МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

„КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

## КУРСОВА РОБОТА

з дисципліни «Об’єктно-орієнтоване програмування»   
на тему

## Шаблони проєктування в ООП. Електронна дошка оголошень

Виконав студент

ІІ курсу групи КП-03  
Довженко Роман Олександрович залікова книжка КП-0302

Керівник роботи

доцент, к.т.н. Заболотня Т.М. Оцінка

(дата, підпис)

КИЇВ 2022

## ЗМІСТ

**ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ ТА ТЕРМІНІВ** 3

## ВСТУП 4

## СТРУКТУРНО-АЛГОРИТМІЧНА ОРГАНІЗАЦІЯ ПРОГРАМНОЇ СИСТЕМИ 6

* 1. Модульна організація програми 6
  2. Функціональні характеристики 7

## ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ЗА ДОПОМОГОЮ ШАБЛОНІВ ПРОЄКТУВАННЯ 8

* 1. Обґрунтування вибору та опис шаблонів проєктування для

програмної реалізації електронної дошки оголошень 8

* 1. Діаграма класів 16
  2. Опис результатів роботи програми 17

## ВИСНОВКИ 26

## СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 27

## ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ ТА ТЕРМІНІВ

|  |  |
| --- | --- |
| TCP - | (англ. Transmission Control Protocol) (протокол передавання ); |
| Web - | (англ. World Wide Web) Глобальний інформаційний простір, заснований на фізичній інфраструктурі Інтернету і протоколі передачі даних HTTP |
| RPC - | (англ. Remote Procedure Call) Клас технологій, що дозволяють програмам викликати функції чи процедури у іншому адресному просторі. |
| URL - | (англ. Uniform Resource Locator) Уніфікований локатор ресурсів або адреса ресурсу. Стандартизована адреса певного ресурсу (такого як документ, чи зображення) в інтернеті (чи деінде) |
| IP - | (від англ. Internet Protocol) Протокол мережевого рівня для передавання датаграм між мережами. |
| XML - | (від англ. eXtensible Markup Language) стандарт побудови мов розмітки ієрархічно структурованих даних для обміну між різними застосунками, зокрема, через Інтернет. |
| XML-RPC - | (від англ. eXtensible Markup Language Remote Procedure Call) стандарт/протокол виклику віддалених процедур, що використовує XML для кодування своїх повідомлень та HTTP як транспортний механізм. |
| CRUD - | акронім, що позначає чотири базові функції, що використовуються під час роботи з базами даних: створення (англ. create), читання (англ. read), модифікація (англ. update), видалення (англ. delete). |

## ВСТУП

Дана курсова робота присвячена розробці програмного забезпечення електронної дошки оголошень за допомогою використання шаблонів проєктування. В процесі продажу певних речей часто виникає необхідність спеціально обладнаного програмного забезпечення.

Проблемами місцевого продажу є низька швидкість та ефективність. Виникає необхідність створення програмного забезпечення, яке б з найбільш можливою швидкістю та охопленням поширювало оголошення. Саме тому пропонується розробка програмних засобів електронної дошки оголошень.

Завдяки тому, що розроблений комплекс програм використовує компільовану мову програмування Java, створений додаток може працювати на будь-якій комп'ютерній архітектурі, для якої існує реалізація віртуальної машини Java. Тому розроблене програмне забезпечення є дійсно актуальним. Дана тематика обрана для виконання курсової роботи тому, що результати абстрагування об’єктів у цій предметній галузі дозволяють застосувати вивчені принципи та методи об’єктно‑орієнтованого програмування для створення програмного забезпечення, зокрема шаблони проєктування.

*Об’єктом* дослідження є процес створення серверної та клієнтської частини для додатка електронної дошки оголошень з використанням шаблонів проєктування.

*Метою роботи* є розроблення програмного забезпечення електронної дошки оголошень з використанням шаблонів проєктування.

Для досягнення визначеної мети необхідно виконати такі *завдання*:

* абстрагувати об’єкти предметної галузі;
* розробити структурну організацію ПЗ за допомогою застосування основних принципів ООП та шаблонів проєктування;
* визначити та описати функціональні характеристики програми;
* обґрунтувати вибір шаблонів проєктування, використаних для побудови програми;
* розробити дизайн та його логіку;
* виконати реалізацію програмного забезпечення відповідно до вимог технічного завдання;
* виконати тестування розробленої програми;
* оформити документацію з курсової роботи.

Розроблене ПЗ електронної дошки оголошень складається з декількох модулів: модуль мережевої взаємодії, клієнтський модуль, модуль для роботи з базами даних, серверний модуль, а також окремо модулі для кожної складової.

Реалізовані шаблони проєктування: Одинак, Віддалений Заступник, Адаптер, Спостерігач, Модель-Вид-Контролер, Посередник.

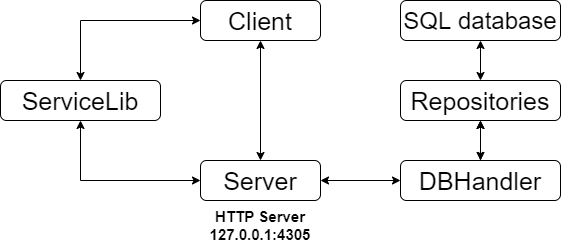
До функціональних можливостей програми належать: створення, редагування та перегляд оголошень, створення та редагування користувача.

Для функціонування розробленої програми необхідно забезпечити наявність на комп’ютері 300 Мбайт вільного дискового простору + 1Гбайт для кешу та встановлених певних додаткових бібліотек: com.apache.xmlrpc, mysql-connector-java-8.0.29 та JDK 1.6 і вище.

Розроблене програмне забезпечення може бути використане для онлайн купівлі/продажу товарів та послуг як приватним особам, так і представникам бізнесу.

Пояснювальна записка складається зі вступу, трьох розділів, загальних висновків, списку використаних джерел (4-ти найменувань, з них 4 – іноземною мовою). Робота містить 18 рисунок. Загальний обсяг роботи – 27 друкованих сторінки, з них 24 сторінок основного тексту та 1 сторінка списку використаних джерел.

1. **ОПИС СТРУКТУРНО-ЛОГІЧНОЇ СХЕМИ ПРОГРАМИ**
   1. **Модульна організація програми**



*Рис. 1.1.1. Модульна організація програми*

У програмі можна виділити декілька основних логічних модулів, що

мають певну самостійність і обмінюються один з одним даними. Модулям

необов’язково відповідають конкретні класи чи ієрархії класів. Деякі з них містять вкладені модулі.

Модуль **Client** реалізує головні функції програми. Він зберігає всередині класи контролери, класи моделі, класи представлень, клас клієнта, медіатор.

Кожен клас контролер є екземпляром класу графічного представлення fxml файла, в якому є метод *initialize*. Робота модуля клієнта починається з статичного методу main. В класі Main через клас медіатор встановлюється та відкривається вікно аутентифікації.

Модуль **Server** відкриває сокет для підключення клієнта, при підключенні клієнта відкриває новий потік і чекає наступного. При ініціалізації сервера створюється обробник запитів до бази даних і серверний сервіс для отримання запитів від клієнта та відправлення відповідей.

Модуль **ServiceLib** містить необхідні для функціонування клієнта та сервера інтерфейси, об’єкти, класи обробники, контролери та сервіси.

Модуль **DBHandler** є підмодулем модуля Server. Цей модуль містить модуль с репозиторіями, сам обробник і клас конфігурації підключення.

Модуль **Repositories** є підмодулем модуля DBHandler. Цей модуль містить класи репозиторії для виконання CRUD функцій з таблицями бази даних та клас запитів.

* 1. **Функціональні характеристики**

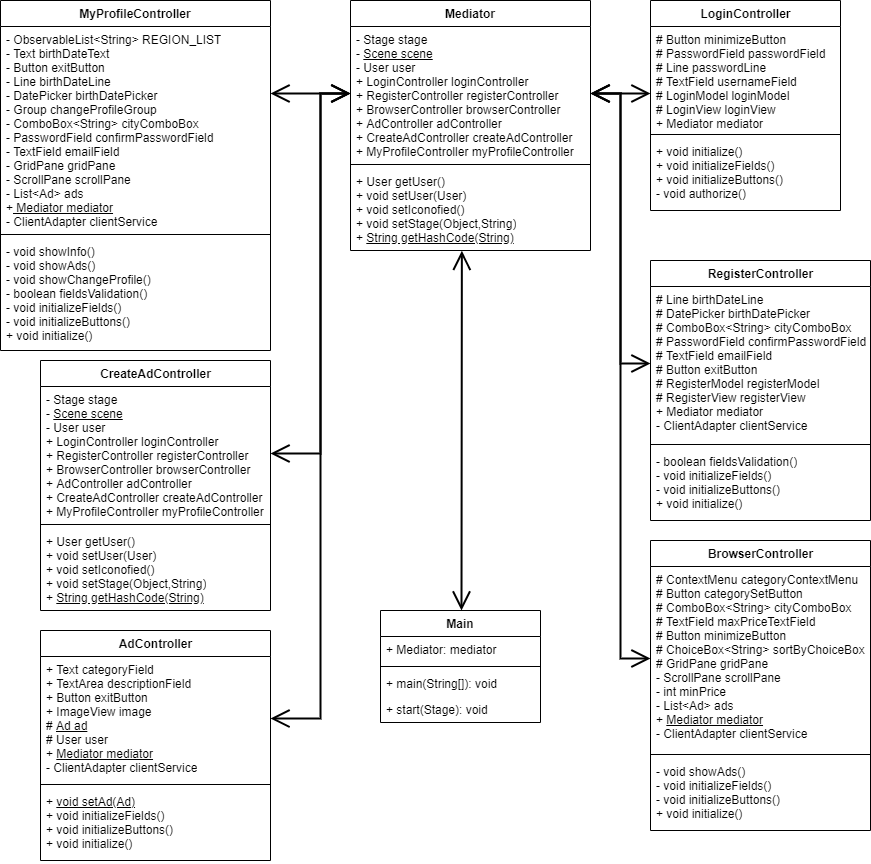
Система надає такі можливості як:

* Реєстрація користувача
* Вхід користувача до системи
* Перегляд наявних оголошень
* Створення оголошення
* Пошук оголошень
* Фільтрація оголошень
* Видалення оголошення
* Редагування оголошення
* Редагування профіля

1. **ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ЗА ДОПОМОГОЮ ШАБЛОНІВ ПРОЄКТУВАННЯ** 
   1. **Обґрунтування вибору та опис шаблонів проєктування** **для програмної реалізації електронної дошки оголошень**
2. **Посередник**

Посередник (англ. Mediator) - поведінковий шаблон проєктування, що забезпечує взаємодію безлічі об'єктів, формуючи при цьому слабкий зв’язок і позбавляє об'єкти необхідності явно посилатися один на одного. Діаграма класів представлена на рис. 2.1.1.

.



*Рис. 2.1.1. Діаграма класів, які входять до шаблону «Посередник»*

*Учасники шаблону:*

* **Mediator** (Посередник) визначає інтерфейс для взаємодії та обміну інформацією з компонентами. Містить код взаємодії кількох компонентів між собою. Цей об’єкт не тільки зберігає посилання на всі свої компоненти, але й сам їх створює, керуючи подальшим життєвим циклом.
* **Component** (Компонент) класи MyProfileController, LoginController, CreateAdController, RegisterController, AdController, BrowserController, Main. Компоненти — це об’єкти, що містять бізнес-логіку програми. Кожен компонент має посилання на об’єкт посередника. Завдяки цьому компоненти можна повторно використовувати в інших програмах, зв’язавши їх з посередником іншого типу. Компоненти не повинні спілкуватися один з одним безпосередньо. Якщо в компоненті відбувається певна подія, він повинен викликати метод медіатора.

*Результати використання конструкції:*

* Створення централізованого способу зміни вікна додатку;
* Усунення залежності між класами контролерами, що дає змогу використовувати їх повторно;
* Спрощення взаємодії між компонентами;

*Обґрунтування використання шаблону:*

Для зменшення взаємопов'язаності безлічі класів контролерів, зв'язки між якими складні та заплутані було використано шаблон Посередник. Використання шаблону Посередник необхідно для централізованого способу зміни вікна додатку, згортання вікна додатка.

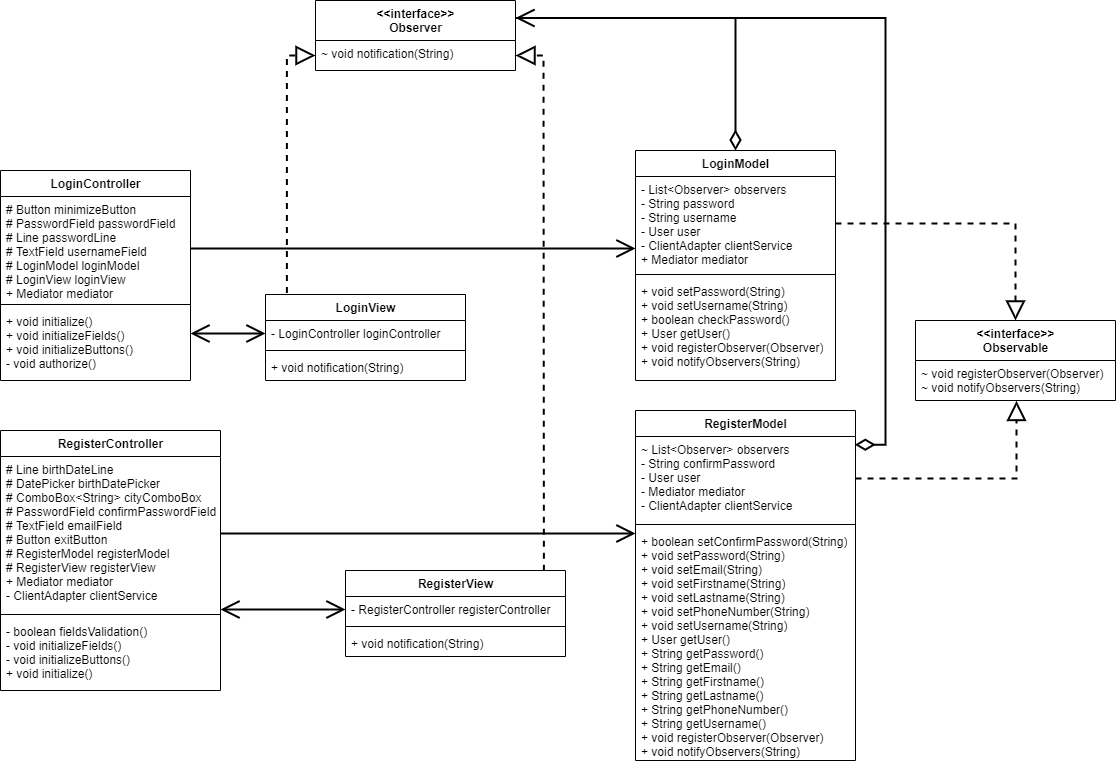
* 1. **Модель – Вид – Контролер + Спостерігач**

Модель – Вид – Контролер (англ. MVC) - Шаблон поділу даних програми та керуючої логіки на три окремі компоненти: модель, вид та контролер.

* Модель (Model) надає дані і реагує команди контролера, змінюючи свій стан.
* Вид (View) відповідає за відображення даних моделі користувачеві, реагуючи зміни моделі (завдяки використанню шаблону Спостерігач).
* Контролер (Controller) інтерпретує дії користувача, сповіщаючи модель необхідність змін.

Оскільки створений додаток використовує JavaFX. Контролер жорстко зав'язаний на форму, тому саме тут ми отримуємо посилання на графічні елементи вікна. JavaFX додаток будується трохи специфічно. Коли програма приймає fxml-файл, його контролер (вказаний у fxml-файлі) має відношення лише до цієї формі. Тобто це не Контролер програми, а Контролер конкретної форми. Компоненту Вид відводиться роль «автоматичного відображення» результату Моделі. Дана схема описана патерном "Спостерігач" (Observer).

Спостерігач — це поведінковий патерн проєктування, який створює механізм підписки, що дає змогу одним об’єктам стежити й реагувати на події, які відбуваються в інших об’єктах.

Модель в MVC представлені є Видавцем, тобто компонент Вид підписується на Модель і при зміні свого стану Модель проходиться по списку підписників і інформує їх. Діаграма класів представлена на рис. 2.1.2. **

*Рис. 2.1.2. Діаграма класів, які входять до шаблонів «Модель – Вид – Контролер + Спостерігач»*

*Учасники шаблону “MVC“:*

* **Model** (LoginModel, RegisterModel) – Модель надає дані та методи роботи з ними. Модель не залежить від подання (не знає як дані візуалізувати) та контролера (не має точок взаємодії з користувачем), просто надаючи доступ до даних та управління ними. Модель побудована таким чином, що при зміні свого стану інформує всіх підписників;
* **Controller** (LoginController, RegisterController) забезпечує «зв'язок» між користувачем та системою, містить графічні елементи вікна. Контролює та спрямовує дані від користувача до системи та навпаки. Використовує модель та подання для реалізації необхідної дії;
* **View** (LoginView, RegisterView) – відповідає за візуалізацію даних користувачу. Не обробляє введені дані користувача.

*Учасники шаблону “Спостерігач“:*

* **Observer** (LoginView, RegisterView) - визначає інтерфейс, яким користується видавець для надсилання сповіщень. Виконує у відповідь на сповіщення, яке надійшло від видавця певні дії з графічними елементами вікна. Ці класи мають дотримуються загального інтерфейсу, щоб видавець не залежав від конкретних класів підписників.
* **Observable** (LoginModel, RegisteModel) - володіє внутрішнім станом, зміни якого цікаво відслідковувати підписникам. Містить механізм підписки: список підписників та методи підписки/відписки. Підписники зберігаються у змінній observers. Повідомлення розсилається за допомогою notifyObservers() з параметром

*Результати використання конструкції:*

* Результатом використання архітектурного шаблону MVC є розділення даних і бізнес логіки від візуалізації;
* Збільшена можливість повторного використання коду;
* Використання шаблону Спостерігач прибрало залежності між компонентами шаблону Модель-Вид-Контролер;
* Між Моделлю та Представленням забезпечений динамічний односторонній зв’язок

*Обґрунтування використання шаблону:*

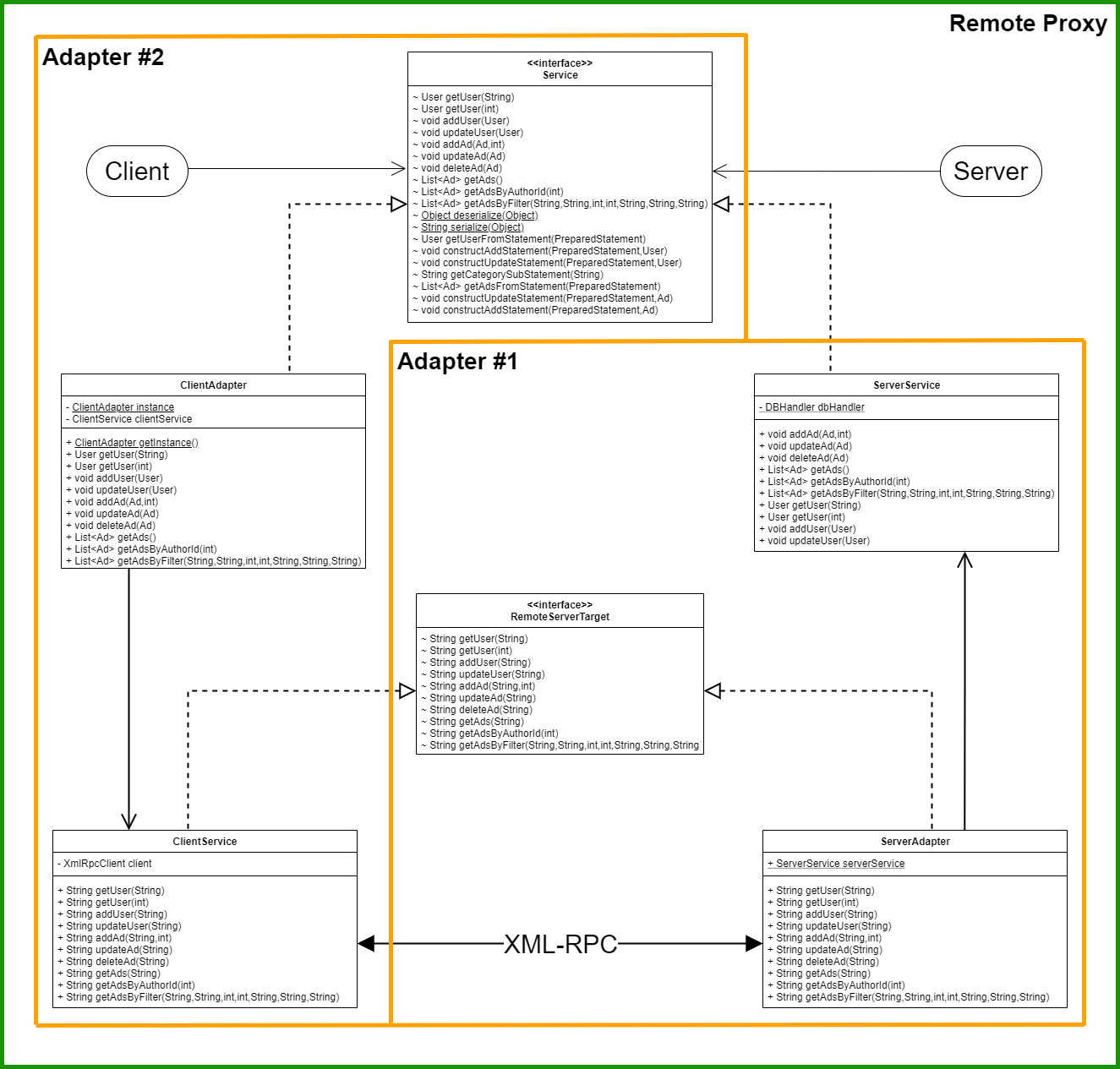
Для розроблення був використаний архітектурний шаблон Модель – Вид – Контролер. Використання цього шаблону необхідно для того, щоб розділити логічні частини додатка і створювати їх окремо один від одного, тобто створювати незалежні блоки коду, котрі можна будь-як змінювати, не торкаючись інших.

Оскільки відповідно шаблону Модель – Вид – Контролер Модель при зміні свого стану має повідомити про це своє Представлення був використаний один з ефективних способів досягнення такої функції – шаблон Спостерігач.

* 1. **Адаптер + Віддалений Заступник**

Шаблон Віддалений Заступник(англ. Remote Proxy) надає локальний об'єкт, який посилається на реальний об'єкт в іншому місці, через мережеве з'єднання. Заступник виконує необхідні дії для кодування запитів на передачу мережі і приймає результати від віддаленого ресурсу.

Адаптер — це структурний шаблон проєктування, що дає змогу об’єктам із несумісними інтерфейсами працювати разом.

****

*Рис. 2.1.3. Діаграма класів, які входять до шаблонів «Адаптер + Віддалений Заступник»*

*Учасники шаблону “Віддалений заступник”*:

* **ServiceInterface** (Service) – Інтерфейс сервісу визначає загальний інтерфейс для сервісу й замісника.
* **Service** (ClientAdapter) – реалізація сервісного інтерфейсу на клієнтській частині програми. Містить корисну бізнес – логіку. Є адаптером, містить посилання на сервіс призначений для роботи з серверною частиною;
* **Proxy** (ServerService) – реалізація сервісного інтерфейсу на серверній частині програми. Містить адаптер, необхідний для правильного функціонування віддаленого виклику процедур та повернення даних до клієнтської частини ;

*Учасники шаблону “Адаптер”*:

* **Client Interface** (Service, RemoteServerTarget) – описує протокол, через який клієнт може працювати з іншими класами.
* **Adapter** (ClientAdapter, ServerAdapter) – може одночасно працювати і з клієнтом/сервером, і з сервісом. Він реалізує клієнтський інтерфейс і містить посилання на об’єкт сервісу. Адаптер отримує виклики від клієнта через методи клієнтського інтерфейсу, а потім конвертує їх у виклики методів загорнутого об’єкта в потрібному форматі.
* **Service** (ClientService, ServerService) – клас що має незрозумілий для Клієнта інтерфейс, тому не може використовувати цей клас безпосередньо.

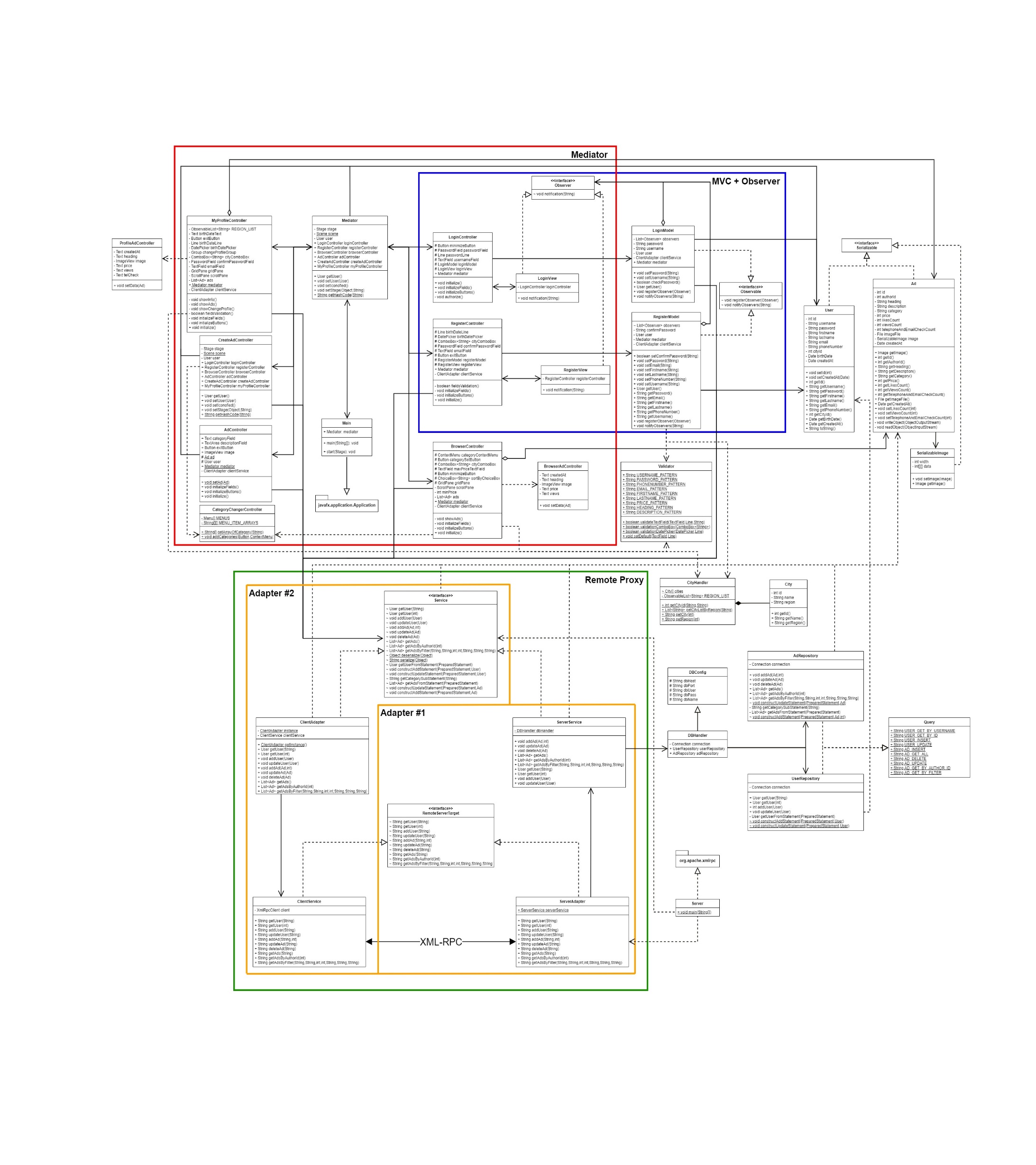
*Результати використання конструкції:*

* Шаблон Адаптер дозволив створити об’єкт-прокладку, який перетворюватиме виклики програми у формат, зрозумілий сторонньому класу;
* Використання шаблону Віддалений Заступник дозволило створити клієнт-серверну архітектуру;
* Завдяки однаковому інтерфейсу об’єкт-замісник можна передати до будь-якого коду, що очікує на сервісний об’єкт;

*Обґрунтування використання шаблону:*

При створенні клієнт-серверної архітектури з’явилась необхідність використати шаблон Віддалений Заступник, оскільки клієнтська і серверна фактично використовують один інтерфейс. Для створення клієнт-серверної архітектури було використано сторонню бібліотеку для XML-RPC зв’язку, а саме org.apache.xmlrpc. Виклик методів на сервері в цій бібліотеці можливий за умов, що методи на серверній частині приймають в якості параметрів лише примітиви, а повертають лише дані в вигляді рядка. При вже створеному інтерфейсу є очевидним те, що здійснити віддалений виклик методів просто неможливий. Саме тому було двічі використано структурний шаблон Адаптер. Перший Адаптер реалізує інтерфейс Service, що є одночасно інтерфейсом для шаблону Віддалений Заступник, і містить в собі посилання на ClientService, що реалізує інтерфейс RemoteServerTarget. При виклику певного методу в Адаптері клієнтської частини викликається метод в клієнтському сервісі, що вже безпосередньо надсилає запит до серверної частини. Серверна частина містить вже свій адаптер, що реалізує інтерфейс RemoteServerTarget і містить посилання на клас, що реалізує звичний для клієнта інтерфейс Service. Після виклику в сервісному класі необхідного методу повертається певний об’єкт на адаптер, який в свою чергу передає відповідь на клієнтський сервіс, з якого, за допомогою адаптера, клієнт отримує необхідні дані.

* 1. **Діаграма класів**



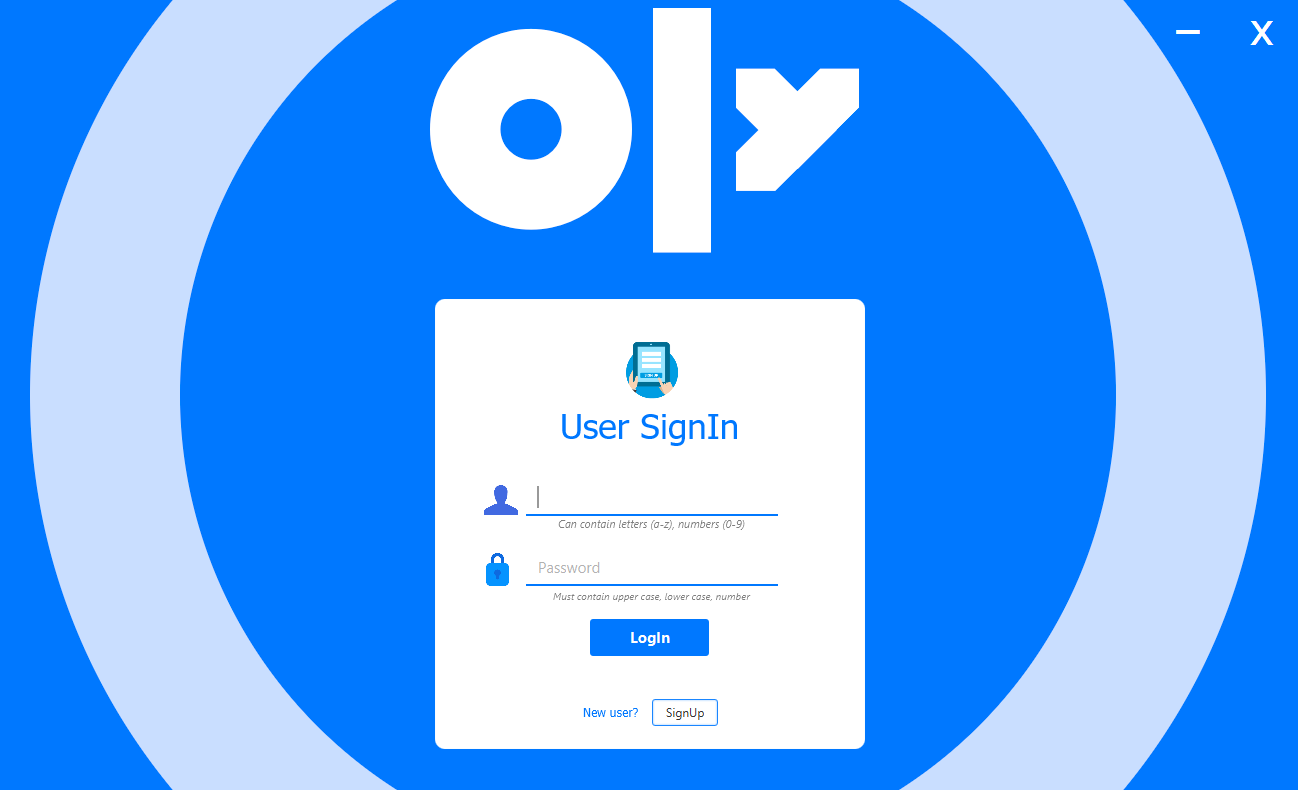
*Рис. 2.2. Діаграма класів електронної дошки оголошень*

* 1. **Опис результатів роботи програми**

Для того щоб запустити додаток спершу потрібно запустити сервер.

Після запуску серверу можна запустити клієнт. При запуску клієнта користувача вітає вікно аутентифікації, де користувач може увійти за вже створеним профілем, чи створити новий натиснувши на кнопку реєстрації.

Вікно аутентифікації подано на рис. 2.3.1.



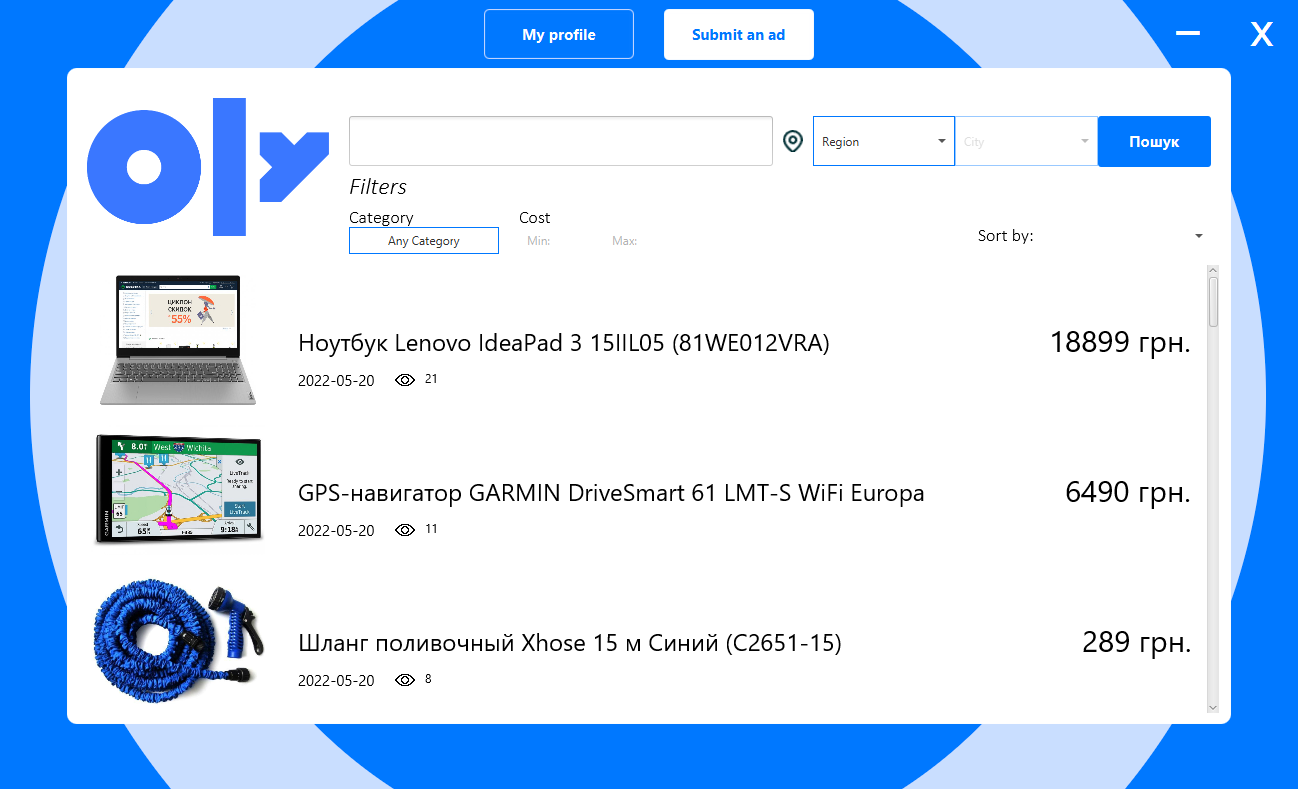
*Рис. 2.3.1. Вікно аутентифікації*

При натисканні кнопки реєстрації користувачу стає доступне вікно реєстрації (рис.2.3.2), де користувачу пропонується ввести необхідні для реєстрації дані. Варто зауважити, що всі полі даних мають перевірку на правильність, і для поля імені користувача – на унікальність.

Після введення даних користувач може натиснути кнопку реєстрації, чи натиснути кнопку повернення до аутентифікації в противному випадку. При вдалій реєстрації користувача знову перекине до вікна аутентифікації. Поля вікна аутентифікації також мають перевірку на правильність введених даних.

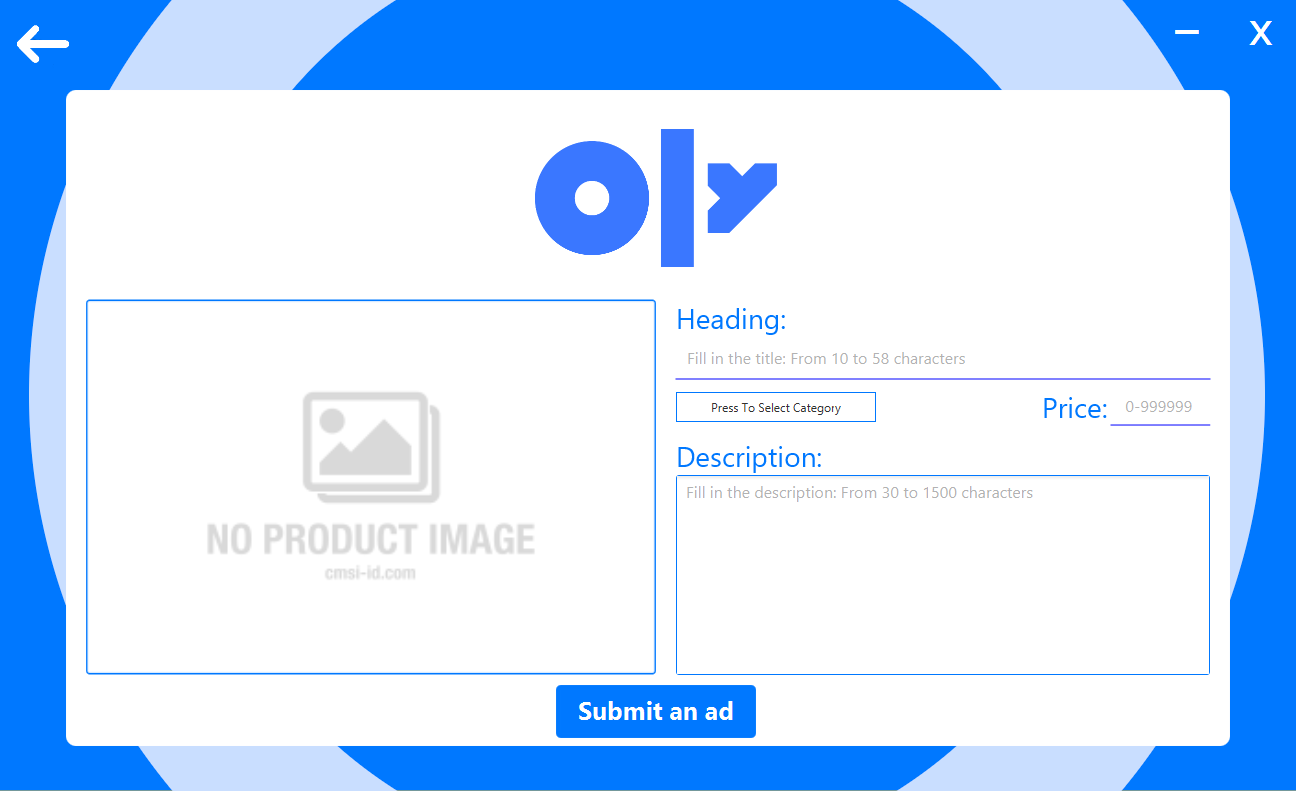


*Рис. 2.3.2. Вікно реєстрації*

При успішній аутентифікації користувач перейде до головного вікна (далі “браузер”) (рис. 2.3.3). Браузер містить список оголошень, кнопку переходу до вікна профілю, кнопку створення оголошення, кнопку пошуку, поле пошуку, полі для задання діапазону ціни, кнопку вибору категорії, випадаючий список для вибору способу сортування, випадаючі списки для вибору області та міста. 

*Рис. 2.3.3. Головне вікно*

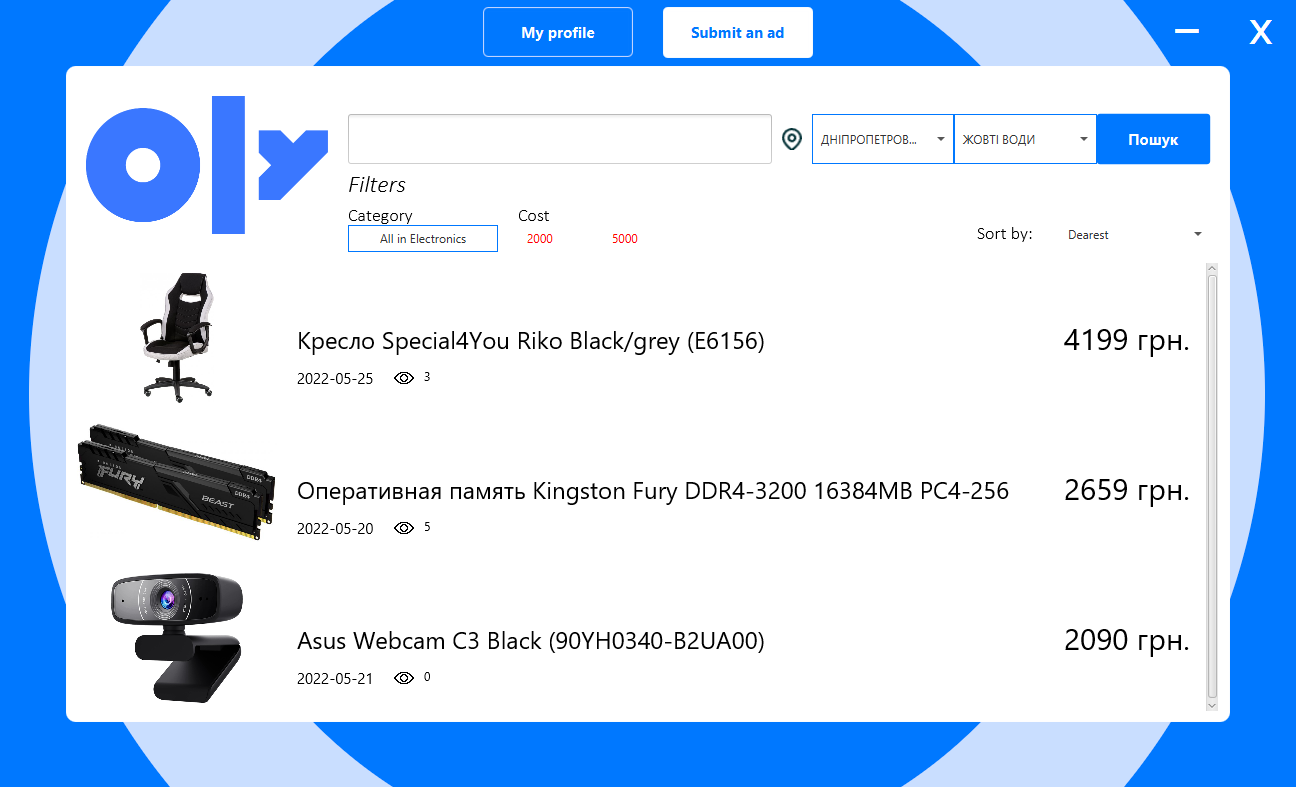
Кнопка створення нового оголошення відкриває нове вікно (рис. 2.3.4), де користувач може додати зображення, написати заголовок оголошення, обрати категорію, задати ціну та написати опис.



*Рис. 2.3.4. Вікно створення оголошення*

Всі полі містять перевірку на правильність введених даних. Діалогове вікно вибору зображення не дозволить користувачу обрати щось інше, окрім зображення. Після введення даних користувач може натиснути кнопку подання оголошення, як наслідок відкриється вікно браузеру, до списку оголошень буде додано нове оголошення.

Кнопка пошуку виконає пошук за вказаними фільтрами та сортує список оголошень за відповідним способом. (рис. 2.3.5)

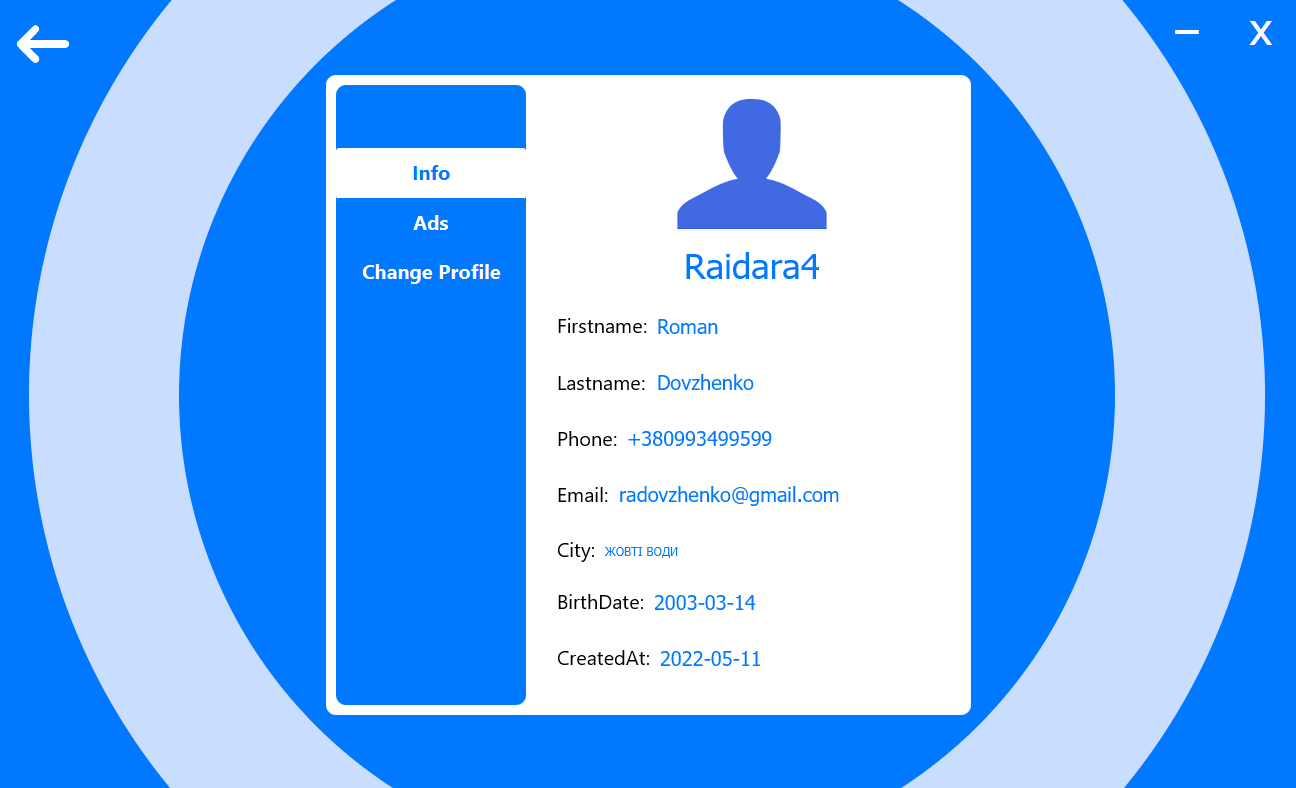


*Рис. 2.3.5. Приклад пошуку*

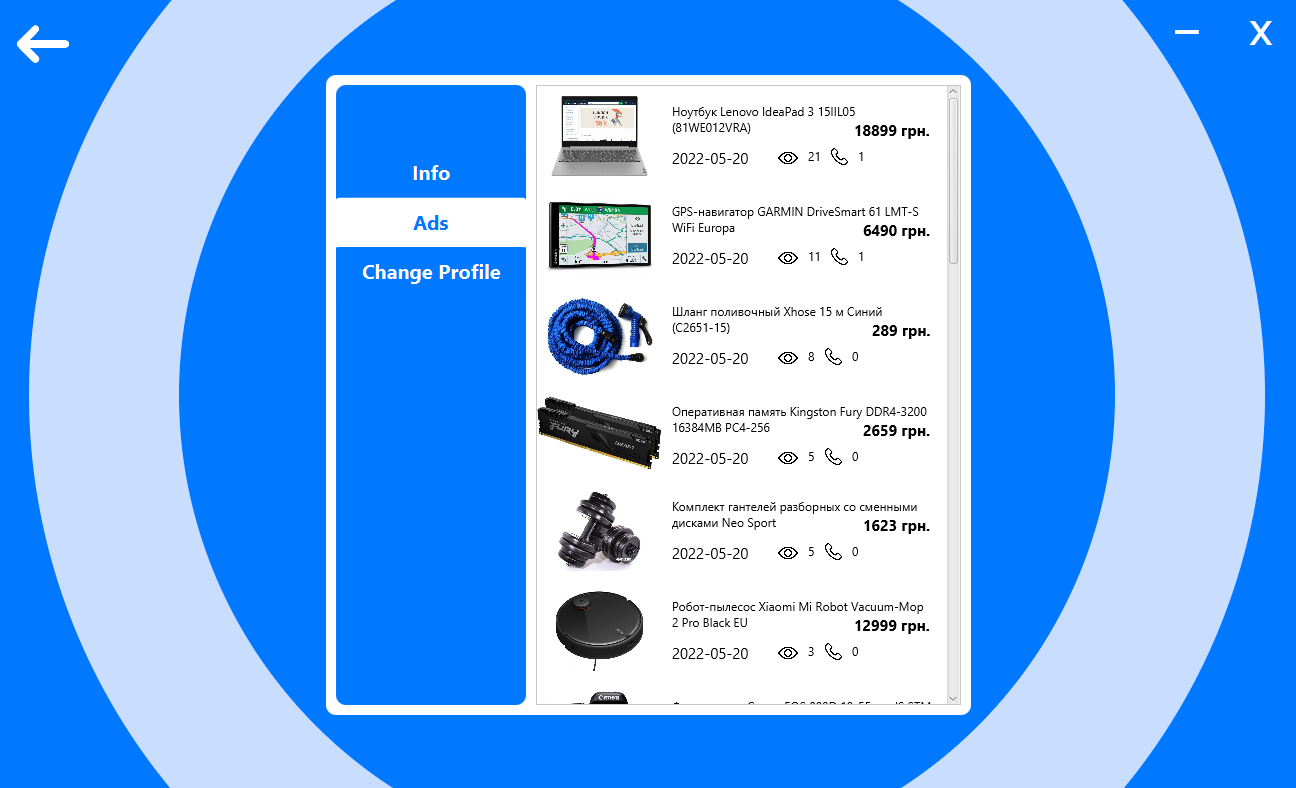
Кнопка “Мій профіль” відкриє користувачу нове вікно, що містить 3 вкладки: s

Вкладка №1 – персональні дані користувача. (рис. 2.3.6)

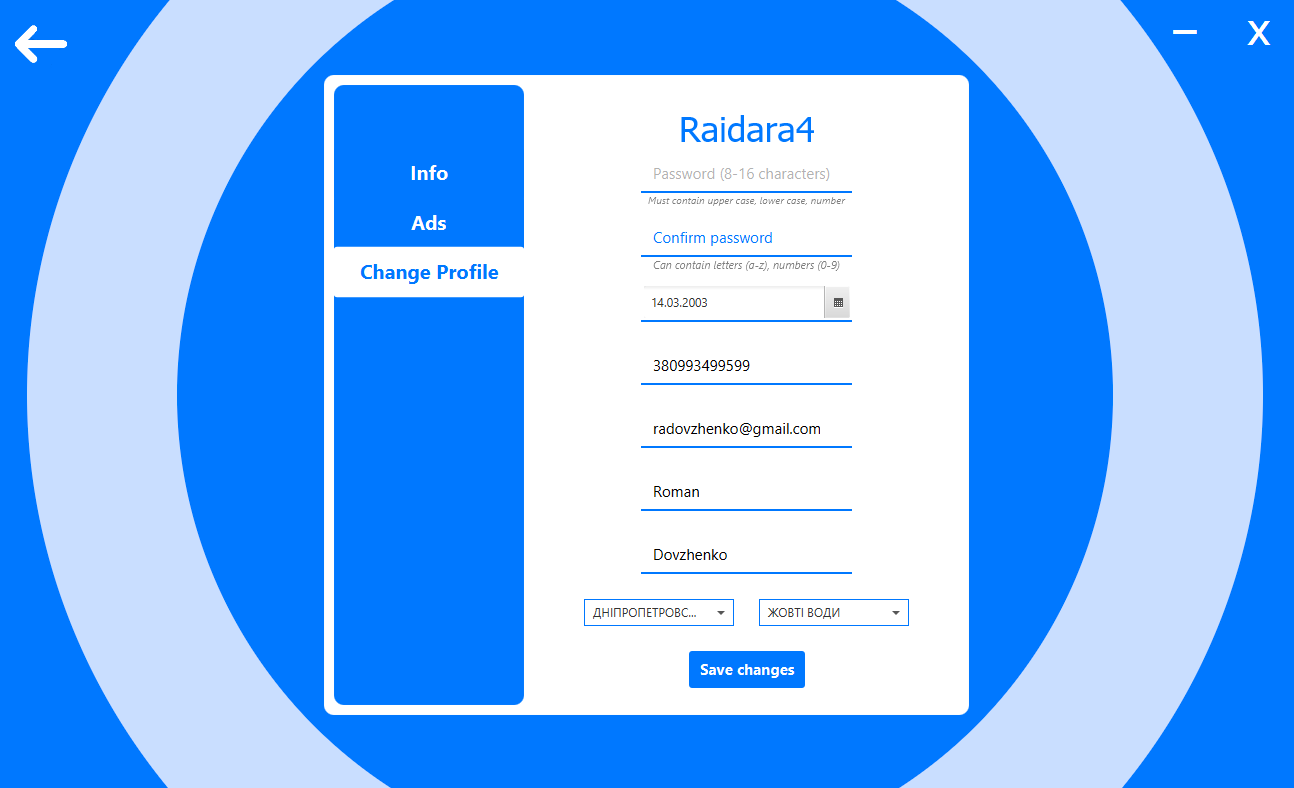
Вкладка №2 – перегляд списку оголошень користувача. (рис. 2.3.7)

Вкладка №3 – режим зміни даних профілю користувача, де кожне поле після введення даних проходить перевірку. Містить кнопку підтвердження зміни. (рис. 2.3.8) 

*Рис. 2.3.6. Персональні дані*

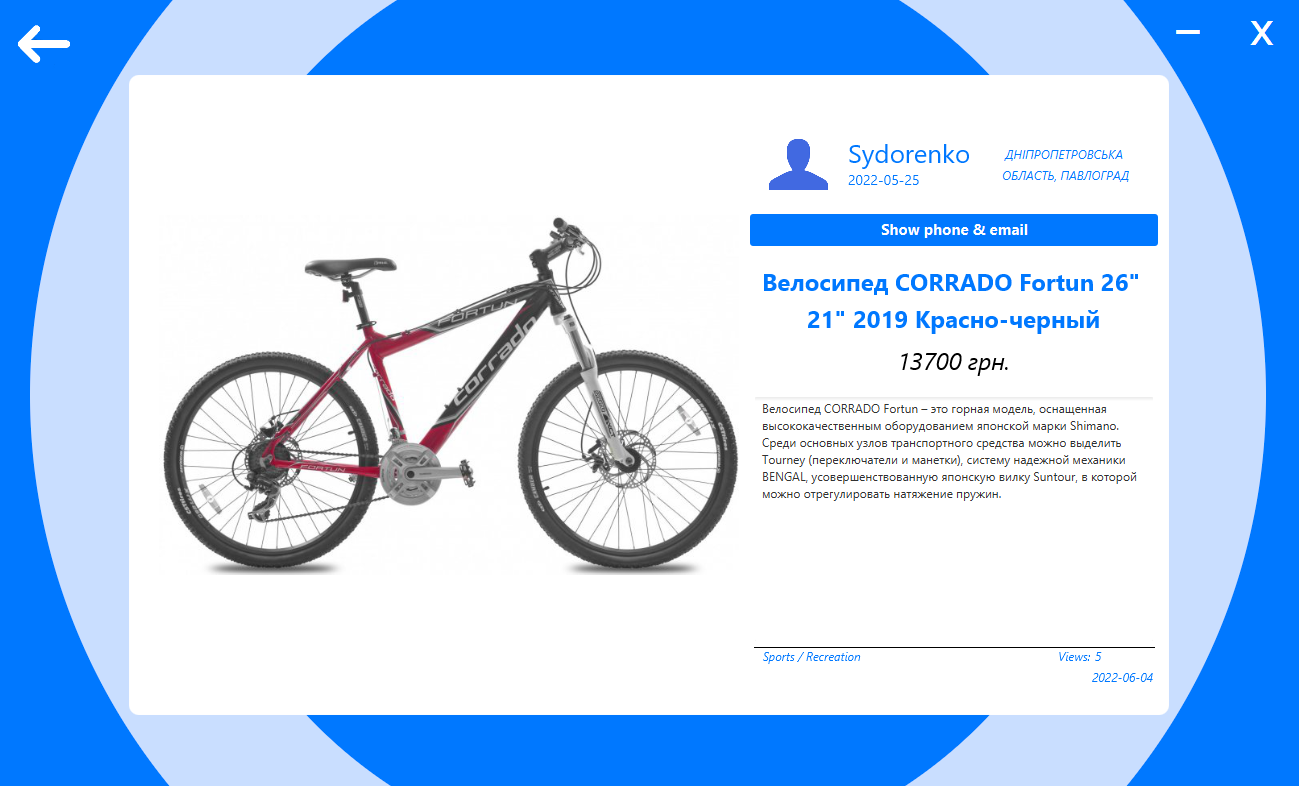


*Рис. 2.3.7. Список оголошень користувача*

**

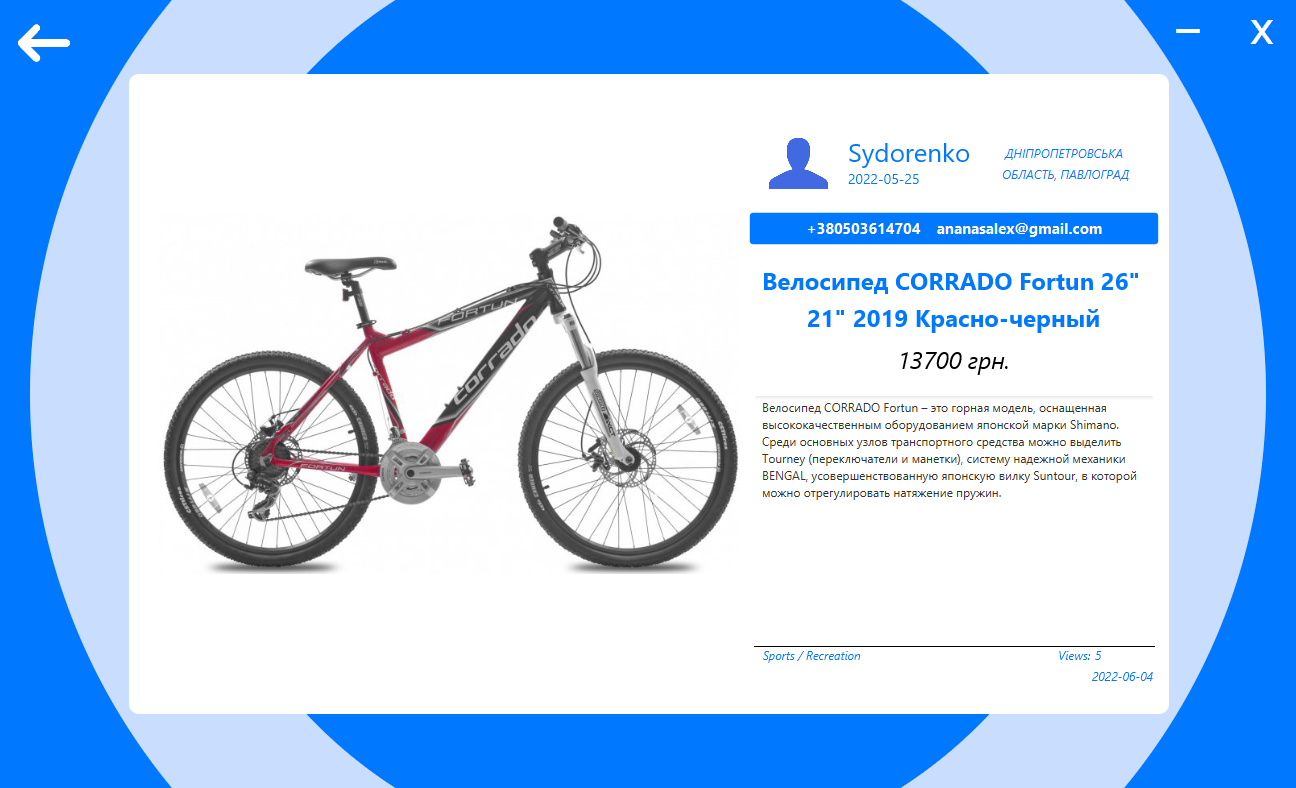
*Рис. 2.3.8. Зміна профілю*

При натисненні на оголошення у головному вікні чи вікні перегляду власник оголошень відкриється вікно демонстрації оголошення (рис 2.3.9)

**

*Рис. 2.3.9. Вікно демонстрації оголошення*

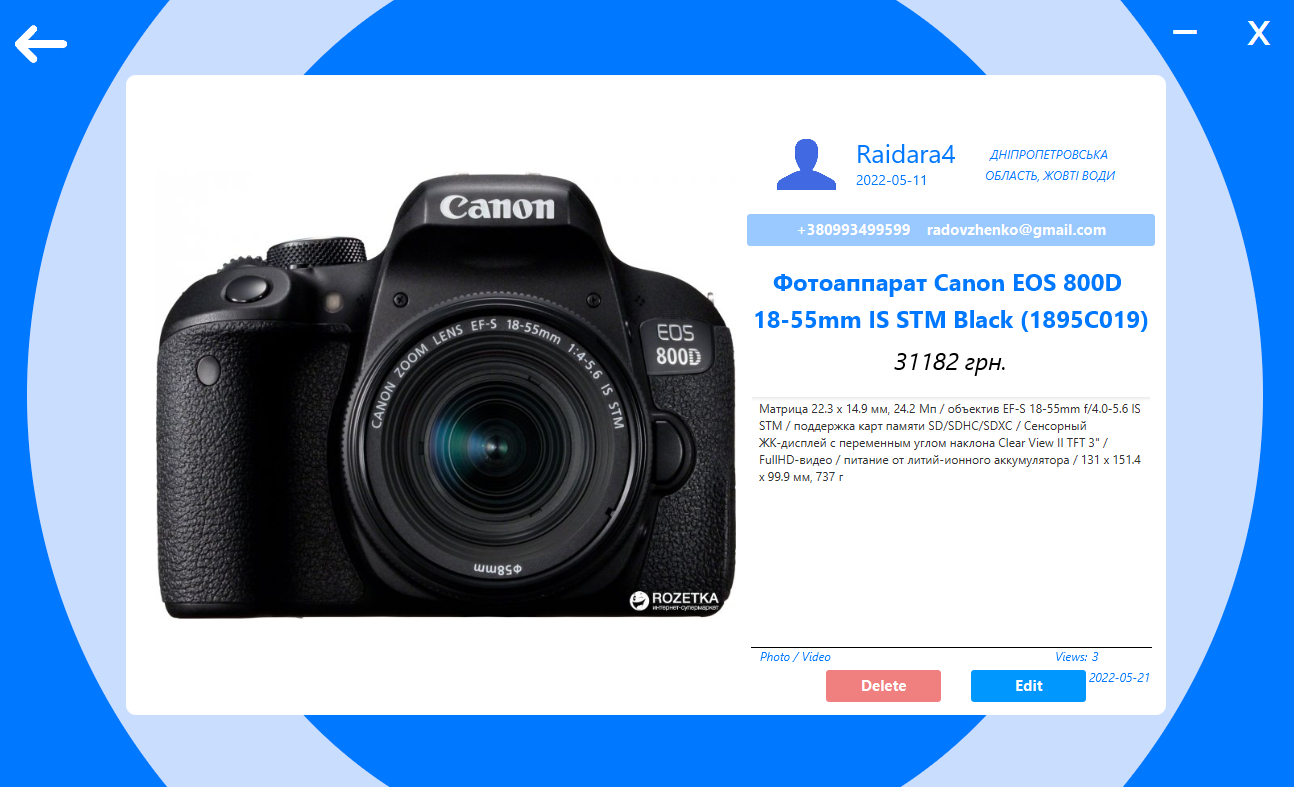
На вікні демонстрації оголошення користувач, що не є творцем оголошення може натиснути кнопку перегляду номера телефона та електронної пошти автора оголошення (рис 2.3.10). При натисненні кнопки автор оголошення дізнається про те, що було надано доступ до його даних шляхом збільшення лічильника кількості переглядів даних (рис. 2.3.7).

****

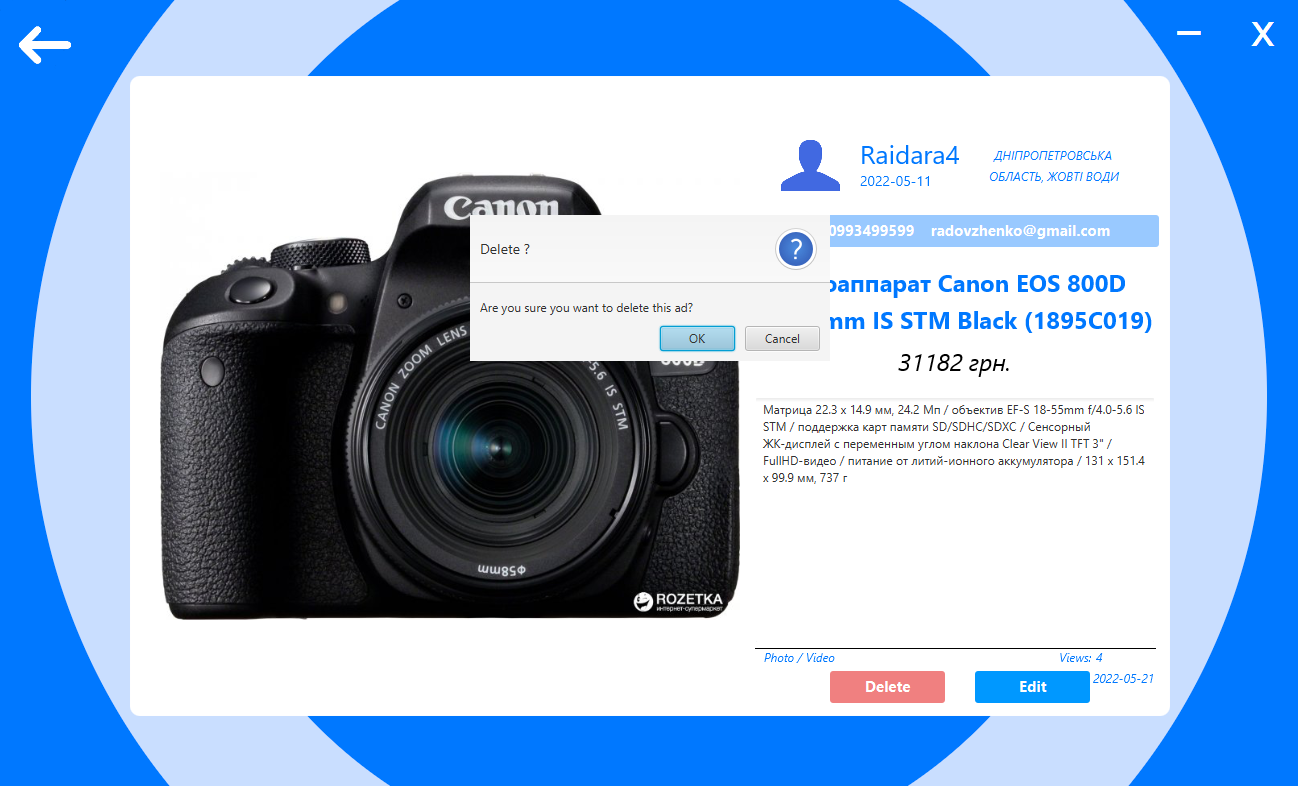
*Рис. 2.3.10. Вікно демонстрації оголошення після натиснення кнопки перегляду даних*

На вікні демонстрації оголошення користувачу, що э творцем буде доступно ще 2 додаткових кноки ( видалення оголошення, зміна оголошення) (рис 2.3.11).

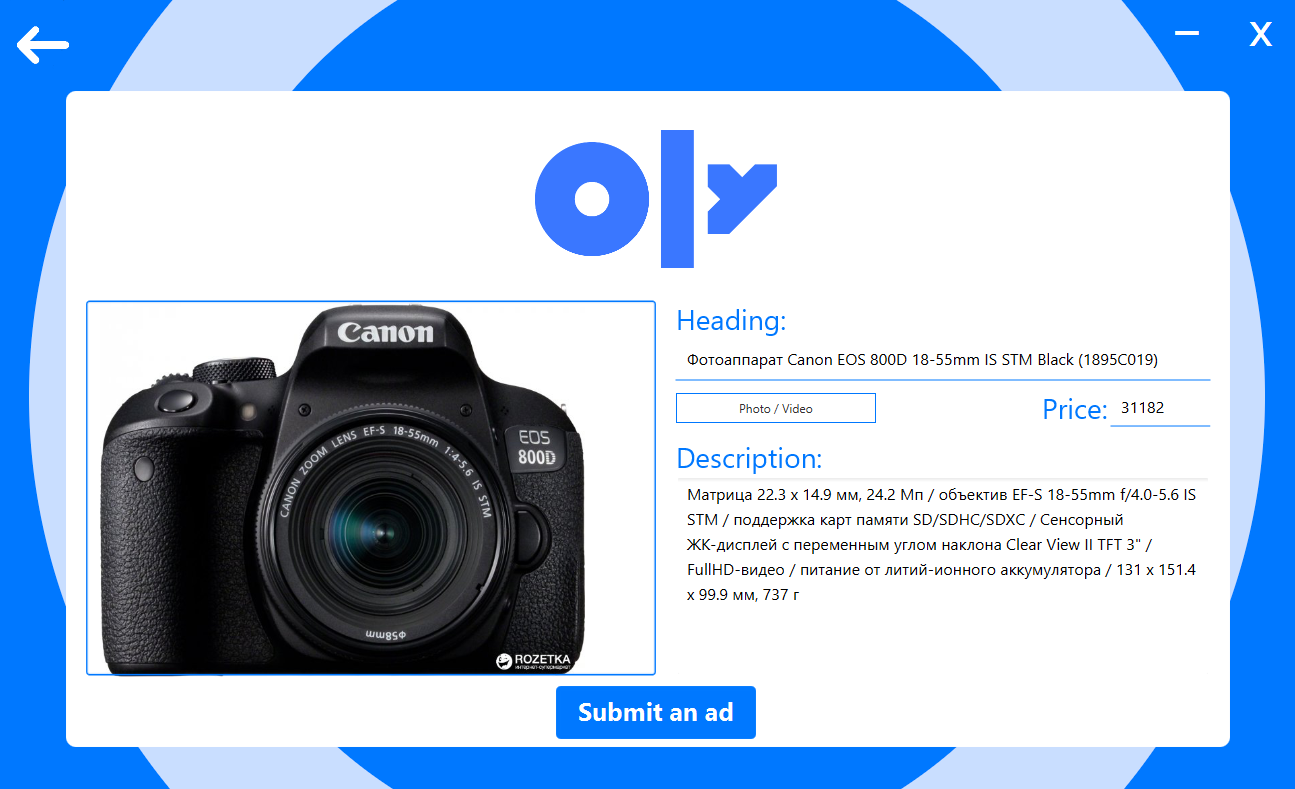
Натиснення кнопки видалення відкриває діалогове вікно підтвердження видалення (рис 2.3.12). Натискання кнопки зміни оголошення відкриває вікно редагування даних оголошення (рис 2.3.13).



*Рис. 2.3.11. Вікно демонстрації для власника оголошення*



*Рис. 2.3.12. Діалогове вікно видалення оголошення*



*Рис. 2.3.13. Вікно редагування даних*

## ВИСНОВКИ

Метою даної курсової роботи було розроблення програмного забезпечення електронної дошки оголошень з використанням шаблонів проєктування. Підставою для розроблення стало завдання на виконання курсової роботи з дисципліни «Об’єктно-орієнтоване програмування» студентами ІІ курсу кафедри ПЗКС НТУУ «КПІ ім. І.Сікорського».

Для досягнення поставленої мети у повному обсязі виконано завдання, визначені у аркуші завдання на курсову роботу; розроблено графічні матеріали; реалізовано всі вимоги до програмного продукту, програмного та апаратного забезпечення, наведені у технічному завданні; створено відповідну документацію.

Розроблене програмне забезпечення дозволяє створювати та поширювати оголошення.

Програму створено на основі використання шаблонів проєктування: зокрема до структури розробленого програмного забезпечення входить реалізація п’яти шаблонів, які належать до різних груп шаблонів проєктування (структурних, породжувальних та поведінкових).

Для розроблення програмного забезпечення використано мову програмування Java. Платформу Windows.

Перспективним напрямком подальшого дослідження даної тематики є розширення функціоналу даного додатку.

**СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ**

1. Приёмы объектно-ориентированного проєктирования. Паттерны проєктирования / Эрих Гамма, Ричард Хелм, Ральф Джонсон, Джон Влиссидес. – 1994. – 395 с.
2. Refactoring Guru [Електронний ресурс]: [Веб-сайт]. - Режим доступу:

https://refactoring.guru/uk (дата звернення 20.05.2022) - Назва з екрана.

1. CoderLessons [Електронний ресурс]: [Веб-сайт]. - Режим доступу:

https://coderlessons.com (дата звернення 10.06.2022) - Назва з екрана.

1. Реализация паттерна Observer в MVC (Model View Controller) для JavaFX [Електронний ресурс]: [Веб-сайт]. - Режим доступу:

https://maxsite.org/page/java-mvc-observer (дата звернення 12.06.2022) - Назва з екрана.