувача бази даних і пароль бази даних.

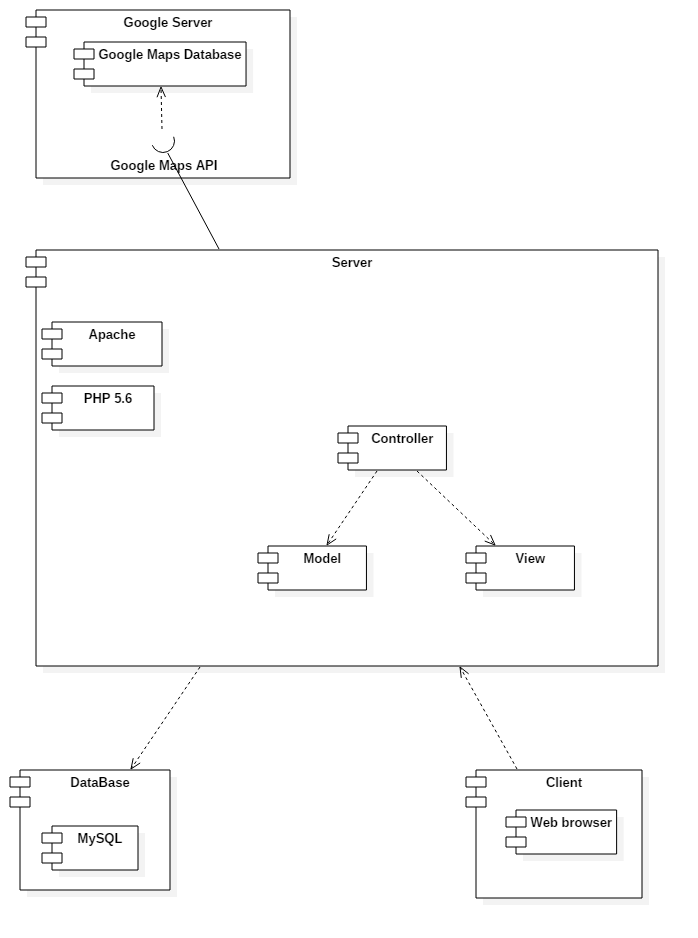


Рис. 3.1.1. Діаграма розгортання системи

Параметри налаштування бази даних зберігаються в папці config з файлом db\_params.php:

return array(

'host' => 'localhost',

'dbname' => 'lostFoundBase',

'user' => 'admin',

'password' => '322',

);

Основні налагодження роботи системи знаходяться в папці components з файлом Router.php:

class Router

{

private $routes;

public function \_\_construct()

{

$routesPath = ROOT . '/config/routes.php';

$this->routes = include($routesPath);

}

private function getURI()

{

if (!empty($\_SERVER['REQUEST\_URI'])) {

return trim($\_SERVER['REQUEST\_URI'], '/');

}

}

public function run()

{

$uri = $this->getURI();

foreach ($this->routes as $uriPattern => $path) {

if (preg\_match("~$uriPattern~", $uri)) {

$internalRoute = preg\_replace("~$uriPattern~", $path, $uri);

$segments = explode('/', $internalRoute);

$controllerName = array\_shift($segments) . 'Controller';

$controllerName = ucfirst($controllerName);

$actionName = 'action' . ucfirst(array\_shift($segments));

$parameters = $segments;

$controllerFile = ROOT . '/controllers/' .

$controllerName . '.php';

if (file\_exists($controllerFile)) {

include\_once($controllerFile);

}

$controllerObject = new $controllerName;

$result = call\_user\_func\_array(array($controllerObject, $actionName), $parameters);

if ($result != null) {

break;

}

}

}

}

}

З метою забезпечення можливості дистанційної роботи і незалежності від операційної системи користувача, даний програмний комплекс потребує наявності сервера і комп’ютера користувача. При необхідності, комп’ютер користувача може виконувати і роль сервера.

Для роботи з даним програмним продуктом можна використовувати будь-яку з розповсюджених ОС (Windows, UNIX, LINUX, Мас OS SOLARIS). Тому конфігурація комп’ютера, що необхідна для роботи буде залежати головним чином від вибору операційної системи.

В операційній системі Windows 7 для більш-менш зручної роботи програми буде достатньо наступної конфігурації:

1. Процесор: з тактовою частотою від 1300 MHz;

2. ОЗП: 1 ГБ (для 32-бітної версії) або 2 ГБ (для 64-бітної версії)

3. Відео-карта: будь-якої марки та типу, мінімум пам’яті 512 Mb;

4. Жорсткий диск: 16 ГБ для 32-бітної версії) або 20 ГБ (для 64-бітної версії) вільного простору на жорсткому диску;

5. Монітор: для зручнішого огляду веб-сайту, краще використати монітор, діагональ якого не менше 17 дюймів, 1280х1024, при встановленій частоті 60 Гц, якістю кольорової передачі 32 біта;

6. Маніпулятор миші: будь-якої марки та типу;

7. Клавіатура: будь-якої марки та типу;

Таким чином розроблений програмний код не потребує спеціалізованого апаратного забезпечення, додаткових налаштувань та засобів для розгортання крім стандартних для більшості сучасних сайтів.

Так, як розроблений веб-сайт є кросс-браузерним, для роботи з ним можна скористатися будь-яким веб-браузером (IE, Opera, Mozilla FireFox, Safari, Google Chrome), але він обов’язково повинен підтримувати CSS3 та HTML5. Також для того, щоб відкрити сайт на телефоні або планшеті потрібно, щоб на ньому були останні оновлення, для коректного відображення верстки.

Отож описавши та проаналізувавши встановлення та налаштування системи ми бачимо, що програмний код не потребує спеціалізованого апаратного забезпечення, додаткових налаштувань та засобів для розгортання крім стандартних для більшості сучасних сайтів.

## 3.2. [Структура](file:///C:\планшет\Магістерська%20робота1.docx#_Toc423399391) інтерфейсу. Інтерфейс та порядок роботи з веб-сервісом бюро знахідок

Для авторизації на веб-сайті потрібно натиснути кнопку «Авторизация» (рис. 3.2.1), після чого з’являється форма для вводу даних користувача.

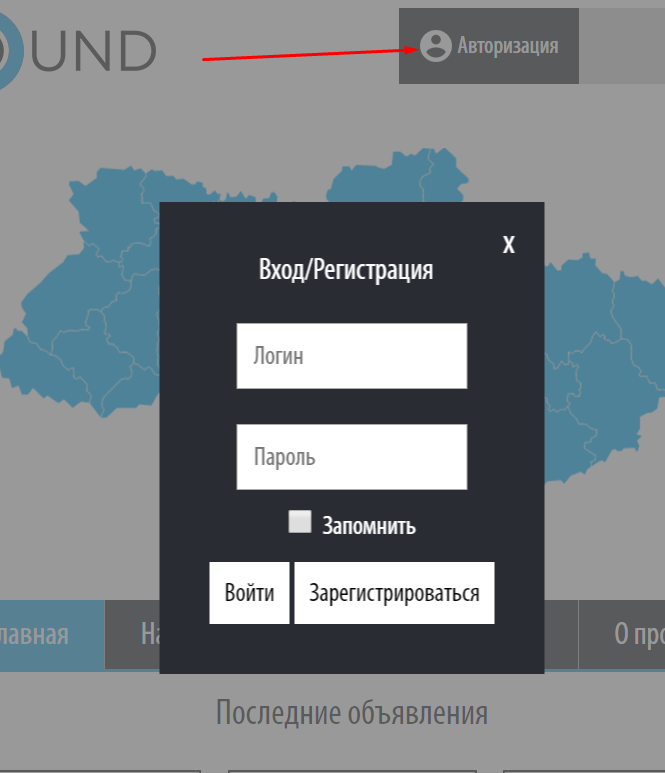


Рис.3.2.1. Форма авторизації

Якщо ж користувач ще не зареєстрований, потрібно натиснути кнопку «Зарегистрироваться» і користувача пере направить на сторінку реєстрації (рис. 3.2.2).

Після того як користувач зареєструвався або авторизувався він може перейти в свій профіль, де буде відображено інформацію про нього та його оголошення (рис. 3.2.3).

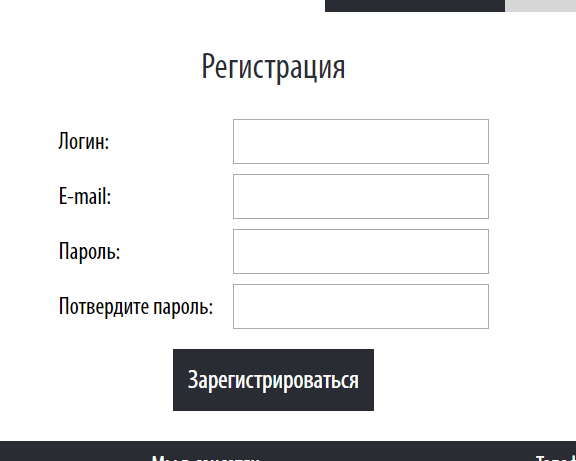


Рис.3.2.2. Форма реєстрації

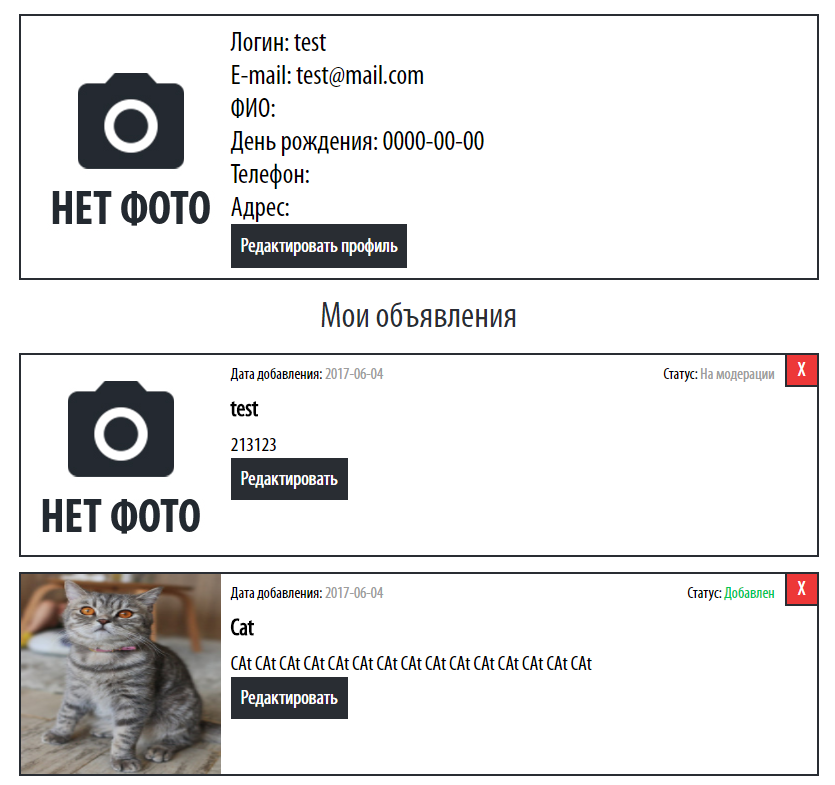


Рис.3.2.3. Сторінка профілю користувача

В своєму профілі, користувач може редагувати інформацію про себе натиснувши кнопку «Редактировать профиль» та редагувати і видаляти оголошення. Біля кожної об’яви відображається її статус, це було зроблено, щоб користувач розумів на якій стадії його об’ява.

Тепер користувач авторизований і він з легкістю може додати нове оголошення, для цього йому потрібно натиснути кнопку «Добавить находку» або «Добавить пропажу», після його перенаправляє на сторінку с формою створення оголошення (рис.3.2.4).

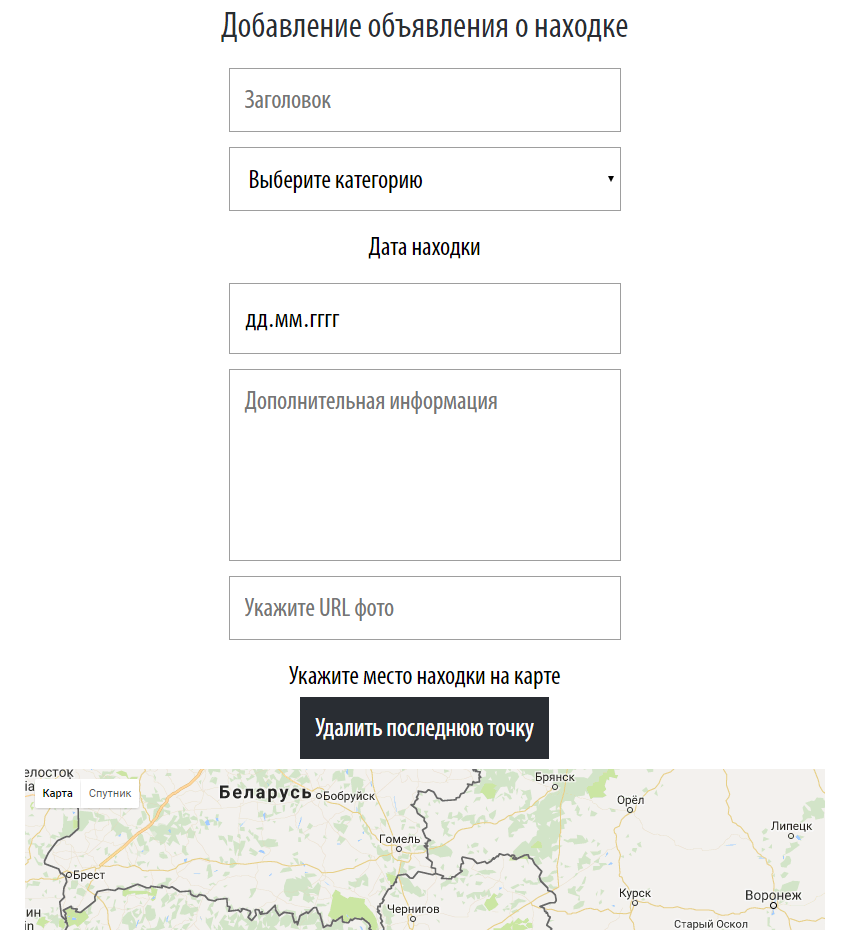


Рис.3.2.4. Форма створення оголошення

На цій сторінці користувачу обов’язково потрібно заповнити такі поля: назву оголошення, зі списку вибрати до якого типу належить річ та вказати дату знайденої/загубленої речі. Також користувач може вказати доповнену інформацію, картинку речі та місце на карті.

Якщо модератор схвалить об’яву, то вона буде відображатись на сайті і її можна буде більш детально переглянути (рис.3.2.5).

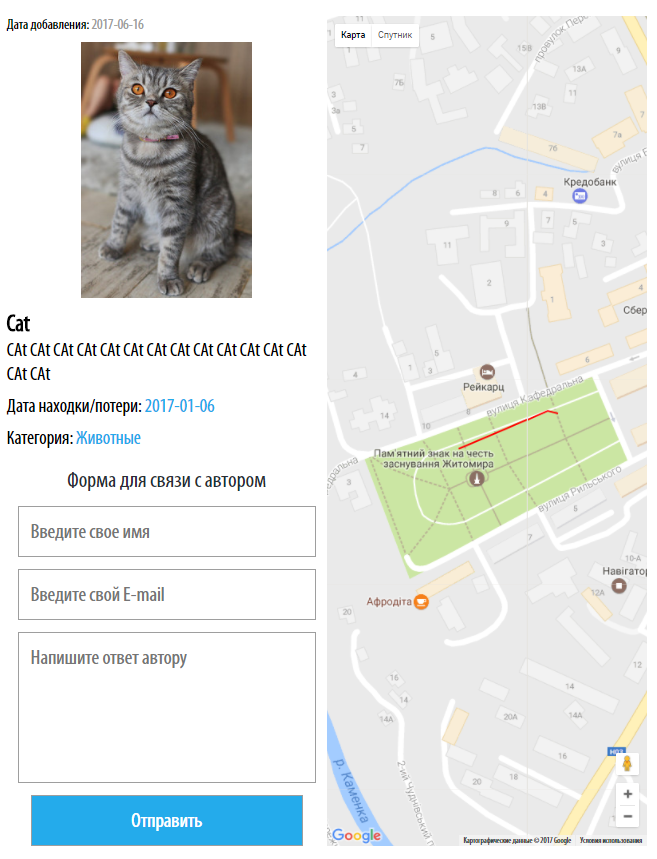


Рис.3.2.5. Сторінка детального перегляду оголошення

Щоб зв’язатись з автором оголошення, користувачеві потрібно заповнити форму. У цій формі всі поля є обов’язковими.

Якщо користувач втратив свою річ і хоче її повернути, він може скористатися фільтром для пошуку цієї речі на сайті (рис.3.2.6).

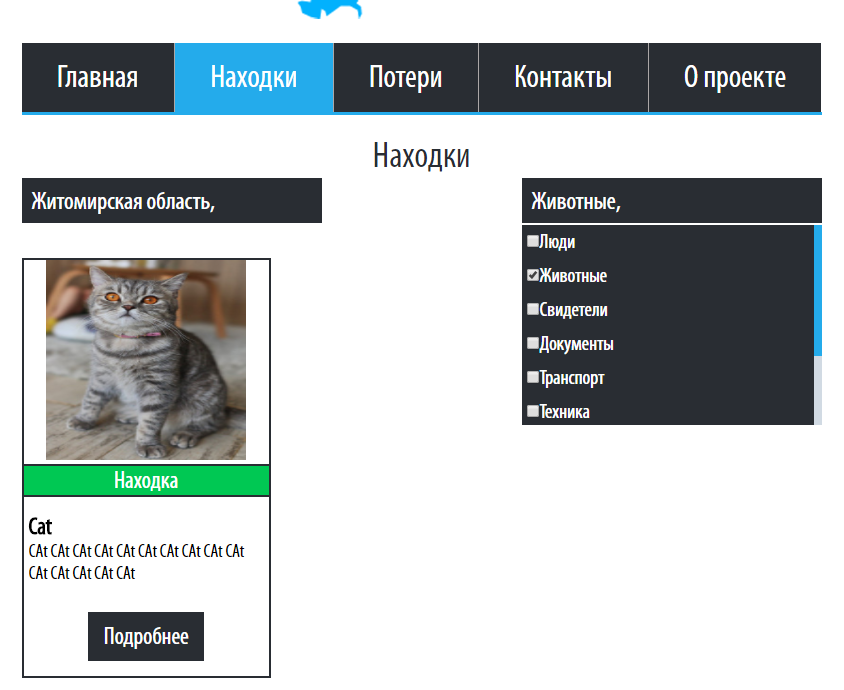


Рис.3.2.6. Сторінка с фільтрацією оголошень

Також він може скористатися пошуком на сайті (рис.3.2.7).

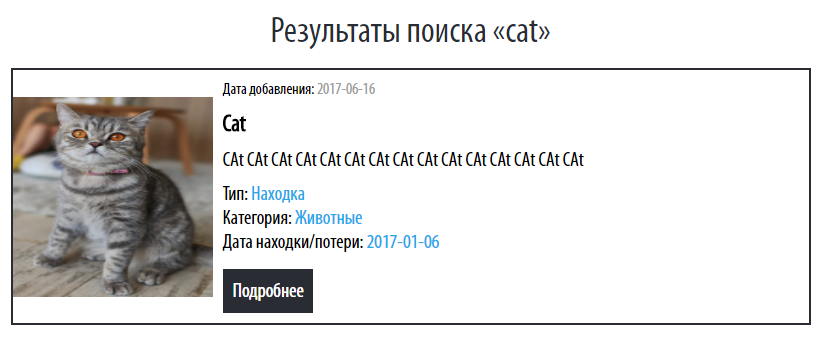


Рис.3.2.7. Сторінка пошуку

Для модерації оголошень потрібно перейти на сторінку модерації, для цього у модератора з’являється додаткова кнопка завдяки якій він це робить. Після чого він схвалює або відхиляє оголошення за допомогою кнопок «Принять» та «Отклонить» (рис.3.2.8).



Рис.3.2.8. Сторінка модерації

Отже покроковий опис роботи з даним програмним продуктом показує, що веб-сервіс має простий та зрозумілий користувачу структурний інтерфейс, а також логічний порядок роботи з ним, що забезпечує швидке навчання роботи з сайтом.

Відповідність представлень сторінкам сайту наведено в табл.3.2.1.

Таблиця 3.2.1

Відповідність представлень сторінкам сайту

|  |  |
| --- | --- |
| Сторінка сайту | Представлення |
| Головна | site/index.php |
| Знахідки/Втрати | site/lossesFinds.php |
| Контакти | site/contacts.php |
| Про проект | site/about.php |
| Особистий кабінет користувача | user/profile.php |
| Модерація | moderator/moderation.php |
| Статистика | admin/statistics.php |
| Детальний перегляд оголошення | ads/detail.php |
| Результати пошуку | ads/search.php |
| Реєстрація | user/register.php |
| Створення нового оголошення | ads/createAds.php |
| Редагування оголошення | ads/editAds.php |

## [3.3.Тестування](file:///C:\планшет\Магістерська%20робота1.docx#_Toc423399394) роботи програмного комплексу

Планується три етапи проведення процесу тестування. Перший етап полягає в аналізі постановки задачі, складанні критичного чек-листа, складанні тест плану, а також часткового прогону функціональних тестів. Другий етап буде присвячений деталізації функціонального чек-листа і детальному прогону функціональних тестів з виявленням і описом дефектів. На третьому етапі буде вироблено стрес-тестування з описом знайдених дефектів. Таким чином, досягається максимальна деталізація глибини тестування, що, в свою чергу, дозволяє більш точно визначити витрачені ресурси, а так само дозволяє розробникам ПЗ почати виправляти дефекти на ранніх етапах.

На першому етапі буде застосований Смоук тестінг при якому будуть уточнюватися вимоги, визначаться і конфігурироваться тестові середовища. До початку другого етапу будуть сформований критичний чек-лист, а так же чек-лист по функціональному тестуванню і юзер сторі. На другому етапі проводиться детальне тестування функціоналу ПЗ, збираються і описуються дефекти.

Кожен чек-лист проганяється для кожного браузера. Третій етап завершує роботи по тестуванню. В ньому проводиться встановлений набір тестів для виявлення вразливостей. Такий вид тестування досить витратний за часом, тому необхідний набір тест кейсів.

Браузери затверджені до перевірки:

− Internet Explorer 11

− Firefox 53

− Google Chrome 58

− Opera 44

− Safari 10.1

Смоук тестування. Мета: Накидати скелет чек-листів для функціонального тестування та стрес-тестування.

Даний метод застосовується з мінімальним набором тестів і мінімальною постановкою задачі. Метою даного тестування не є виявлення помилок, хоча якщо на даному етапі будуть виявлені явні дефекти, то вони будуть зафіксовані.

Функціональне тестування. Мета: Виявлення функціональних помилок, невідповідностей в постановці задач і очікуванням користувача шляхом реалізації стандартних, а також нетривіальних тестових сценаріїв.

Таблиця 3.3.1

Класифікація функцій

|  |  |
| --- | --- |
| Вид тестування | Дата |
| Тестування авторизації/реєстрації | 01.05.17 |
| Додавання об’яв | 03.05.17 |
| Редагування об’яв | 05.05.17 |
| Видалення об’яв | 07.05.17 |
| Тестування модерації | 08.05.17 |
| Тестування пошуку | 11.05.17 |
| Тестування фільтрації об’яв | 14.05.17 |
| Тестування навантаження на сервер | 16.05.17 |
| Тестування кроссбраузерності | 18.05.17 |
| Тестування форми «Обратная связь» | 20.05.17 |
| Тестування редагування профілю | 21.05.17 |

Тестування в певному середовищі. Мета: Перевірити коректну роботу і дизайн ПЗ в різних браузерах і при різних дозволах монітору.

Таблиця 3.3.2

Параметри конфігураційного тестування

|  |  |  |  |
| --- | --- | --- | --- |
| Браузер/Розмір екрану | 1024\*768 | 1280\*800 | 1680\*1050 |
| Internet Explorer 11 | Так | Так | Так |
| Firefox 53 | Так | Так | Так |
| Google Chrome 58 | Так | Так | Так |
| Opera 44 | Так | Так | Так |
| Safari 10.1 | Так | Так | Так |

Окремі конфігурації для перевірки кожен браузер з відключеним JS і відключені зображення.

Стрес-тестування Мета: Виявити уразливості в роботі Проекту шляхом використання великих даних, довгі посилання, некоректні дані тощо.

Тестування безпеки – будемо перевіряти чи немає у користувачів доступу до службових / закритим сторінкам а так само проводить перевірку захисту всіх критично важливих сторінок (наприклад, розділу адміністрування сайту) від зовнішнього впливу.

Інструменти для проведення тестування:

* 1. Xenu Link Sleuth це один з найбільш корисних інструментів в пошуковій оптимізації. За допомогою даного інструменту ми будемо шукати биті (непрацюючі) посилання на заданому ресурсі, складати карту сайту, знаходити не унікальні тайтли
  2. JMeter це інструмент для проведення навантажувального тестування, що розробляється Apache Software Foundation, підпроекту Jakarta. Хоча спочатку JMeter розроблявся як засіб тестування веб-застосунків, натепер він здатний проводити навантажувальні тести для JDBC-з'єднань, FTP, LDAP, SOAP, JMS, POP3, IMAP, HTTP і TCP.
  3. Для кросбраузерного тестування використовуймо Інтернет сервіси, які можуть надати реальні пристрої з необхідними браузерами в роздільне користування. Browserstack.com - хмарний симулятор популярних операційних систем і браузерів. За допомогою програми PICT та метода allpairs будуємо таблицю перевірок.

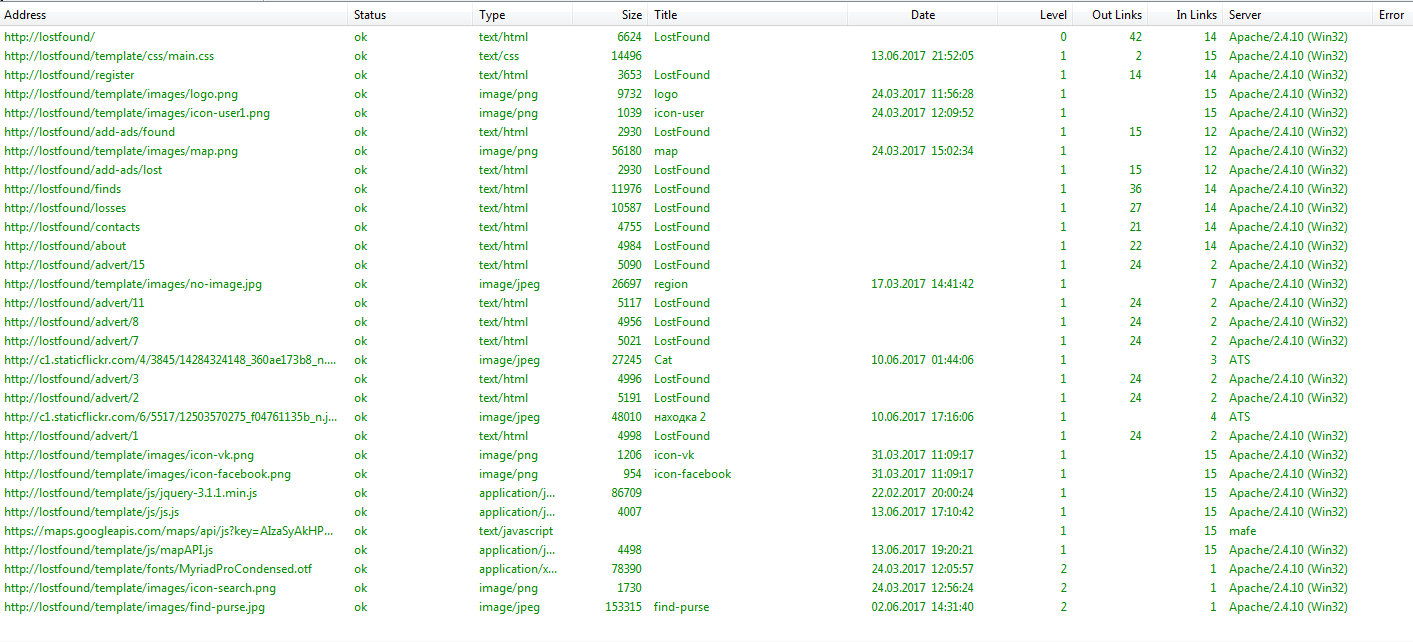


Рис. 3.3.1. Перевірка сайту на биті посилання

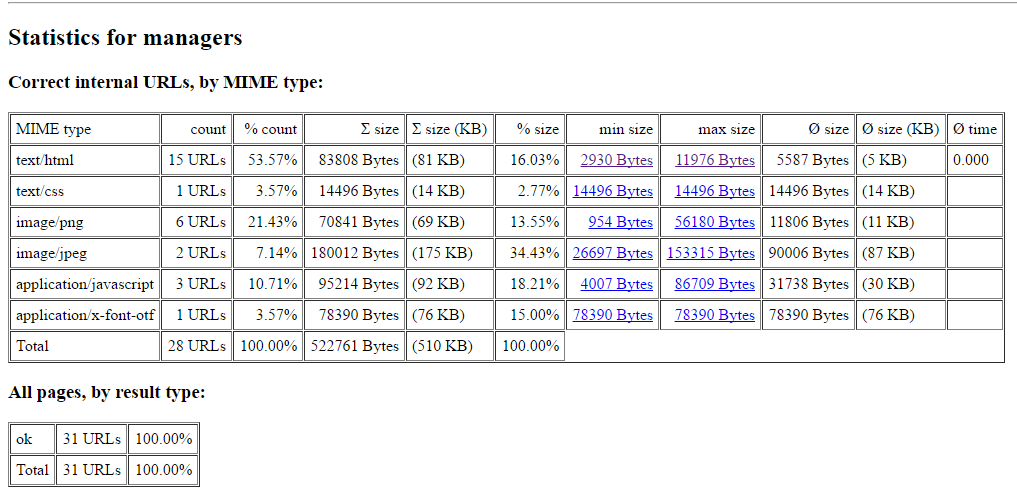


Рис. 3.3.2. Загальна статистика перевірки сайту

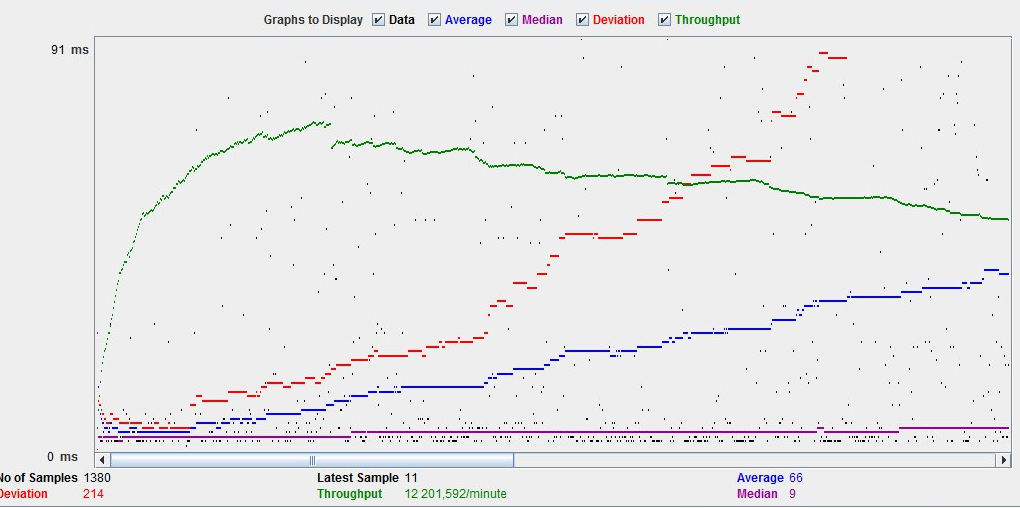


Рис. 3.3.3. Графік при навантаженому тестуванню

Для навантаженого тестування було вказано 50 користувачів які одночасно зайшли на сайта та почали працювати з ним. Як видно з (рис. 3.3.3) максимальний відгук користувачу 91 ms.

Отже було проведено комплексне тестування веб-сервісу бюро знахідок, що виявило основні помилки та застереження щодо роботи програми. Всі помилки було виправлено, на даний час сайт працює коректно та виконує всі поставлені задачі.

## Висновки до розділу 3

В даному розділі було наведено порядок налаштування системи та визначено, що розроблений програмний код не потребує спеціалізованого апаратного забезпечення, додаткових налаштувань та засобів для розгортання крім стандартних для більшості сучасних сайтів.

Покроковий опис роботи з даним програмним продуктом показує, що веб-сервіс має простий та зрозумілий користувачу структурний інтерфейс, а також логічний порядок роботи з ним, що забезпечує швидке навчання роботи з сайтом.

Реалізований додаток містить в своєму складі інтерфейси адміністратора, модератора та користувача. Для адміністраторів передбачено інтерфейс для перегляду статистики. Для модератора передбачено модерація оголошень створеними користувачами. Для зареєстрованих користувачів передбачено інтерфейс для перегляду своїх оглошень, створеня, редагування та видалення оголошень, відправлення повідомлення автору об’яви.

Було проведене комплексне тестування, що виявило основні помилки та застереження щодо роботи програми. Всі помилки було виправлено, на даний час сайт працює коректно та виконує всі поставлені задачі.

# ВИСНОВКИ

Під час виконання випускної роботи бакалавра було спроектовано та реалізовано веб-сервіс бюро знахідок, 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 бюро знахідок та необхідність реалізації засобів управління контентом сайту написання функціональних модулів, створення, редагування, видалення, фільтрація та пошук оголошень.
2. Проаналізовано аналоги програмного продукту, такі як: «Всеукраинское виртуальное онлайн-бюро находок «Верни»», «Информационно поисковый портал РФ «Бюро находок»», «Единая Национальная Служба «Бюро Находок»», громадська організація «LUCKFIND» та визначено, що основним рисами нової системи мають бути наявний веб-інтерфейс, модуль створення, редагування та видалення оголошень, модуль модерації, модуль відправки повідомлення автору створеного оголошення, модуль фільтрації та реєстрації з наступною авторизацією.
3. Обрано архітектурні шаблони – клієнт-серверний та паттерн MVC. Встановлено, що даний програмний продукт використовує архітектурні шаблони, які забезпечують технологічні можливості для побудови гнучкого додатку із дотриманням принципів ООП. Такий підхід надає можливість на успішну реалізацію проекту та подальшу його модифікацію та розвиток.
4. Для реалізації було вибрано наступні засоби: мову розмітки гіпертекстових документів HTML5; каскадні таблиці стилів CSS3; браузерну мову програмування JavaScript та бібліотеку jQuery; мову програмування РНР; середовище керування базами даних MySQL; пакет OpenServer; графічний редактор Adobe Photoshop CC 2014; середа розробки Atom 1.14.4.
5. Визначено функціонал, який передбачає реєстрацію та авторизацію користувачів, створення, редагування та видалення оголошень. Веб-сайт містить методи модерації, фільтрація та пошуку оголошень.
6. Створено базу даних у відповідності до вимог реляційної моделі та встановили зв'язок за ідентифікаційними полями. База даних складається з 2 таблиць: users та ads.
7. Реалізовано 15 функцій, які виконують пошукові запити та запити на вставку, видалення та зміну даних в таблиці.
8. Описано покрокову роботу з програмним продуктом. Опис вказує, що веб-сервіс має простий та зрозумілий користувачу структурний інтерфейс, а також логічний порядок роботи з ним, що забезпечує швидке навчання роботи з сайтом.
9. Проаналізовано програмний код та визначено, що він не потребує спеціалізованого апаратного забезпечення, додаткових налаштувань та засобів для розгортання крім стандартних для більшості сучасних сайтів.
10. Проведено тестування веб-сервісу бюро знахідок. Виявлені помилки було виправлено. Сайт працює коректно та виконує всі поставлені задачі.

В подальшому даний веб-сервіс можна покращити шляхом оптимізації та додавання нових модулів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Петюшкин А. В. «HTML. Экспресс-курс», – СПб.: БХВ – Петербург, 2003,– 256 с.
2. Э. Харрис, PHP/MySQL для начинающих, пер. с англ. М.: КУДИЦ–ОБРАЗ, 2005, 384 с.
3. Шеллі Пауерс, Вивчаючи Javascript, М.: МетаТехнология, 2006, 240 с.
4. С. Стефанов, Шаблони Javascript. СПб.: БХВ, Петербург, 2013, 124 с.
5. Д. Герман, Ефективний Javascript. М.: БИНОМ, 2016, 236 с.
6. Довідниково-інформаційний портал htmlbook [Електроний ресурс]. Доступ за посиланням: http://htmlbook.ru/. [Дата звернення 20.03.2017 р.]
7. Современный учебник Javascript [Електроний ресур]. Доступ за посиланням: http://learn.javascript.ru. [Дата звернення 12.04.2017 р.]
8. Керівництво по PHP [Електроний ресурс]. Доступ за посиланням: http://php.net/manual. [Дата звернення 20.04.2017 р.]
9. Веб-сайт «Документація по PHP5» [Електроний ресурс]. Доступ за посиланням: http://www.php.su/php/?php/. [Дата звернення 25.05.2017 р.]
10. Э. Харрис, PHP/MySQL для начинающих, пер. с англ. М.: КУДИЦ–ОБРАЗ, 2005, 384 с.
11. М. Фаулер, К. Скотт. UML в кратком изложении. Применение стандартного языка объектного моделирования. М: «Мир», 1999, 191 с.
12. М. Гленфорд, Т. Баджетт, С. Кори, Искусство тестирования программ, 3-е изд. Москва: Дилектика, 2012, 345 с.
13. С.А. Мартишин, В.А. Симонов, М.В. Храпченко, Проектирование и реализация баз данных в СУБД MySQL c использованием MySQL Workbench, Учебное пособие. М.: ИНФРА-М, 2012, 160 с.
14. Мартин Фаулер. Шаблоны корпоративных приложений (Signature Series) = Patterns of Enterprise Application Architecture (Addison-Wesley Signature Series). — М.: «Вильямс», 2012. — 544 с.
15. 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.
16. Кузнецов Максим, Симдянов Игорь. PHP на примерах. — 2-е изд. перераб. и доп. — СПб.: «БХВ-Петербург», 2011. — С. 400
17. Кузнецов Максим, Симдянов Игорь. Объектно-ориентированное программирование на PHP. — СПб.: «БХВ-Петербург», 2007. — С. 608.
18. Стив Макконнелл. Совершенный код = Code complete. — СПб.: Питер, 2005. — С. 896.
19. 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.
20. Кузнецов Максим, Симдянов Игорь. PHP. Практика создания Web-сайтов. — 2-е изд. перераб. и доп. — СПб.: «БХВ-Петербург», 2008. — С. 1264.

# ДОДАТКИ

# ДОДАТОК А

Лістинг коду моделі користувача

<?php

class User extends Model

{

private static $roleUser = "1";

public function \_\_construct()

{

parent::\_\_construct();

}

public function addNewUser($dataUser)

{

$checkUser = User::checkRegisterUser($dataUser);

foreach($checkUser AS $error)

{

print $error."<br>";

}

if(empty($checkUser)) {

$userId = User::insertUser($dataUser);

$dataUser = $this->getUserById($userId);

$this->authUser($dataUser, false);

}

}

public static function checkRegisterUser($dataUser)

{

$err = array();

$login = $dataUser["login"];

$email = $dataUser["email"];

if(!preg\_match("/^[a-zA-Z0-9]+$/",$login)) {

$err[] = "Логин может состоять только из букв английского алфавита и цифр";

}

if(strlen($login) < 3 or strlen($login) > 12) {

$err[] = "Логин должен быть не меньше 3-х символов и не больше 30";

}

if($dataUser["password"] != $dataUser["confirm-password"])

$err[] = "Пароли не совпадают";

$checkLogin = self::$db->query("SELECT COUNT(id) FROM users WHERE login='$login'")->fetchColumn();

if($checkLogin > 0) {

$err[] = "Пользователь с таким логином уже существует в базе данных";

}

$checkEmail = self::$db->query("SELECT COUNT(id) FROM users WHERE email='$email'")->fetchColumn();

if($checkEmail > 0) {

$err[] = "Пользователь с такой электронной почтой уже существует в базе данных";

}

return $err;

}

public static function insertUser($dataUser)

{

$login = $dataUser["login"];

$password = md5(md5(trim($dataUser["password"])));

$email = $dataUser['email'];

$role = self::$roleUser;

$photo = "/template/images/no-image.jpg";

$query = "INSERT INTO users (id, login, password, email, id\_role, photo\_url) VALUES ('','$login','$password','$email','$role', '$photo')";

$sth = self::$db->exec($query);

return self::$db->lastInsertId();

}

public function authUser($dataUser, $checkUser = true)

{

if($checkUser) {

$dataUser = User::checkAuthUser($dataUser);

}

$\_SESSION["login"] = $dataUser["login"];

$\_SESSION["id"] = $dataUser["id"];

$\_SESSION["role"] = $dataUser["id\_role"];

$\_SESSION["USER"] = true;

}

public static function checkAuthUser($dataUser)

{

$login = $dataUser["login"];

$password = $dataUser["password"];

if (empty($login) or empty($password)) {

exit ("Вы ввели не всю информацию, вернитесь назад и заполните все поля!");

}

$login = stripslashes($login);

$login = htmlspecialchars($login);

$password = stripslashes($password);

$password = htmlspecialchars($password);

$login = trim($login);

$password = md5(md5(trim($password)));

$result = self::$db->query("SELECT \* FROM users WHERE login = '$login' OR email= '$login'");

$row = $result->fetch(PDO::FETCH\_ASSOC);

$basePassword = $row["password"];

if(empty($basePassword)) {

exit ("Извините, введённый вами login, email или пароль неверный.");

} else {

if($basePassword == $password)

{

return $row;

}

else {

exit ("Извините, введённый вами пароль неверный.");

}

}

}

public function getUserById($id)

{

$id = intval($id);

if ($id) {

$result = self::$db->query('SELECT \* FROM users WHERE id=' . $id);

$result->setFetchMode(PDO::FETCH\_ASSOC);

return $result->fetch();

}

}

public function editProfile($userId, $newInfoUser)

{

$query = "UPDATE users SET name = '".$newInfoUser['editFio']."', date\_birthday = '".$newInfoUser['editDateBirthday']."', phone = '".$newInfoUser['editPhone']."', address = '".$newInfoUser['editAddress']."' WHERE id = '$userId'";

$sth = self::$db->exec($query);

}

}

**ДОДАТОК Б**

Лістинг коду моделі оголошень

<?php

class Ads extends Model

{

private $statusDel = "4";

private $statusModeration = "2";

private $statusAdded = "1";

private $defaultImage = "/template/images/no-image.jpg";

public function \_\_construct()

{

parent::\_\_construct();

}

public function addNewAds($dataAd, $userId)

{

$name = $dataAd["adName"];

$type = $dataAd["type"];

$description = $dataAd['adDescription'];

$type\_category = $dataAd["adTypeCategory"];

$date\_lost\_found = $dataAd["adDate"];

$coordinates\_map = $dataAd["coordinates"];

$date\_create\_ad = date("Y-m-d");

$region = $dataAd["region"];

if(empty($dataAd["adUrlImage"])) {

$image\_url = $this->defaultImage;

} else {

$image\_url = $dataAd["adUrlImage"];

}

$query = self::$db->query('SELECT id FROM regions WHERE name\_region="' .$region.'"');

$id\_region = $query->fetchAll();

$id\_region = $id\_region[0]["id"];

$query = "INSERT INTO advert\_info\_place VALUES ('','$coordinates\_map','$id\_region')";

$sth = self::$db->exec($query);

$id\_place = self::$db->lastInsertId();

$query = "INSERT INTO ads VALUES ('','$name','$type','$description','$type\_category','$image\_url','$date\_lost\_found','$date\_create\_ad','$this->statusModeration','$userId', '$id\_place')";

$sth = self::$db->exec($query);

}

public function getUserAdsById($userId)

{

$userId = intval($userId);

if ($userId) {

$result = self::$db->query('SELECT \* FROM ads JOIN advert\_status on ads.id\_status = advert\_status.id WHERE id\_user=' . $userId.' AND id\_status <>'.$this->statusDel);

$result = $result->fetchAll();

return $result;

}

}

public function deletedAdvert($advertId, $userId)

{

$check = $this->checkUserIsAdvert($advertId, $userId);

if($check) {

Ads::changeStatusAdvert($advertId, $this->statusDel);

}

}

public function checkUserIsAdvert($advertId, $userId)

{

$result = self::$db->query('SELECT COUNT(\*) FROM ads WHERE id\_user=' . $userId.' AND id='.$advertId);

$result = $result->fetch();

if($result[0] != 0) {

return true;

} else {

return false;

}

}

public static function changeStatusAdvert($advertId, $status)

{

$query = "UPDATE ads SET id\_status = '$status' WHERE id = '$advertId'";

$sth = self::$db->exec($query);

}

public function getAdvertList($type = null, $status = '1', $filters = null, $search = null)

{

$checkType = Ads::checkTypeAdvert($type);

$filters = Ads::getQueryFilter($filters);

$search = Ads::getQuerySearch($search);

$result = self::$db->query('SELECT ads.\*, advert\_status.name\_status, regions.name\_region, advert\_type\_category.name\_type\_category FROM ads

JOIN advert\_status on ads.id\_status = advert\_status.id

JOIN advert\_info\_place on ads.id\_place = advert\_info\_place.id

JOIN regions on advert\_info\_place.id\_region = regions.id

JOIN advert\_type\_category on ads.id\_type\_category = advert\_type\_category.id

WHERE id\_status = '.$status.''.$checkType.''.$filters.''.$search.' ORDER BY ads.id DESC;');

$result = $result->fetchAll();

return $result;

}

public static function getQueryFilter($filters)

{

$filter = "";

if($filters != null) {

$filters = array\_slice($filters, 1);

foreach ($filters as $key => $value) {

if(!empty($filters[$key])) {

$filters[$key] = explode(",", $filters[$key]);

$filter .= " AND ( ";

foreach ($filters[$key] as $elemFiltr) {

$filter .= $key." = '$elemFiltr' OR ";

}

$filter = substr($filter, 0, -3);

$filter .= ") ";

}

}

}

return $filter;

}

public static function checkTypeAdvert($type)

{

if($type == "losses") {

return " AND id\_type = 2";

} elseif($type == "finds") {

return " AND id\_type = 1";

}

return "";

}

public function getAdvertById($id, $status = "1")

{

$id = intval($id);

if ($id) {

$statusString = ' AND id\_status = '.$status;

if($status == null) {

$statusString = '';

}

$result = self::$db->query('SELECT ads.\*, advert\_info\_place.coordinates\_map, regions.name\_region, advert\_type\_category.name\_type\_category FROM ads

JOIN advert\_info\_place on ads.id\_place = advert\_info\_place.id

JOIN regions on advert\_info\_place.id\_region = regions.id

JOIN advert\_type\_category on ads.id\_type\_category = advert\_type\_category.id

WHERE ads.id = '.$id.''.$statusString);

$result = $result->fetch(PDO::FETCH\_ASSOC);

return $result;

}

}

public static function getQuerySearch($searchText)

{

$search = "";

if(!empty($searchText)) {

$search = " AND ( name LIKE '%".$searchText['searchText']."%' OR description LIKE '%".$searchText['searchText']."%' ) ";

}

return $search;

}

public static function sendMessage($advertId)

{

$advretInfo = $this->getAdvertById($advertId);

}

public function checkStatusAdvert($advertId)

{

$result = self::$db->query('SELECT COUNT(\*) FROM ads WHERE id\_status=' . $this->statusAdded.' AND id='.$advertId);

$result = $result->fetch();

if($result[0] != 0) {

return true;

} else {

return false;

}

}

public function updateAdvert($newInfoAdvert)

{

$name = $newInfoAdvert["adName"];

$type = $newInfoAdvert["type"];

$description = $newInfoAdvert['adDescription'];

$type\_category = $newInfoAdvert["adTypeCategory"];

$image = $newInfoAdvert['adUrlImage'];

$date\_lost\_found = $newInfoAdvert["adDate"];

$coordinates\_map = $newInfoAdvert["coordinates"];

$date\_create\_ad = date("Y-m-d");

$region = $newInfoAdvert["region"];

$id\_advert = $newInfoAdvert['id'];

$id\_place = $newInfoAdvert['id\_place'];

$query = self::$db->query('SELECT id FROM regions WHERE name\_region="' .$region.'"');

$id\_region = $query->fetchAll();

$id\_region = $id\_region[0]["id"];

$query = "UPDATE advert\_info\_place SET coordinates\_map = '".$coordinates\_map."', id\_region = '".$id\_region."' WHERE id = '".$id\_place."'";

$sth = self::$db->exec($query);

$query = "UPDATE ads SET name = '".$name."', id\_type = '".$type."', description = '".$description."', id\_type\_category = '".$type\_category."', image\_url = '".$image."', date\_lost\_found = '".$date\_lost\_found."', date\_create\_ad = '".$date\_create\_ad."', id\_status = '".$this->statusModeration."' WHERE id = '".$id\_advert."'";

$sth = self::$db->exec($query);

}

}

# ДОДАТОК В

Лістинг коду моделі модератора

<?php

class Moderator extends Model

{

private static $roleModerator = "2";

private static $advertStatusModeration = "2";

private static $advertStatusAdded = "1";

private static $advertStatusRejected = "3";

public function \_\_construct()

{

parent::\_\_construct();

}

public function viewAdverts($userId)

{

if(isset($userId)) {

$checkRole = $this->checkRole($userId);

if($checkRole) {

return Moderator::createArrayUsersAdverts();

} else {

return false;

}

} else {

return false;

}

}

public function checkRole($userId)

{

$result = self::$db->query('SELECT COUNT(\*) FROM users WHERE id=' . $userId.' AND id\_role='.self::$roleModerator);

$result = $result->fetch();

if($result[0] != 0) {

return true;

} else {

return false;

}

}

public static function createArrayUsersAdverts()

{

$modelAds = new Ads();

$modelUser = new User();

$advertList = $modelAds->getAdvertList(null, self::$advertStatusModeration);

for($i = 0; $i < count($advertList); $i++) {

$userInfo = $modelUser->getUserById($advertList[$i]["id\_user"]);

$advertList[$i]["login\_user"] = $userInfo["login"];

}

return $advertList;

}

public function moderation($statusId)

{

$status = "";

$statusId = explode("-", $statusId);

$advertId = $statusId[1];

if($statusId[0] == "accept") {

$status = self::$advertStatusAdded;

} else {

$status = self::$advertStatusRejected;

}

Ads::changeStatusAdvert($advertId, $status);

}

}