Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мережних технологій факультету інформатики

ПЛАТФОРМА ДЛЯ ПОШУКУ ТА ЗАМОВЛЕННЯ ПРОФЕСІЙНОЇ ФОТОЗЙОМКИ

Текстова частина до курсової роботи
за спеціальністю «Інженерія програмного забезпечення»

Керівник курсової роботи:
к. ф.-м. н. Гречко А. В.

Виконала студентка:
Андрусів С.І.

Київ 2020

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мережних технологій факультету інформатики

ЗАТВЕРДЖУЮ

Зав. кафедри інформатики,

к. ф.-м. н. С. С. Гороховський

_____

(підпис)

“____”_____2020 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студента Андрусів Соломії Ігорівни факультету інформатики 3 курсу

Тема: Платформа для пошуку та замовлення професійної фотозйомки

Зміст ТЧ до курсової роботи:

Індивідуальне завдання

Календарний план

Зміст

Перелік умовних позначень

Вступ

Розділ 1 Аналіз предметної області. Постановка завдання курсової роботи

Розділ 2 Теоретичні відомості

Розділ 3 Опис реалізації програмного продукта

Висновки

Перелік використаних джерел

Дата видачі “___” _____ 2020 р. Керівник _____

(підпис)

Завдання отримав _____

(підпис)

Календарний план виконання курсової роботи

Тема: Платформа для пошуку та замовлення професійної фотозйомки

Календарний план виконання роботи:

№ п/п	Назва етапу курсового проекту (роботи)	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу.	10.10.2019	
2.	Ознайомлення з існуючою інформацією по темі	05.11.2019	
3.	Ознайомлення з існуючими системами-аналогами роботи	15.11.2019	
4.	Початок створення практичної частини	15.03.2020	
5.	Початок написання теоретичної частини	30.03.2020	
6.	Подання проміжної версії практичної частини	20.04.2020	
7.	Аналіз практичної частини; її корегування	05.05.2020	
8.	Остаточне завершення написання теоретичної частини роботи та розробки практичної частини; корегування	09.05.2020-11.05.2020	
9.	Створення презентації	11.05.2020	
10.	Захист курсової роботи	20.05.2020	

Студент Андрусів С.І.

Керівник Гречко А.В.

“_____” _____

ЗМІСТ

ПЕРЕЛІК ТЕРМІНІВ ТА УМОВНИХ ПОЗНАЧЕНЬ

1. Креативна індустрія (креативні професії) - відповідно до Розпорядження номер 265 КМУ від 24 квітня 2019 «Про затвердження видів економічної діяльності, які належать до креативних індустрій»[1] характеризується такими видами економічної діяльності: візуальне мистецтво (живопис, графіка, скульптура, фотографія), аудіовізуальне мистецтво (кіно, телебачення, відео, анімація, мультиплікація), дизайн, нові медіа та інформаційно-комунікаційні технології (програмне забезпечення, відеоігри, цифрові технології в мистецтві: 3D-друк, віртуальна, доповнена, змішана реальність тощо) та інші.
2. Нікнейм – унікальне ім’я користувача, що надається при реєстрації на сайті.
3. Валідація – процес перевірки інформації на валідність, тобто чи підходить вона вказаним критеріям.
4. Шаблонізатор – програмне забезпечення, що дозволяє використовувати шаблони для генерації кінцевих HTML-сторінок.

ВСТУП

Платформа для пошуку та замовлення професійної фотозйомки PhotoLAB – веб-сервіс, створений для спрощення процесу пошуку та замовлення зйомок та збільшення можливостей фотографів бути знайденими потенційними клієнтами. На даний момент професійна фотографія в Україні – більше бізнес, аніж творчість, красиве поєднання першого та другого. Зараз попит на професійні фотозйомки зростає, тому такий сайт буде користуватись популярністю та буде унікальним в Україні. При виборі теми роботи старалась знайти щось широковживане у сучасному цифровому світі, але не складне у розумінні для широкого кола користувачів. Тому сайт, на який всі професійні фотографи зможуть "переїхати" з OLX, де зараз продають свої послуги на рівні з вживаною технікою, збільшить шанси кожного з них бути знайденим своїми потенційними клієнтами. Цільовою аудиторією програми є фотографи, що хочуть заробляти на своєму хобі чи офіційно працюють в даній сфері, а також будь-хто, хто має доступ до Інтернету, електронну поштову скриньку та бажання (чи потребу) знайти для себе ідеального «майстра» фотографії.

Мета курсової роботи – зробити свій внесок у розвиток фотографії в Україні, полегшення пошуків фотозйомки для звичайних інтернет-користувачів, та збільшення можливостей самореалізації для фотографів.

Завдання курсової роботи – створити програмний застосунок з зручним та інтуїтивно-зрозумілим інтерфейсом, де користувачі зможуть шукати та замовляти фотозйомки, а кожен фотограф, який хоче заробляти своєю камерою, - бути знайденим своїми потенційними клієнтами.

Об’єктом дослідження є сучасний стан професійної фотографії в Україні, наявність та функціонал веб-сервісів, спрямованих на розвиток комерційної галузі в цій сфері.

Предметом дослідження є способи отримання якісних послуг фотографів замовниками в Україні та можливості фотографів рекламувати свої послуги, щоб знайти клієнтів використовуючи Інтернет.

В результаті дослідження виявлено, що в Україні є дуже багато веб-сайтів фотографів чи студій, на яких можна замовити зйомку у конкретної особи. Однак, користувачам такі сайти потрібно також шукати, тому проблема з пошуком фотозйомок залишається не вирішеною. Результати дослідження показали, що веб-сервіс для багатьох фотографів, де їх рейтинг буде залежати тільки від проведених зйомок, а не від популярності в соцмережах, необхідний і буде активно використовуватись як спеціалістами так і звичайними користувачами.

Основною мовою програмування при розробці платформи став JavaScript, робота виконувалась у середовищі WebStorm від JetBrains. Серверна частина програми реалізована на Node.js. Всі дані зберігаються у реляційний базі даних MySQL, для роботи з нею було використано програму з графічним інтерфейсом MySQL Workbench. Розробка графічного інтерфейсу виконувалась з допомогою HTML-шаблонізатора Pug та фреймворку Bootstrap.

Курсова робота складається з вступу та трьох основних розділів: "Аналіз предметної області. Постановка завдання курсової роботи", "Теоретичні відомості" та "Опис реалізації програмного продукту". У висновках підбиваються основні підсумки роботи. Посилання на наукові роботи та інші джерела інформації знаходяться у списку використаних джерел. Всі матеріали, що доповнюють текст, вкладення та програмний код прикладної програми містяться у "Додатках".

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ. ПОСТАНОВКА ЗАВДАННЯ КУРСОВОЇ РОБОТИ

1.1. Аналіз сучасного стану питання та обґрунтування теми

Сучасна українська фотографія не стоїть в одному ряду з іншими видами креативної індустрії, такими як кіномистецтво, література та навіть мода. Про неї не пишуть статей у газетах та журналах, не говорять у ЗМІ. Сучасна фотографія в Україні розвивається набагато повільніше, ніж в Європі та Америці, адже основним способом пошуку клієнтів для фотографів є сарафанне радіо.

Український фотограф та засновник одного з найвідоміших українських онлайн-проектів присвячених фотографії Photography.in.UA Сергій Топольницький у одному з інтерв'ю виділив три основні галузі комерційної фотографії в Україні:[2]

- фотографія для реклами;
- фотографія для стоків;
- замовлення фотографа для подій (весілля, дні народження, приватні фотосесії тощо).

Від себе я б додала до пункту «фотографія для реклами» fashion-фотографію та зйомки для журналів, тобто це галузь про зйомки для розвитку інших креативної індустрії. Цей тип розвивається найкраще, адже реклама – дуже важливе явище у економіці, попит на таких фотографів тільки збільшується. При цьому, знімаючи для реклами, вашу творчість не потрібно рекламувати окремо. Внаслідок цього – дуже велика конкуренція в цій галузі. Фотографи для модної індустрії зазвичай мають свою тісну спільноту, в яку дуже важко попасти.

Фотографія для стоків розвивається на онлайн-ресурсах, таких-як Shutterstock чи Depositphotos, і фактично є експортом робіт українських

фотографів на світовий фото-ринок. Цей тип не використовують як основний спосіб заробітку.

Фотографів, що працюють на зйомках подій, в Україні найбільше, адже всі аматори та початківці окупували саме цю галузь. У членів українського суспільства завжди знайдуться дати, які хочеться зберегти на фото, тому на ці послуги попит також дуже великий. Звідси і проблеми. Для клієнтів – знайти майстра, що допоможе втілити в життя їхнє бачення події, а не просто витратити кошти на некрасиві кадри, зроблені на професійну техніку. Для фотографів – виділитись серед інших працюючи на виїзд, часто без фотостудії, маючи в кращому випадку веб-сайт, в гіршому – тільки профілі в соціальних мережах, особливо у містах, які не є обласними центрами. Жителі таких міст не завжди мають цілодобовий доступ до інтернету та смартфон, щоб відвідувати популярні веб-сервіси з фотографіями, але вони всі є потенційними клієнтами для таких фотографів.

Платформа для пошуку та замовлення професійної фотозйомки PhotoLAB спрямована на вирішення проблем з замовленням фотографа для певних подій чи особистих фотосесій.

1.2.Огляд існуючих аналогів розробки

Здебільшого веб-сайти присвячені фотографії – це фото-блоги, а не соціальні мережі та платформи для багатьох користувачів. Та все ж вдалось виділити кілька відомих у всьому світі програм, які можна назвати аналогами даного проекту.

Веб-сервіс Behance[3] від Adobe по функціоналу схожий на LinkedIn, але для людей креативних професій. Сервіс представлено у вигляді веб-сайту, також існують додатки для Android та IOS. Програма дає користувачу можливість переглядати вакансії роботи(див. рис. 1.1), створювати власне портфоліо та проекти(див. рис. 1.2), а також здійснювати пошук по проектах або по особах конкретної професії(див. рис. 1.3).

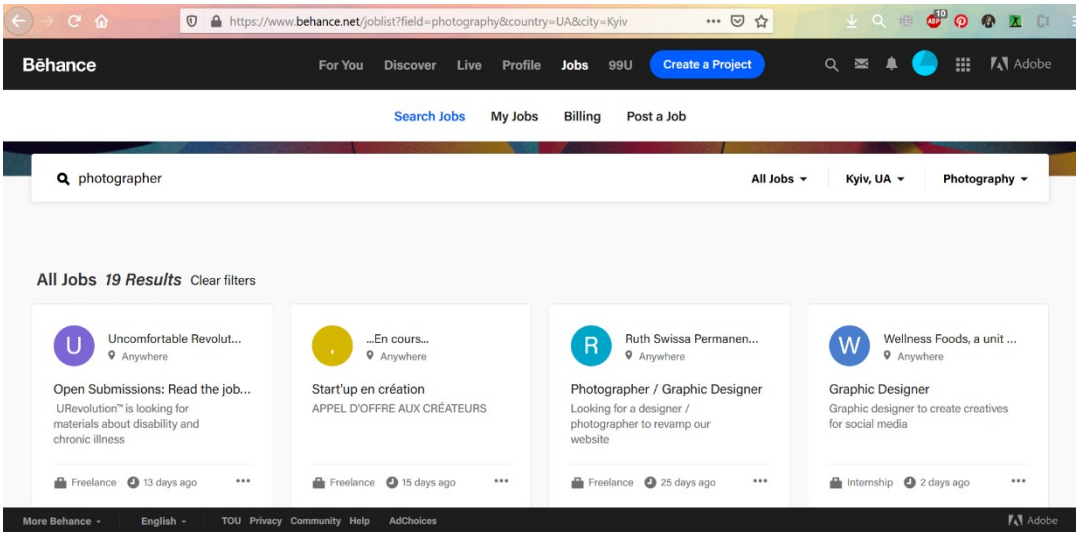


Рисунок 1.1 - Пошук вакансій на Behance

Клієнти можуть зв’язатися з особами, що пропонують послуги за допомогою чату. Для звичайних користувачів сайт дещо перевантажений інформацією, адже окрім фотографів на сайті зібрано багато інших креативних напрямків. Також, створення портфоліо доступне тільки з покупкою бізнес-плану від Adobe.

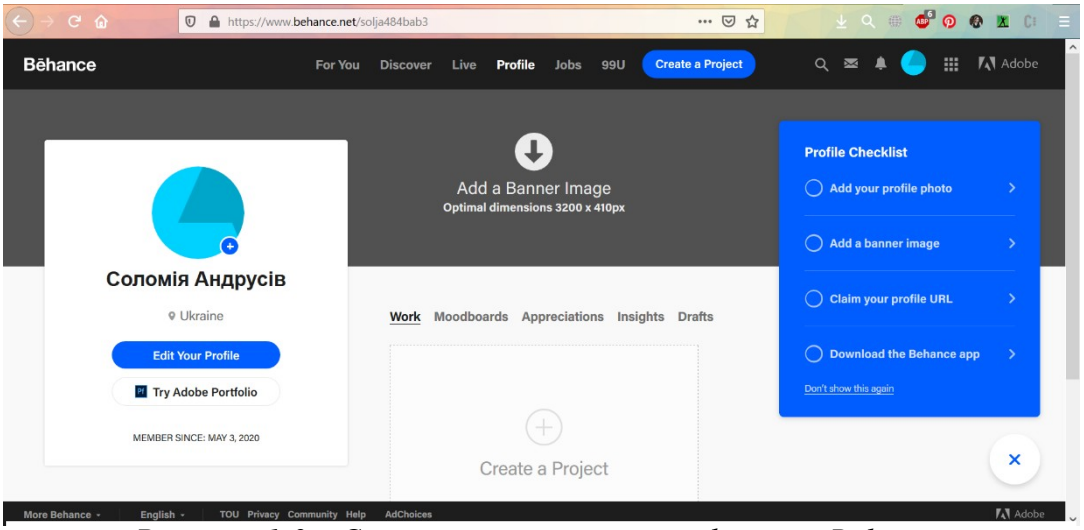
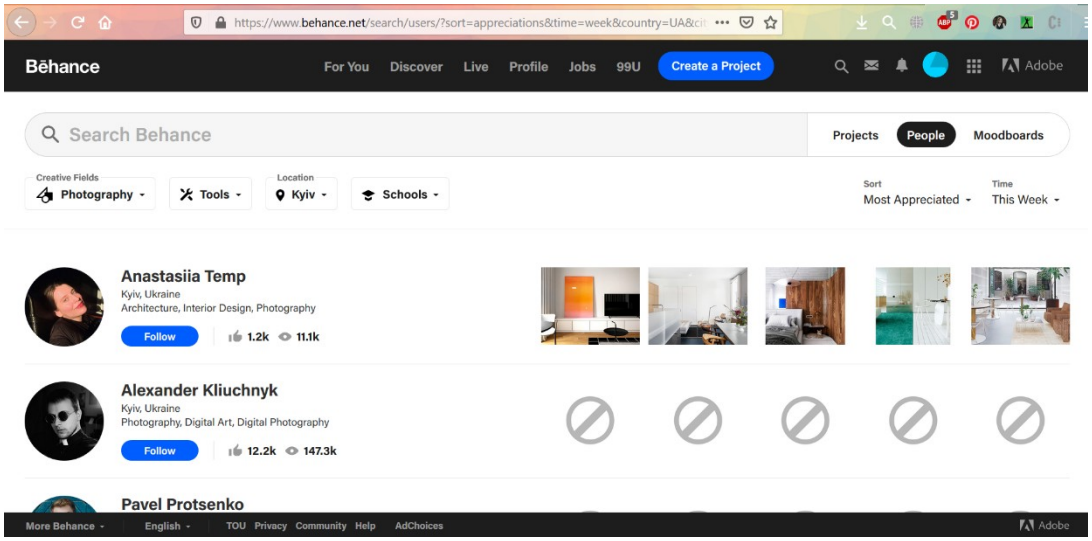


Рисунок 1.2 - Створення власного профілю на Behance

Цей сайт використовується українцями, здебільшого тими, хто уже залучений до креативної індустрії. PhotoLAB, навпаки, розробляється для користувачів, що не залучені до фото-індустрії та хочуть замовити фотозйомку в особистих цілях.



Flickr[4] – американська соціальна мережа для фотографів(див. рис. 1.4), представлена у вигляді веб-застосунку та додатків для Android та IOS. Дозволяє користувачам завантажувати фото та відео, ділитися творчістю, оцінювати роботи інших користувачів, підписуватись на них та залишати коментарі.

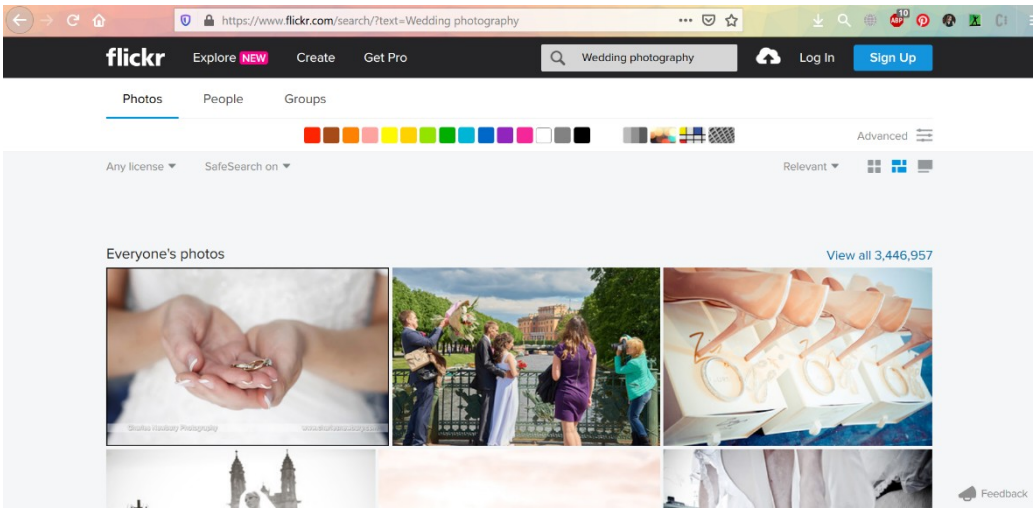


Рисунок 1.4 -Flickr для незареєстрованих користувачів

Також є широкий інструментарій для роботи з фото – після завантаження користувачі можуть переглянути всі налаштування камери, з якими зроблене фото та завантажити його в повному розмірі, якщо фотограф надав відповідний дозвіл. Flickr створений з метою поширення творчості, а не заробітку на ній, тому не містить функціоналу для замовлення фотозйомок, навіть чату.

З власного досвіду можу зазначити, що веб-сервіс зручний у використанні та зовсім непопулярний в Україні, тому він не допоможе вирішити проблему з пошуком клієнтів для фотографів. Дещо відрізняються від Flickr, але мають аналогічний функціонал, такі відомі сайти як Pinterest(див. рис. 1.5) та Unsplash(див. рис. 1.6).

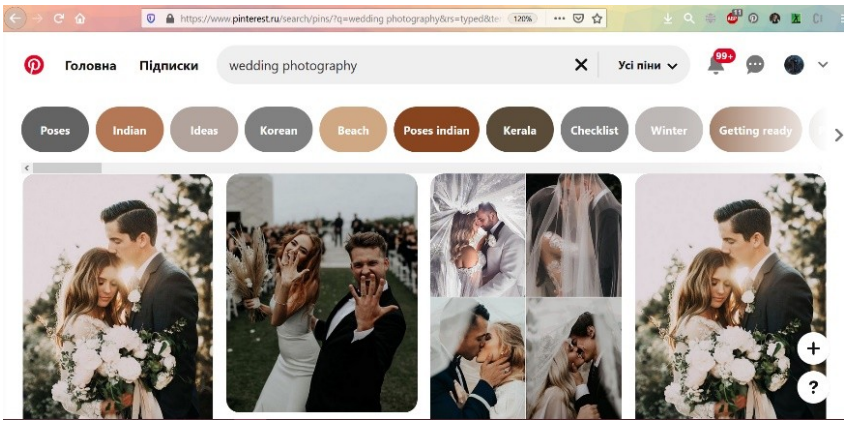


Рисунок 1.5 - Pinterest

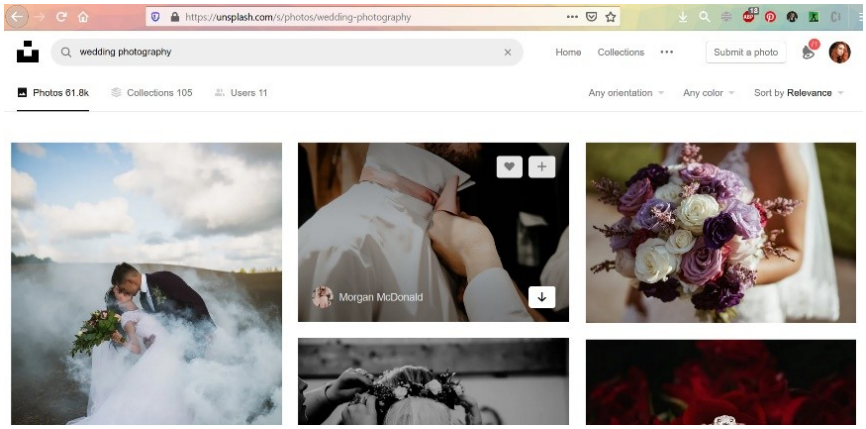


Рисунок 1.6 - Unsplash

Соціальна мережа Instagram[5] на базі Facebook - найбільш поширена платформа для обміну фото на сьогодні. Містить набір бізнес-інструментів та рекламних засобів, що дозволяють розвивати свій бізнес. На відміну від Behance, бізнес-профіль в Instagram можна активувати безплатно. Більшість українських фотографів мають профілі саме на цьому сайті(див. рис. 1.7). Зв'язок між особою, що пропонує послуги чи товари, та потенційним клієнтом здійснюється за допомогою чату. Найбільшою проблемою Instagram є його абсолютна залежність від кількості підписників і вподобань кожного допису, переглядів профілю та інших статистичних даних, а також дуже велика кількість користувачів. Щоб розвивати свій бізнес у цій соціальній мережі потрібно зібрати велику аудиторію, що насправді дуже проблематично для осіб, що не є майстрами з СММ та не готові вкладати в розвиток профілю багато часу і коштів.

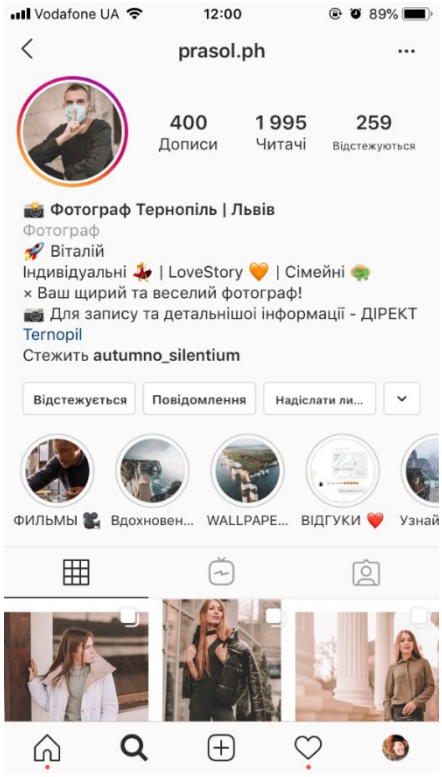


Рисунок 1.7 - Профіль фотографа в Instagram

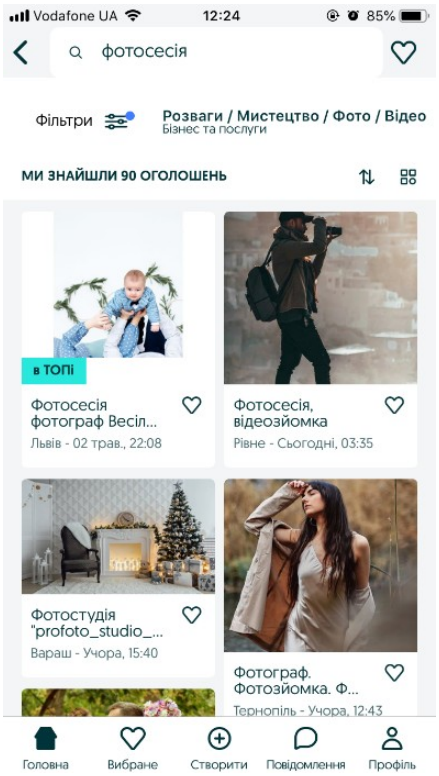


Рисунок 1.8 - Пошук фотозйомки на OLX

OLX.ua[6] – українська версія сервісу онлайн-оголошень про купівлю-продаж товарів та послуг. У програмі реалізовано такий функціонал, як пошук необхідних товарів, включно з фільтрацією за містом, ціною та відповідними рубриками. Спілкування клієнтів з продавцями відбувається за допомогою чату, також продавці обов’язково вказують номер телефону при виставленні товару на продаж. Більшість українських фотографів заробляють, знаходячи клієнтів саме на цьому сайті(див. рис. 1.8). Даний аналог по функціоналу найбільш схожий на платформу PhotoLAB.

1.3.Постановка завдання

Як було зазначено вище, фотографія в Україні розвивається дуже повільно, а веб-сервісів та соціальних мереж, на яких фотографи можуть знайти замовників, дуже мало.

Тому ідея моєї курсової роботи - створити таку платформу, яка була б популярною серед українських інтернет-користувачів, що не залучені до креативної індустрії, та полегшила б для них процес пошуку фотосесій. А для фотографів з професійною технікою, у яких хороше портфоліо та багато теоретичних знань, але немає фотостудії та популярних профілів в соцмережах, знайти потенційних клієнтів.

Основними завданнями курсової роботи є:

1. Дослідження найбільш популярних та широковживаних технологій, що використовуються при розробці веб-застосувань. Вибір тих із них, в яких можна найкращим способом реалізувати повний функціонал веб-сервісів потрібного типу, та які можуть бути використані при розробці курсового проекту.

2. Пошук всіх можливих аналогів розробки, аналіз їх функціональних можливостей, графічного інтерфейсу, типу програми та

популярності серед української аудиторії. Встановлення основних закономірностей в роботі цих програм, які в подальшому можна використати при реалізації курсового проекту.

3. Розробка онлайн-платформи для пошуку та замовлення професійної фотозйомки, в якій буде реалізовано наступні функціональні можливості:
    - реєстрація та авторизація на сайті;
    - перегляд профілів інших користувачів;
    - збереження посилань на вподобані профілі;
    - оцінка фото-профілів та формування їх рейтингу;
    - завантаження фотографій;
    - комунікація між клієнтами та фотографами з метою замовлення фотозйомки;
    - пошук профілів та окремих фотографій, фільтрація отриманих результатів.

РОЗДІЛ 2 ТЕОРЕТИЧНІ ВІДОМОСТІ

В Інтернеті існує безліч веб-сайтів, що випускаються з різною метою для різних типів користувачів. Незважаючи на те, що сфера веб-розробки постійно розвивається, усі сайти можна згрупувати за певними критеріями. В літературі можна знайти багато різних класифікацій відповідно до веб-архітектури, типу занять тощо.

Насправді у всіх веб-сайтів є одна загальна мета – вони повинні приносити дохід. Зазвичай це відбувається прямим способом – користувач здійснює покупку товару чи послуги на сайті. Також, заробіток на сайті може бути непрямим – майже всі безплатні та розважальні сайти заробляють на рекламі. Враховуючи цю інформацію, вибрано класифікацію, що ґрунтується на цілях використання сайту та на тому, яким способом вони приносить дохід одночасно[7]. Таким способом всі веб-застосування можна поділити на три категорії там сім підкатегорій:

1. Комерційні сайти (Commercial)

   Основним завданням всіх таких сайтів є отримання комерційної вигоди, яка може бути виражена як у прямому збільшенні доходів компанії, так і в зростанні інших комерційних показників. Дані сервіси отримують дохід прямим способом, вони повинні також підтримувати техніку електронних платежів за допомогою кредитних карток, PayPal або інших способів оплати.

   1) B2B(Business to Business)

      Дані веб-сайти дозволяють бізнесу здійснювати комерційні операції з іншими бізнесами, призначені для продажу продукції та послуг підприємствам, а не роздрібним споживачам. Зазвичай це робиться при пошуку матеріалів. Серед відомих веб-сайтів такого типу - Oracle, Caterpillar, Trello, Alibaba, Dropbox for Business та дуже багато інших. Бізнес як явище зараз активно

розвивається, в світі стає все більше приватних підприємців та компаній, тому більшість комерційних сайтів поступово розширюються, щоб включати цю підкатегорію.

2) B2C(Business to Consumer)

На відміну від сайтів типу бізнес-до-бізнесу, веб-застосунки цього типу відносяться до процесу роздрібного продажу товарів окремим покупцям. До таких належать сайти для покупок, транспортування, туристичних агентств тощо. Цей тип сайтів підтримують такі великі компанії як Google, Amazon, Apple, McDonalds, Starbucks та багато інших.

3) C2C(Consumer to Consumer)

Основна мета сайтів типу покупець-до-покупця – надати можливість проведення транзакцій між споживачами, тобто дають користувачам можливість здійснювати операції покупки та продажу один в одного всередині сайту. До такого типу можна віднести веб-сайти, що продають пропозиції, оголошення тощо. Прикладами застосувань, що підтримують c2c комерцію, можна вказати Amazon, Olx, eBay, Uber, Airbnb, Etsy, App Store, Google Play, Spotify тощо.

2. Сайти, що пропонують послуги (Service)

Мета веб-сайтів цього типу - представити своїм користувачам різні послуги без будь-яких витрат. Тобто вони є прикладами тієї частини сайтів, що отримує дохід непрямим способом, при цьому не проводячи грошових транзакцій з користувачами.

1) Самообслуговування(Self-service)

Основна мета веб-сайтів самообслуговування - представити клієнтам доступ до їх інформації та здійснення певних операцій. До цього типу можна віднести веб-сайти Інтернет-банкінгу, електронного уряду тощо. В Україні такими сайтами ж privat24, UKRSIB online, Oschad 27/4 тощо. Бренди використовують сайти

самообслуговування для того, щоб відобразити інформацію про товари та послуги цифровим шляхом, щоб клієнти, працівники та партнери могли легко отримувати доступ до нього. Прикладами таких сайтів є всі сайти відомих брендів одягу, такі-як Zara, Bershka, Calvin Kleins та тисячі інших.

2) Інформаційні

Основна мета таких веб-сайтів – представити користувачу інформацію, рекламу чи інші посилання на інші джерела інформації, такі як особисті веб-сайти, веб-сайти організацій / компаній, новини, блоги тощо.

3) Розважальні

Як зрозуміло з назви, дана підкатегорія сайтів не приносить користувачам комерційної вигоди, а створена для приємного проведення часу. До сайтів цього типу відносяться усі ігрові, відео-сайти, сайти-читальні тощо. Прикладами є GoodReads, Netflix, Lostfilm, Ivi.ru, YouTube. Super Mario Galaxy, StarCraft 2, Assassin's Creed, World of Warcraft тощо.

4) Комунікаційні

Комунікаційні веб-сайти створені для підтримання зв'язку між користувачами та забезпечення їх онлайн-спілкування. Сюди відносять усі соціальні мережі та месенджери – Telegram, Viber, WhatsApp, Tinder, Tumblr, Instagram, Facebook, Twitter і тд.

3. Змішані (Mixed)

Сайти, що використовуються в двох чи більше із вищевказаних категорій одночасно. До таких сайтів можна віднести веб-застосування, які пропонують користувачам покупку розваг чи інформації. До них належать усі ігрові веб-сайти, а також Netflix, YouTube та інші розважальні сайти з платною підпискою, а також величезні компанії такі як Amazon та Google, що пропонують користувачам два або більше комерційних сервісів одночасно.

Як бачимо, хоч сайти і можна поділити на певні категорії, більшість з них залишаються змішаними і використовуються у кількох цілях одночасно.

Платформа для пошуку та замовлення професійної фотозйомки створена виключно в інформативних цілях. Незважаючи на те, що на сайті відображається ціна фотосесій, застосунок не проводить грошових транзакцій. Платформа надає користувачам можливість ознайомитися з роботами фотографів та дізнатися їх контактну інформацію. Отже, її можна віднести до інформаційного типу. При цьому, сайт надає клієнтам можливість замовити зйомку, тобто один користувач може надати іншому послугу, тобто сайт надає можливість транзакцій між відвідувачами, тому його також можна віднести до типу покупець-до-покупця(C2C).

Платформа для пошуку та замовлення професійної зйомки «PhotoLAB» - в першу чергу веб-сервіс для фотографів, що дозволяє їм у зручній формі поширювати свою творчість та контакти для зв’язку з потенційними клієнтами, а користувачам – полегшувати пошук фотозйомок. В сучасному світі фотографії все більше використовуються у різних аспектах життя, тому такий сервіс точно не буде зайвим.

Особливо популярною стала фотографія в особистому вжитку: як засіб для самовираження чи збереження спогадів про важливі події, такі як весілля, дні народження, ювілеї, однією з необхідних буденних речей. Тому важливим способом полегшення пошуку буде можливість фільтрації фотографів згідно з типами зйомок, які вони проводять. Зазвичай, фотографи не працюють в одному напрямку, а в кількох суміжних, а іноді навіть кардинально різних. Проаналізувавши кілька джерел[8-10] та додавши до них те, що знаю про фотографію сама, займаючись нею уже кілька років, було складено доволі обширний список жанрів, для вибору на сайті. Деякі жанри можуть повторюватись, але так як не всі користувачі знають, наприклад, що репортажна зйомка включає в себе зйомку весілля, ювілею тощо, розділила її на зрозумілі користувачам поняття, також відкинула такі пункти як «пейзажна

зйомка», які не використовуються у комерційній фотографії. Результуючий список зйомок представлено у додатку А.

При замовленні зйомки для користувачів також важлива її ціна. Сайт розробляється для використання в Україні, тому як основну валюту для вказування ціни можна вибрати гривню. Проте, на практиці фотографи зазвичай встановлюють ціну у більш стабільній валюті – в доларах. Тому, враховуючи економічну ситуацію в Україні, було вибрано все ж другий варіант.

Проблему зв’язку між фотографом та замовником можна вирішити кількома способами. Проаналізувавши усі аналоги розробки, зроблено висновок, що основним способом зв’язку у таких веб-сервісах є чат. Також, не буде зайвим добавити посилання на інші соцмережі, телефон та електронну пошту на випадок, якщо користувачі не будуть відповідати в чаті або цей спосіб не буде для них зручним.

Платформа розробляється таким чином, щоб на рейтинг фотографа не впливали суб'єктивні оцінки користувачів, такі як лайки чи відвідування сторінки. Але для пошуку потрібен хоча б якийсь рейтинг, по якому буде видно, наскільки клієнти задоволені роботою такого фотографа.

РОЗДІЛ 3 ОПИС РЕАЛІЗАЦІЇ ПРОГРАМНОГО ПРОДУКТА

3.1. Аналіз технічного завдання

Як зрозуміло з назви, проект призначений для пошуку та замовлення фотозйомки. Перевага такої платформи у тому, що вона створена для багатьох користувачів, бажаючим замовити зйомку не потрібно годинами переглядати сайти фотографів, щоб знайти те, що їм підходить. Проект розробляється також з метою спрощення пошуку замовників для фотографів – власні веб-сайти не принесуть стільки клієнтів, як платформа, створена спеціально для тих, хто цікавиться фотографією.

Платформа в першу чергу розробляється для користувачів, не залучених до креативної індустрії. Професійна фотографія та інтернет практично не пов’язані між собою і відвідувачі не завжди мають достатньо навичок і досвіду, щоб швидко розібратись у сайті, особливо якщо це платформа для багатьох користувачів, а не лендінг чи блог. Тому інтерфейс повинен бути простим та інтуїтивно-зрозумілим, щоб користувачі могли швидко та легко орієнтуватися на сайті і здійснювати необхідні дії. Кожна сторінка сайту забезпечує виконання не більше 5ти різних функцій, а кількість спливаючих вікон та анімованих компонентів – зменшена до мінімуму, адже вони відволікають увагу користувачів.

В додаток до вищевказаного, сайт повинен надавати достатній зворотній зв’язок відвідувачам. В основному це забезпечено такими функціями:

- підказки при реєстрації – мінімальна довжина паролю, формат номеру телефона тощо;
- після перевірки форм на валідність– причину, чому введені дані не підходять: «Це поле не може бути пустим», «Неправильний формат електронної пошти» і т. д.;
- відображення спінера загрузки при пошуку та фільтрації та виведення відповідного повідомлення, якщо в результаті не знайдено жодної інформації;

- якщо при натисканні на кнопку не повинно відбуватись ніяких змін в графічному інтерфейсі (збереження інформації при редагуванні тощо), виводити повідомлення про те, що операція пройшла успішно – або повідомляти про помилку.

Платформа передбачає існування трьох типів користувачів. Усі авторизовані та не авторизовані користувачі мають можливість виконувати такі функції у програмі:

- здійснювати пошук фотографів(за іменем чи нікнеймом);
- фільтрувати результати пошуку;
- здійснювати пошук фотографій (за хештегами, що при бажанні можуть надаватись фотографами при завантаженні світлини);
- переглядати профілі фотографів та їх фотоальбоми;
- мають доступ до повної контактної інформації фотографів (номер телефону, електронна пошта, посилання на соцмережі та особистий веб-сайт);
- бачити рейтинг профілю, який переглядають;
- можливість зареєструватися та авторизуватися на сайті.

Авторизовані користувачі отримують доступ до таких функцій:

- перегляд власного профілю;
- редагування особистої інформації;
- вихід із системи (розлогінитися);
- можливість переглянути доступні для замовлення дати в календарі (дати розписані на 6 місяців вперед) та замовити зйомку у профілі фотографа;
- збереження вподобаних профілів;
- можливість виставити оцінку профілю фотографа.

Щодо останнього пункту варто зазначити, що кожен фото-профіль має свій рейтинг, який є середнім арифметичним всіх оцінок, виставлених користувачами. Користувач може оцінити профіль в межах від 1 до 10 балів.

Будь-який користувач може поставити тільки одну оцінку окремому профілю, з можливістю її видалення та встановлення нової.

Також, кожен з трьох типів користувачів має свої особливості та відповідний функціонал:

- Адміністратор

  Основні дані у платформі для пошуку і замовлення фотозйомки – особиста інформація користувачів та фотографії. Адміністратор – це стандартний тип користувача з привілейованим доступом, який може вносити зміни в дані сайту, недоступні для інших користувачів. Адміністратор сайту не може змінювати особисту інформацію користувачів. Проте, він має повний доступ до графічних матеріалів та може видаляти фотографії, що містять заборонений або недоцільний контент. До нього відносять рекламні оголошення, насильство, повне або часткове оголення, а також протизаконні, агресивні та порнографічні матеріали.

  В даному веб-сервісі існує тільки один зареєстрований адміністратор, що не має можливості додавати інших.

- Фотограф

  Цей тип користувачів має найбільш повний доступ до функціоналу на сайті. Основні функції фотографа, що не доступні для інших типів користувачів – створення папок та завантаження фотографій. Також він може виконувати операції з цими об’єктами – редагувати інформацію про світлини, додавати теги, назву і опис, перейменовувати папки та видаляти папки і фото. Також фотографи можуть додавати до збереженого та оцінювати профілі інших фотографів. На сторінці профілю фотограф може редагувати особисту інформацію, змінювати типи фотозйомок, якими займається, надати користувачам посилання на всі можливі соціальні мережі для контактного зв’язку та встановлювати новий пароль.

- Клієнт

  Основна особливість цього типу користувачів в тому, що клієнти можуть замовляти фотозйомки, але не завантажувати фото. Також, клієнтам доступні такі функції як редагування особистої інформації, збереження вподобаних профілів у закладки та їх оцінювання.

При проведенні аналізу існуючих аналогів розробки звернула увагу на такі закономірності:

- У всіх розглянутих мною в пункті 1.2 веб-застосуваннях комунікація між користувачами сайту відбувається за допомогою чату. Насправді, потенційним клієнтам зручніше зразу бачити вільні дати для замовлення, так як йдеться в основному про фотозйомки для конкретних подій (весілля, дні народження). Тому вирішено зробити інтерактивний календар із можливістю замовлення зйомки безпосередньо натискаючи на потрібну дату. Фотограф бачить нове замовлення і підтверджує його – або скасовує, вказуючи причину.
- У всіх сайтах присутній пошук світлин за вказаним запитом та акаунтів фотографів за ім’ям або нікнеймом. У більш розвинених веб-застосунках для визначення об’єктів, зображених на фото, використовується нейронна мережа. Це не є основним предметом моєї курсової роботи, тому пошук по фото буде здійснено відповідно до тегів, вказаних користувачем при завантаженні світлини. Цю систему підтримують такі відомі сайти, як Instagram, Flickr та Unsplash.

3.2. Обґрунтування алгоритму й структури програми

Проект розробляється у вигляді багатосторінкового веб-сайту з кількома типами користувачів. Отже, алгоритм має забезпечувати перехід між сторінками та авторизацію на сайті. Для відображення структури сайту та

алгоритму його роботи побудовано блок-схему, що представлена На рисунку Б.1 в додатку Б.

Об’єкти, зображені на блок-схемі – сторінки чи частини сторінок сайту, на які вдалося його логічно поділити. При відображенні більшості з них відбувається прямий чи ajax get-запит до серверу. Кожен з елементів схеми вказує, в якій частині сайту знаходиться користувач та куди він може перейти. Елементи «Клієнт», «Фотограф» та «Адмін» вказують на сторінки, доступні після авторизації під одним з типів користувачів.

Кожна сторінка сайту передбачає виконання певних функцій:

- «Головна» – основна сторінка сайту, яка складається з меню, поля для пошуку та стрічки з профілями фотографів. На ній можна здійснювати пошук (переходячи на сторінку «Пошук»), переходити на сторінки фотографів, що містяться в стрічці, зберігати їх профілі в закладки та здійснювати просту навігацію за пунктами меню: переходити на сторінки «Про нас»; «Вхід» та «Реєстрація» для неавторизованих користувачів; «Профіль» для авторизованих; «Портфоліо» якщо ввійшли як фотограф.
- «Пошук» відображає результати пошуку фото за тегами (з посиланнями на профілі авторів).
- «Про нас» - сторінка з статичною інформацією про сайт.
- «Особистий профіль» - сторінка з статичною інформацією про користувача. Містить кнопку переходу на «Налаштування» та кнопки що відображують підпункти «Контакти», «Збережені фотографи», «Замовлені зйомки» та «Календар».
- «Портфоліо» - сторінка, куди фотографи завантажують роботи. Містить повний функціонал роботи з папками та зображеннями. Неавторизовані користувачі можуть тільки переглядати існуючу інформацію в гостьовому варіанті сторінки.

- «Контакти» - частина профілю, що відображає контактну інформацію фотографа з електронною поштою, телефоном та посиланнями на соцмережі.
- «Календар» - частина профілю, що містить інформацію про вільні дати для зйомок, подану у вигляді календаря. Клієнт може замовити зйомку натиснувши на активну дату, фотограф – активувати та деактивувати дати, підтверджувати та скасовувати замовлення.
- «Збережені фотографи» - частина профілю, що містить картки з короткою інформацією про фотографів, посиланням на їх профіль та можливістю видалити їх з обраного.
- «Сторінка адміна» - сторінка з фотографіями, які адміністратор переглядає та видаляє при потребі.
- «Замовлені зйомки» - частина профілю клієнта, де міститься інформація про всі замовлення та стан їх виконання(«Скасовано», «В процесі», «Підтверджено»).
- «Вхід» та «Реєстрація» - модальне вікно входу та сторінка реєстрації відповідно.

3.3. Обґрунтування вибору засобів розробки

Дослідивши найбільш популярні та широковживані технології для веб-розробки, зупинилась на тих, що можуть найкраще реалізувати функціонал багатосторінкової платформи для кількох типів користувачів(див. рис. 3.1). Структуру проекту можна поділити на три частини: клієнтську, серверну та базу даних.

Веб-застосунок виконувався у середовищі розробки WebStorm від JetBrains. Це зручна багатофункціональна програма для веб-розробки, що має установлені плагіни для роботи з Node.js та може напряму взаємодіяти з Git. Програмний код зберігається у репозиторії системи контролю версій GitHub. Представлення сайту відбувається у браузері Mozilla Firefox.

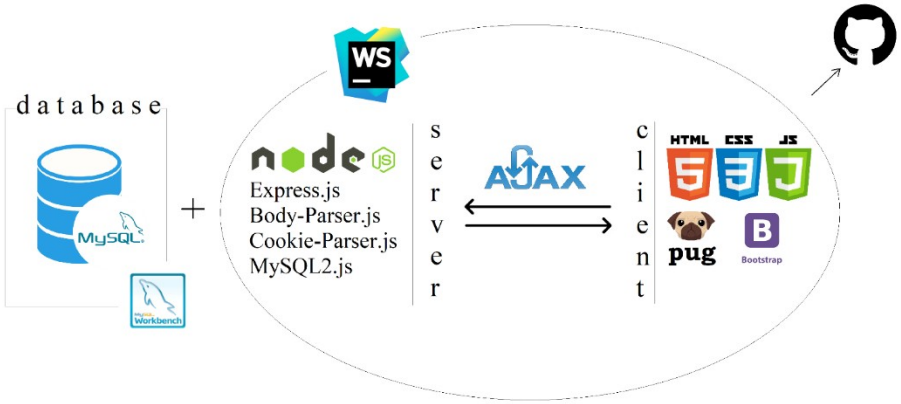


Рисунок 3.1 - Схема використаних засобів розробки

Серверна частина:

- Node.js - спрощена серверна мова програмування, що в основному використовується для розробки веб-застосувань. Має багато переваг:
  - Підтримується усіма браузерами.
  - Дозволяє легко обмінюватися даними з клієнтською частиною.
  - Підтримує безліч фреймворків, що значно спрощує написання коду.
  - Для роботи з Node.js не потрібно вивчати нову мову програмування. Коли серверна та клієнтська частини написані однією мовою та виконуються в одному середовищі розробки – це пришвидшує процес написання програми, особливо враховуючи людський фактор: програмісту не потрібно «переключатися» з однієї мови на іншу при написанні кожного запиту до сервера.
- Express.js – популярний фреймворк для роботи з Node.js, що спрощує обробку HTTP- запитів та надає багато корисних функцій, наприклад, дозволяє працювати з шаблонізаторами. Його можна легко доповнити сторонніми модулями.
- BodyParser.js – фреймворк, призначений для обробки даних, що приходять на сервер з клієнта і роботи з JSON-форматом даних.
- CookieParser.js – фреймворк, що підтримує роботу cookies.
- MySQL2.js – фреймворк, що відповідає за підключення та виконання запитів до бази даних MySQL.

Клієнтська частина:

- HTML - стандартна мову розмітки, яка використовується при розробці графічного інтерфейсу веб-застосувань.
- CSS – стандартна мова стилів для HTML-документів. Оскільки використовувала Bootstrap і власні CSS класи тільки коригували відображення елементів, у використанні LESS чи SASS не було потреби.
- JavaScript – мова програмування, що використовується у веб-розробці для динамічного відображення змін елементів HTML
- Pug – шаблонізатор для Node.js, що використовується для розробки графічного інтерфейсу замість HTML з метою спрощення процесу верстання сайту і зменшення кількості коду, конвертується в HTML при компіляції.
- Bootstrap – бібліотека стилів для JS та CSS відкритим кодом, Pug-Bootstrap – спеціально розроблена бібліотека для шаблонізатора Pug, що включає в себе міксини Bootstrap-елементів.
- Ajax – JavaScript технологія, що дозволяє асинхронно відправляти запити з клієнта на сервер, при цьому не перезавантажуючи сторінку.

База даних:

- MySQL – одна з найпопулярніших СКБД з відкритим кодом для управління реляційними базами даних, що підтримує SQL-синтаксис. Підтримується практично будь-яким програмним забезпеченням та використовує власний багатопоточний сервер для доступу до даних.
- MySQL Workbench – стандартна графічна програма для роботи з MySQL.

3.4. Опис розробки програми

Розробка веб-сайту зазвичай розділяється на такі три частини, як клієнтська, серверна та створення бази даних і робота з нею.

Серверна частина реалізована на Node.js

До серверної частини відноситься:

- файл server.js, де створюється та запускається сервер і знаходяться всі функції, написані для роботи з ним.
- папка node_modules, в якій містяться всі стандартні бібліотеки, необхідні для роботи з Node. Також тут знаходяться всі встановлені за допомогою npm фреймворки, зокрема express.js, mysql2.js, body-parser.js та cookie-parser.js.
- config.json, у якому зберігається деяка статична інформація: дані, необхідні для встановлення зв'язку з базою даних mysql(ім’я хоста, ім’я користувача, пароль та назва бази даних), ключ для хешування, заголовок html-документа та посилання на лінк для заповнення аватарів без фото.
- custom_modules.js – Java Script файл, підключений до серверної частини як додатковий модуль, містить функції, що використовуються для реалізації пошуку фото за тегами.

Клієнтську частину реалізовано за допомогою JavaScript та JQuery, для верстки сайту використано фреймворк Bootstrap та HTML препроцесор Pug. При створенні проекту Node.js WebStorm генерує стандартну ієрархію папок. Зокрема усі .js та .css файли знаходяться у директоріях public/javascripts та public/stylesheets відповідно. Файли .pug розташовані у папці views.

До клієнтської частини належать:

- JavaScript
  - requests.js – файл, що містить усі методи які виконують ajax-запити до серверу, сюди увійшли функції логіну та реєстрації, редагування додавання і видалення фото та папок, встановлення і видалення оцінки профілю, додавання та видалення з вподобаних;
  - validation.js – файл, що містить усі методи для перевірки полів форми на валідність;

- edit.js – містить усі методи, призначені для роботи з редагуванням профільної інформації всіх типів користувачів;
- search.js – файл, що містить усі функції, що використовуються при пошуку та фільтрації;
- html.js – файл, що містить функції, які генерують та вставляють в елементи html-код.

- CSS
  - general.css – містить класи, що встановлюють загальний вигляд елементів – розмір шрифту, відступи, кольори тексту тощо;
  - menuitems.css – містить класи, що описують вигляд посилань та всі класи, які змінюють стилі при наведенні курсора;
  - photos.css – містить класи, призначені для роботи з фото.
  - calendar.css – містить класи, що описують вигляд календаря для замовлення зйомок, усіх станів його комірок та карток замовлень.
- PUG-Bootstrap – підключена бібліотека з стилями Bootstrap для використання у шаблонізаторі pug.
- Pug – шаблонізатор Node.js, що використовується замість HTML для верстання сайту. Дозволяє включати один файл в інший.
  - navbar.pug – містить «шапку» HTML-сторінки та меню, що відображається для авторизованих користувачів;
  - navbar_unsign.pug – містить «шапку» HTML-сторінки та меню, що відображається для неавторизованих користувачів;
  - links.pug – містить посилання на стилі;
  - scripts.pug – містить посилання на файли .js;
  - modals.pug – у файлі зібрано усі модальні вікна, що не потребують надісланих з серверу даних;
  - home.pug – головна сторінка сайту; містить поле для пошуку та стрічку з картками фотографів і посиланнями на їх профілі;
  - profile.pug – сторінка профілю користувача;

- profileph.pug – сторінка профілю фотографа(тут же міститься календар для замовлення зйомки);
- photoalbum.pug – сторінка з папками та роботами фотографа;
- search.pug – сторінка з результатами пошуку по фото, містить поле пошуку та картки фото з посиланнями на профілі фотографів;
- admin.pug – сторінка адміністратора з фотографіями, які він може видаляти;
- edit.pug – сторінка з формою для редагування інформації;
- registration.pug – сторінка з формою для реєстрації;
- about.pug – сторінка з статичною інформацією про сайт.

Взаємодія сервера з клієнтом відбувається двома способами.

Запити до серверу методом POST здійснюються за допомогою Ajax. Сервер обробляє отримані дані та надсилає в Ajax інформацію про успішне виконання чи виникнення помилки за допомогою команди send({об’єкт з даними}).

Запити методом GET відбуваються при переході між сторінками. На сервері відправляються запити до бази даних, необхідна інформація групується та відправляється безпосередньо в файл розмітки .pug за допомогою команди render(шлях до потрібної сторінки.pug,{об’єкт з даними})

Для збереження даних про авторизацію використовуються файли cookies. В них зберігаються такі 4 параметри: унікальне ім’я користувача, унікальний ідентифікатор користувача в базі даних, роль користувача та змінна типу Boolean про те, що він авторизувався в системі.

Хешування паролів відбувається з використанням вбудованого модуля crypto.js, що підтримує обчислення хеш-сум OpenSSL, аутентифікацію з HMAC та шифрування-розшифровки. Ключ до хешування зберігається у файлі конфігурації. Дані хешуються повторно 6 разів за допомогою контрольних сум SHA-256.

3.5. Створення об'єктів і розробка головної програми

Створення об’єктів та реалізація функціональних вимог відбувається відповідно до принципів ООП, наскільки це можливо на JavaScript. Усі функції виконуються окремими методами з відповідними назвами. Частини програмного коду, що використовуються два чи більше разів також винесено в окремі методи. Розглянемо програмну реалізацію функціональних вимог, зазначених у 1.3:

- Реєстрація

  Реєстрація відбувається у два етапи. При переході на сторінку реєстрації сервер загружає з бази даних усі імені зареєстрованих користувачів та їх електронну пошту.

  Користувач заповнює форму відповідно до обраного типу користувача. В ній вказує такі дані, як нікнейм, електронну пошту та місто. Якщо користувач реєструється як фотограф, вказує також стаж роботи, середню ціну зйомки, ПІБ(по-батькові необов’язково) та місце роботи (якщо таке є). Також він може зразу вибрати усі типи зйомок, якими займається. Останніми є поля для введення та повторення введеного паролю.

  Введення всіх даних відбувається всередині форми в поля input вілповідного типу(number для ціни, email для пошти, password для пароля тощо). Стаж роботи вибирається з одної із запропонованих опцій елементу select.

  При натисканні кнопки «Зареєструватись» викликається метод regClient() та regPh() відповідно до вказаного типу користувача. Так як при реєстрації фотографа реалізується більший функціонал, надалі буде розбиратись метод regPh(). Уся введена користувачами інформація проходить перевірку на валідність. Графічно валідація відбувається за допомогою Bootstrap-класів is-valid та is-invalid. Коли полю встановлюється клас is-invalid стає видимим повідомлення про причину невалідності з класом invalid-feedback. Другий етап валідації – перевірка введеного нікнейму та електронної пошти з тими, які прийшли з серверу при загрузці сторінки валідації. Ці поля повинні бути унікальними. Якщо

такий нікнейм чи пошта вже існує, користувач побачить відповідне повідомлення. Коли всі дані введено у правильному форматі і введений два рази пароль збігається, формується об’єкт data з відповідними полями та переводить його у JSON-формат.

Виконується ajax-запит методом POST до серверу на url= '/registerph'. На сервері пароль хешується і виконується вставка у таблиці users та photographers(для реєстрації клієнта url='/register' і вставка відбувається тільки в users).

Якщо реєстрація пройшла успішно, дані входу користувача записуються в cookies і сервер повертає в ajax об’єкт {'success': 'yes', 'href':адреса переходу}. В клієнтській частині відбувається перехід за цією адресою використовуючи метод window.location.replace.

Реєстрація завершена і користувач перейшов на сторінку щойно створеного особистого профілю.

- Авторизація на сайті

  При натисканні на кнопку «Вхід» відкривається модальне вікно. Користувач вводить логін і пароль та натискає на кнопку підтвердження. Викликається метод login(), а якому виконується ajax-запит методом POST на '/login'. На сервері виконується запит до бази даних, який шукає користувача з вказаною електронною адресою. Якщо такий користувач не знайдений, в ajax повертається success:no і користувачу виводить відповідне повідомлення. Якщо користувача з таким логіном знайдено, введений в форму входу пароль хешується і звіряється з тим, що зберігається в базі даних. Якщо паролі збігаються, в cookies записуються дані входу – нікнейм, роль користувача, його id і те, що він успішно авторизувався. Після цього на сторону клієнта відправляються дані про успішну авторизацію і на клієнті за допомогою window.location.replace користувача перенаправляє на головну сторінку, але уже з іншим url – '/home/:username'. Користувач успішно авторизувався на сайті!

- Перегляд профілів інших користувачів

Користувач може перейти на профіль фотографа натиснувши на його нікнейм. Посилання відправляє користувача на '/guest/:username'. На сервері за нікнеймом фотографа з бази даних збирається вся інформація для відображення на профілі. Якщо все пройшло успішно, виконується res.render сторінки profileph.pug, на якій відображається вся необхідна інформація.

- Збереження посилань на вподобані профілі

  Користувач може зберегти посилання на профіль фотографа натиснувши на символ закладки. Після натискання на закладку викликається функція addToFavorites() з двома параметрами – id вподобаного фотографа та id користувача який його зберігає. У функції виконується ajax-запит методом POST до серверу на url= '/addfavorite'. На сервері відбувається вставка нового рядку в таблицю favorites. Якщо все пройшло успішно, і назад в ajax повернувся об’єкт з параметром success:yes, то значок закладки з контуру змінюється на повністю зафарбований, з onclick-функцією deleteFromFavorites(…) з такими ж параметрами.

- Оцінка фото-профілів та формування їх рейтингу

  Клієнт може поставити оцінку фотографу при відвідуванню його профілю. У верхньому правому кутку профілю знаходяться 10 незаповнених «зірок». Так як оцінки, виставлені користувачем, завжди в межах від 1 до 10, то середній рейтинг варіюється від 10 до 0 (якщо профілю не поставлено жодної оцінки). При натисканні на іконку зірки викликається функція vote() з трьома параметрами – id користувача який ставить оцінку, id фотографа якого оцінюють і власне оцінка, що відповідає номеру зірки. У методі відбувається ajax запит методом POST на '/vote'. На сервері дані записуються у таблицю ratings, потім виконується ще один запит, в якому за допомогою агрегатних функцій Round та Avg обраховується новий рейтинг. Назад в ajax приходить підтвердження успішної операції та інформація про оновлений рейтинг. Після цього викликається метод setRatings(), що змінює зовнішній вид іконок на заповнені рівно на стільки балів, скільки став новий рейтинг.

Під іконками зірок з’являється надпис, у скільки балів користувач оцінив профіль та кнопочка «Видалити голос» з onclick-функцією unvote().

- Завантаження фотографій

  Завантажувати фотографії можна авторизованим фотографам. На сторінці фото-профілю користувач натискає на кнопку «Додати фото». Фото додається в активну папку. Ідентифікатор активної папки зберігається в localStorage, і змінюється при натисканні на назву папки. Отже, при натисканні на кнопку відкривається модальне вікно з полями для введення посилання на фото, назви, опису та тегів. Крім посилання усі елементи є необов’язковими. При натисканні кнопки підтвердження викликається метод addPhotoLink(). У ньому виконується валідація тегів та перевірка чи не пусте поле з посиланням. Якщо дані коректні, вони відправляються на сервер за допомогою ajax-запиту методом пост на url='/addphoto'. На сервері відбувається вставка нового рядка в таблицю photos. Після успішного виконання операції з базою даних, в ajax відправляється підтвердження. Форма очищується від введених значень, у папку додається нове фото, модальне вікно закривається. Фото додано!

- Замовлення фотозйомки

  Клієнт натискає на активну дату на календарі і відкривається діалогове вікно замовлення зйомки. В ньому користувач повинен ввести тему зйомки і надати контакт для зв’язку. Ці поля не можуть бути порожніми. Також можна ввести додаткове повідомлення за бажанням. При натисканні «Підтвердити замовлення» викликається функція makeOrder(). Функція валідує поля вводу і відправляє ajax-запит POST на url='/order'. На сервері дані про нове замовлення записуються в базу даних і йому надається статус «в процесі». Якщо при цьому не виникло помилок, користувач бачить повідомлення про те, що замовлення пройшло успішно і через 15 секунд модальне вікно закривається. Нове замовлення з’являється у профілі користувача в розділі «Фотозйомки». На цій сторінці містяться також дані про всі скасовані та прийняті замовлення.

Фотограф заходить на профіль і бачить картку з новим замовленням під календарем у колонці «Нові». Він ознайомлюється з інформацією про замовлення і може підтвердити або скасувати його за допомогою відповідних кнопок. Внаслідок натискання одної з них відкривається модальне вікно. Якщо це кнопка скасування замовлення, то фотографу надається можливість вказати користувачу причину відмови.

Якщо це підтвердження то фотограф вказує контактну інформацію для подальшого зв’язку і може ввести додаткове повідомлення, наприклад обумовити час зустрічі. Після натискання кнопки підтвердження викликається функція approve(). В ній дані відправляються на '/approve' за допомогою ajax методом POST. На сервері відбувається редагування замовлення в таблиці orders, додається нова інформація і статус змінюється із «у процесі» на «підтверджено». Після цього виконується ще одна операція з базою даних, яка видаляє рядок з датою замовлення з таблиці free dates. Тепер дата забронbована. Сервер повертає дані про успішне виконання в ajax і картка замовлення переміщується у колонку підтверджених. Проте його все ще можна скасувати.

При скасуванні викликається метод cancel(), запит POST на '/cancel' і статус замовлення в таблиці orders змінюється на «Скасовано». Після цього дані про замовлення переміщуються в колонку «Скасовані». Фотограф може відновити скасовану заявку та скасувати підтверджену. Скасоване замовлення видаляється з бази даних після того, як клієнт ознайомиться з інформацією про відмову та видалить замовлення зі свого профілю. Після цього воно зникає з профілю фотографа і дата замовлення стає знову вільною.

- Фільтрація

  Фільтрація фотографів відбувається за 4ма ознаками: тип зйомки, місто, стаж та ціна зйомки. Для ціни зйомки можна задати необхідні межі(від 1 до 5000$). Інші критерії вибираються з елементів select.

При натисканні на кнопку «Фільтрувати» викликається функція filter(). Користувачу відображається спінер загрузки, а в методі виконується ajax-запит POST на url='filter', формується запит до бази даних на основі вказаних параметрів. Якщо запит успішний і знайдено хоча б одного фотографа, ця інформація відправляється назад в ajax разом з параметром success : yes. Тоді на фронтенді формуються нові картки фотографів, старі зникають і зникає спінер загрузки. Якщо в ajax повертається success : no - виводиться повідомлення про те, що нічого не знайдено.

Кнопка «Відмінити» на панелі фільтрації викликає метод nofilter(…), що повертає стрічку до попереднього вигляду.

- Пошук профілів та окремих фотографій

  При натисканні на кнопку пошуку викликається метод search(), з localStorage дістаються дані про те, яким способом відбувається пошук - за особами чи за фото - та поле пошуку перевіряється на наявність введеного запиту.

  При пошуку за особами викликається функція searchPerson() з параметром пошуку який містить всю введену в поле інформацію до першого пробілу. Виконується запит ajax на '/searchph'. На сервері з бази даних вибираються усі фотографи з іменем, прізвищем чи нікнеймом, що відповідає введеному запиту. Дані відправляються на клієнтську частину і відображаються на головній сторінці.

  При пошуку за фото форма відправляє запит get на /search. При запуску сервера з усіх фото, що містяться в базі даних, було сформовано інвертований список (Map), у якому key=тег та value=ідентифікатори фото, що містять цей тег. При натисканні на кнопку пошуку на сервері запит ділиться на окремі слова, в інвертованому списку шукаємо номери фото, що містять будь-яке з цих слів. Потім масив цих ідентифікаторів сортується за частотою, тобто фото, що містять найбільше тегів з запиту, будуть показуватися користувачу першими. Після цього з бази даних дістається інформація про фото з вказаними ідентифікаторами та

користувача перенаправляє на сторінку search.pug, де власне відображаються результати пошуку.

3.6. Опис файлів даних та інтерфейсу програми

Платформа для пошуку та замовлення професійної фотозйомки – веб-застосунок, що працює з базою даних mySQL. Схема таблиць та зв’язків між ними представлена на рисунку В.1 у додатку В.

Опис таблиць та їх компонентів:

1. «Users» - таблиця містить загальні дані про усіх зареєстрованих користувачів веб-сайту. Первинний ключ – user_id. Альтернативні ключі – email та username.

   Атрибути:

   - user_id – унікальний номер користувача в базі даних;
   - email – електронна пошта;
   - username – унікальне ім’я користувача;
   - user_pass – пароль(видозмінений за допомогою хеш-функції);
   - avatar_link – посилання на зображення аватару профіля;
   - city – назва міста;
   - about – додаткова інформація про себе;
   - phone – номер телефону;
   - role_id – зовнішній ключ, що вказує на тип зареєстрованого користувача.

2. «Roles» - таблиця, що містить всі типи користувачів сайту. Потенційний ключ – role_id. Атрибут role є альтернативним ключем, адже назва кожного типу є унікальною.

   Атрибути:

   - role_id – унікальний номер ролі користувача в базі даних;
   - name – назва типу користувача («Фотограф», «Клієнт» та «Адмін»).

3. «Photographers» - таблиця містить дані про користувачів, що зареєструвалися як фотографи. Первинний ключ – ph_id. Альтернативний ключ – user_id.

   Атрибути:

   - ph_id – унікальний номер фотографа в базі даних;
   - lastname – прізвище;
   - firstname – ім’я;
   - fathername – по-батькові;
   - price – середня ціна зйомки;
   - exp – стаж роботи фотографом;
   - organization – якщо користувач працевлаштований як фотограф, може вказати тут назву фотостудії чи організації в якій працює.
   - user_id – зовнішній ключ, що вказує на загальну інформацію користувача.

4. «Favorites» - таблиця, призначена для збереження вподобаних користувачами фото-профілів. Первинний ключ – id. Альтернативний ключ – user_id(PPK) + ph_id(PPK).

   Атрибути:

   - id – унікальний номер вподобання в базі даних;
   - user_id – зовнішній ключ, посилання на профіль користувача, якому належать збережене;
   - ph_id – зовнішній ключ, посилання на профіль фотографа, якого вподобав користувач.

5. «Ratings» - таблиця містить оцінку, поставлену користувачем фото-профілю. Первинний ключ – id. Альтернативний ключ – user_id (PPK) + ph_id (PPK).

   Атрибути:

   - id – унікальний номер оцінки в базі даних;
   - mark – оцінка профілю(може бути від 1 до 10);

- user_id – зовнішній ключ, посилання на профіль, що поставив оцінку;
- ph_id – зовнішній ключ, посилання на профіль фотографа, якому поставили оцінку.

6. «Orders» - таблиця містить дані про замовлену фотозйомку. Первинний ключ – order_id. Альтернативний ключ – user_id (PPK) + ph_id (PPK) + date(PPK).

   Атрибути:

   - order_id – унікальний номер замовлення в базі даних;
   - date – дата замовленої фотозйомки;
   - status – зовнішній ключ, посилання на статус опрацювання замовлення;
   - topic – коротко сформульована тема замовлення;
   - contact_ph – контактна інформація фотографа для подальшого зв’язку;
   - contact_cl – контактна інформація клієнта для подальшого зв’язку;
   - message_cl – повідомлення, яке клієнт може додати до замовлення;
   - message_ph – повідомлення, яке додає фотограф після опрацювання замовлення(«Замовлення скасовано» у випадку, якщо фотограф відхилив замовлення додається за замовчуванням);
   - ph_id – зовнішній ключ, посилання на профіль фотографа у якого замовили зйомку;
   - cl_id – зовнішній ключ, посилання на профіль користувача типу клієнт, що здійснив замовлення.

7. «Order status» - таблиця містить усі можливі статуси опрацювання замовлення(всього їх три – підтверджене, скасоване та в процесі

обробки). Первинний ключ – status_id. Альтернативний ключ – name.

Атрибути:

- status_id – унікальний номер статусу в базі даних;
- name – зміст статусу.

8. «Free dates» - таблиця містить усі дати, відкриті користувачем для замовлення. Первинний ключ – date_id. Альтернативний ключ – ph_id(PPK) + date(PPK).

   Атрибути:

   - date_id – унікальний номер екземпляру сутності в базі даних;
   - date – власне вільна дата;
   - ph_id – зовнішній ключ, посилання на фотографа.

9. «Types» - таблиця містить усі типи зйомок, перераховані в додатку А. Первинний ключ – type_id. Альтернативний ключ – name.

   Атрибути:

   - type_id – унікальний номер типу в базі даних;
   - name – назва типу.

10. «Shoots» - таблиця, утворена внаслідок перетворення в реляційну модель таблиць «Types» та «Photographers» ER-моделі зі зв’язком багато-до-багатьох. Первинний ключ – ph_id(PPK) + type_id(PPK).

    Атрибути:

    - ph_id – зовнішній ключ, посилання на фотографа;
    - type_id – зовнішній ключ, посилання на тип зйомки.

11. «Social networks» - таблиця містить список соціальних мереж, посилання на які може надати фотограф як свою контактну інформацію для клієнтів. Первинний ключ – social_id. Альтернативний ключ – site_name.

    Атрибути:

    - social_id – електронна пошта;
    - site_name – назва соціальної мережі чи веб-сервісу.

- icon – назва css-класу, яку потрібно надати елементу, щоб відобразилась іконка з логотипом конкретного веб-сайту.

12. «Accounts» - таблиця зв’язку багато-до-багатьох між «Photographers» та «Social networks». Первинний ключ – ph_id(PPK) + social_id(PPK).

    Атрибути:

    - link – посилання на акаунт фотографа в даній соцмережі;
    - ph_id – зовнішній ключ, посилання на фотографа;
    - social_id – зовнішній ключ, посилання на сайт.

13. «Folders» - таблиця містить усі папки фотографа. Первинний ключ – folder_id.

    Атрибути:

    - folder_id – унікальний номер папки в базі даних;
    - name – назва папки;
    - ph_id – зовнішній ключ, посилання на фотографа, якому належить папка.

14. «Photos» - таблиця містить дані про окреме фото. Первинний ключ – photo_id.

    Атрибути:

    - photo_id – електронна пошта;
    - link – посилання на фото;
    - title – назва фото;
    - descr – опис фото;
    - tags – теги фото, по яких потім здійснюється пошук;
    - folder_id – зовнішній ключ, посилання на папку, в якій міститься фото.

Взаємодія з базою даних відбувається на сервері за допомогою фреймворку mysql2.js, що дозволяє здійснювати асинхронні запити до бази даних. При доступі до бази даних використовується багато параметричних запитів. Виконання таких запитів супроводжується безпосереднім вводом

користувачами конкретних значень параметрів запиту в поля форми. Тому перед записуванням цих даних до бази обов’язково проводиться перевірка на валідність.

Валідація полів здійснюється на клієнтській частині за допомогою регулярних виразів. Загалом у програмі присутні такі функції для перевірки введеної інформації:

1. validEmail() – перевіряє введену електронну пошту. У введеному тексті обов’язково мусять бути присутні @ та . у такому ж порядку і хоча б один символ перед ними та після них.
   /^(([^<>()[\]\\.,;:\s@\"]+(\.[^<>()[\]\\.,;:\s@\"]+)*)|(\".+\"))@(([[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$/
2. validNames() – перевіряє введений ПІБ користувача та назву міст. Допускаються усі літери латиниці та кирилиці а також апостроф і дефіс. Ім’я повинно бути не менше двох та не більше 45 символів.
   /^[a-zA-Za-яA-ЯiIїЇыЫэЭёЁъЪ\-']+$/
3. validPassword() – функція, що перевіряє введений пароль. Пароль мусить бути не коротшим за 6 символів, може містити букви, цифри, тире та нижнє підкреслення.
   /^[a-zA-Z0-9a-яA-ЯiIїЇ\-_]+$/
4. validPhone() – функція, що перевіряє валідність введеного телефону. Може бути рядок з 10 цифр, також допускаються формати (123) 456-7890 та 123-456-7890.
   /^(\()?\d{3}(\))?(-|\s)?\d{3}(-|\s)\d{4}$/
5. validEmpty() – функція призначена для перевірки полів, які не можуть бути порожніми, але не мають наперед відомого вигляду введних даних.
   /^\s+$/
6. validTags() – функція призначена для перевірки введених тегів до фото. На випадок, якщо користувач вирішить ставити # чи інші

символи. Допускаються тільки букви та цифри і пробіли між тегами.

/^[a-zA-Z0-9а-яА-ЯіІїЇ\s]+$/;

Інтерфейс веб-сайту реалізовано графічно. Він інтерактивний та динамічно змінюється, містить розвинуту систему меню та діалогових вікон.

Меню дещо відрізняється для авторизованих користувачів та для простих відвідувачів сайту(див. рис. 3.2). Усі елементи меню (крім кнопок «Вхід» та «Вихід») призначені для переходу між сторінками:

- «головна»('/' та '/home');
- «профіль»('/profile');
- «фотоальбом»('/photoalbum');
- «про нас»('/about');
- «реєстрація»('/registry').

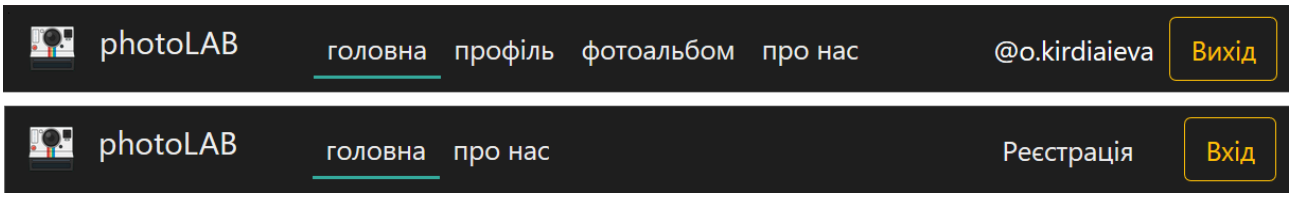


Рисунок 3.2 - Меню сайту

При натисканні на кнопку вхід відкривається діалогове вікно з полями для введення логіну та паролю. Кнопка вихід відобрежає діалогове вікно, яке просить підтвердити, що користувач точно хоче вийти з системи.(див. рис. 3.3).

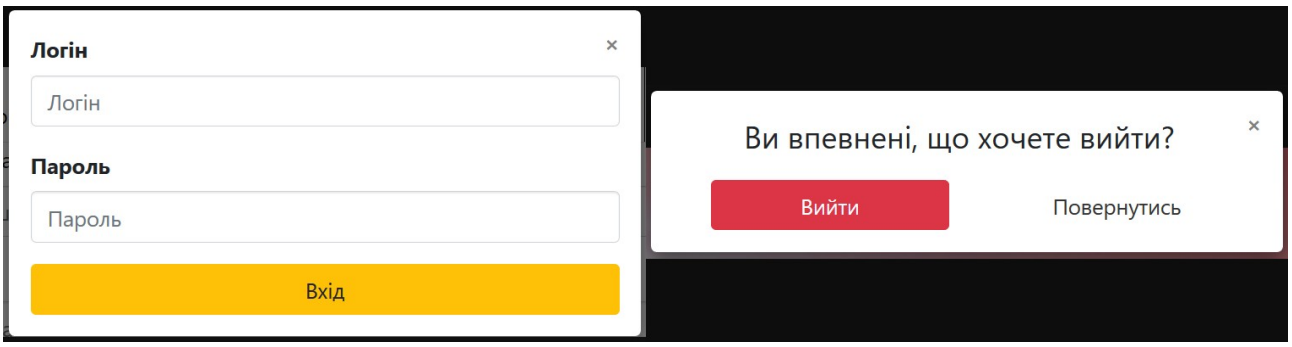


Рисунок 3.3 - Діалогові вікна для входу та виходу

Також система передбачає виведення діалогових вікон при створенні та редагуванні папки та створенні, редагуванні і видаленні світлини(див. рис. Е.5 в

додатку Е). Ці діалогові вікна містять форми з полями для заповнення інформації, що валідуються та відправляють ajax-запит на сервер. Якщо він успішний, вікно закривається і на сторінці з’являється щойно створена папка чи фотокартка.

3.7. Тестування програми і результати її виконання

Тестування веб-сайту проводилося у браузері Mozilla Firefox

Незареєстрованим користувачам доступні такі можливості системи:

- переглядати профілі фотографів та бачити їх рейтинг та всю контактну інформацію (див. рис. 3.4);

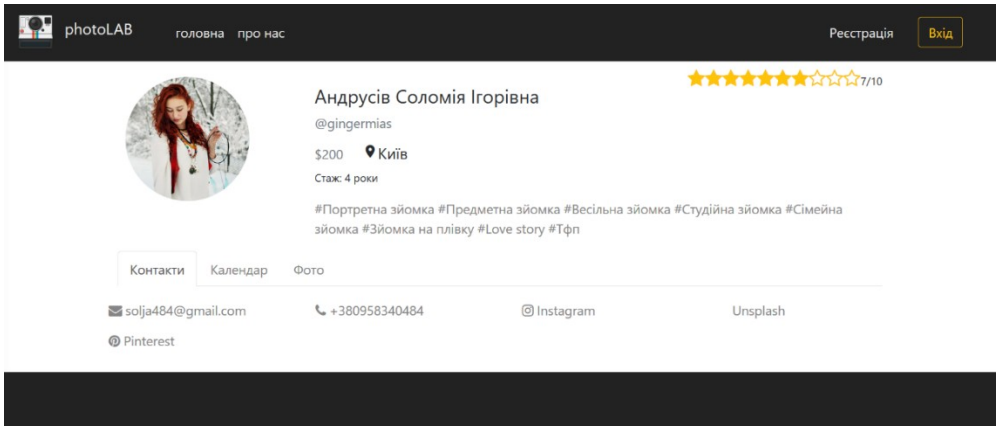


Рисунок 3.4 - Перегляд профілю фотографа

- здійснювати пошук фотографа за іменем чи нікнеймом(див. рис. 3.5);

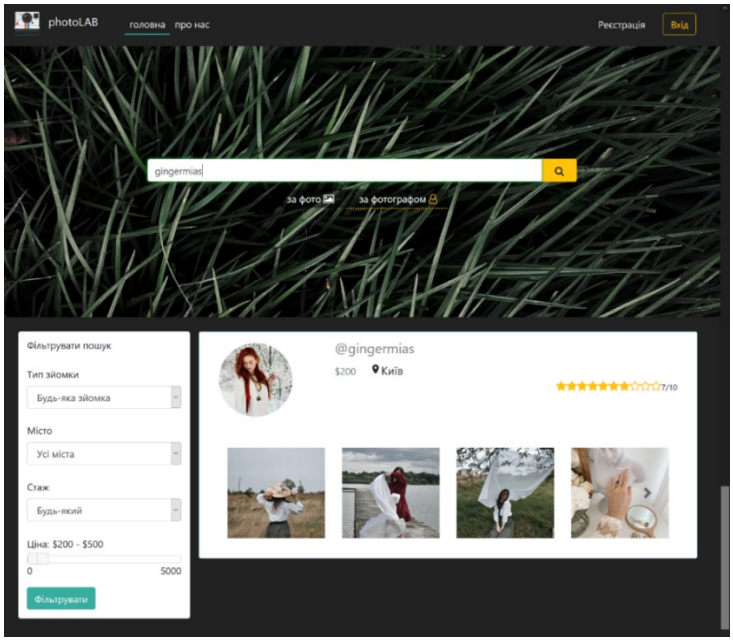


Рисунок 3.5 - Пошук за фотографом

- фільтрувати стрічку(див. рис. 3.6)

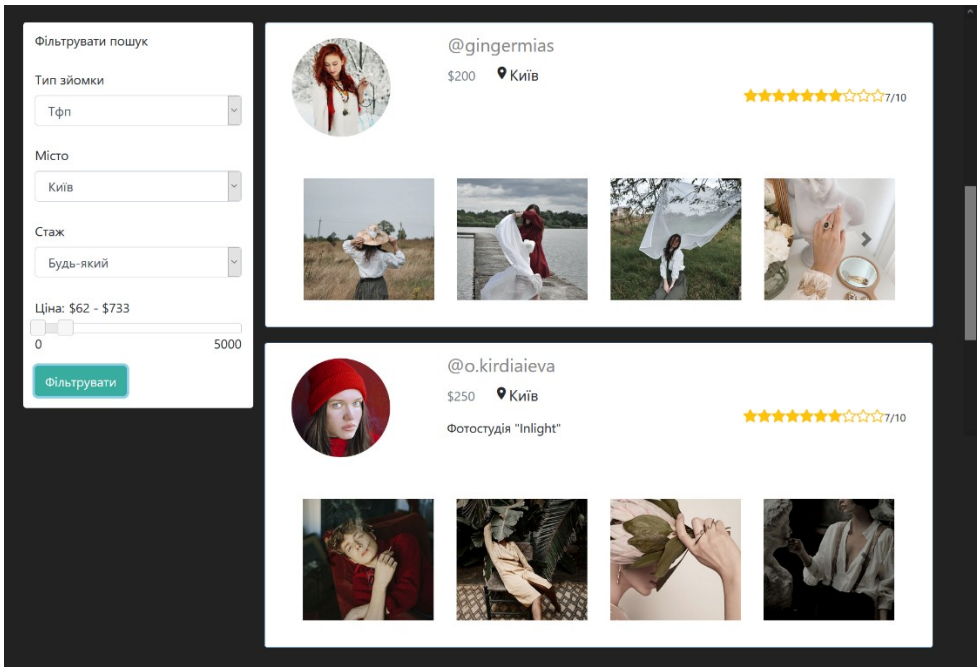


Рисунок 3.6 - Фільтрація фотографів

- здійснювати пошук фотографій(див. рис. 3.7);

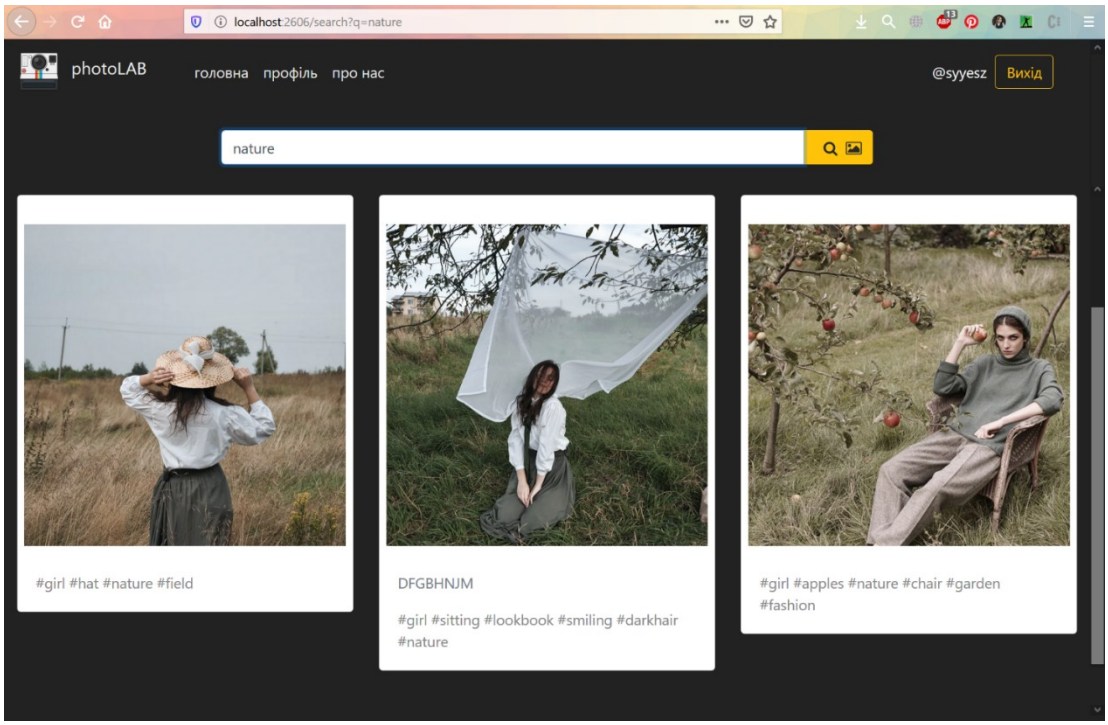


Рисунок 3.7 - Пошук за фото

- переглядати фотоальбоми (див. рис. 3.8);

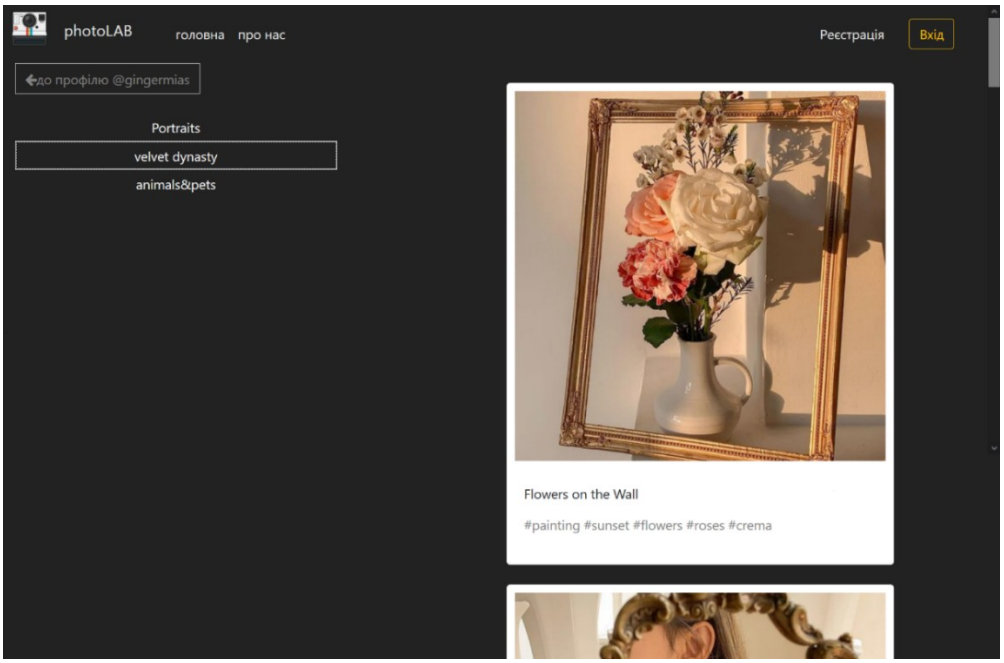


Рисунок 3.8 - Перегляд фотографій

- Переглядати календар з вільними датами(див. рис. 3.9);

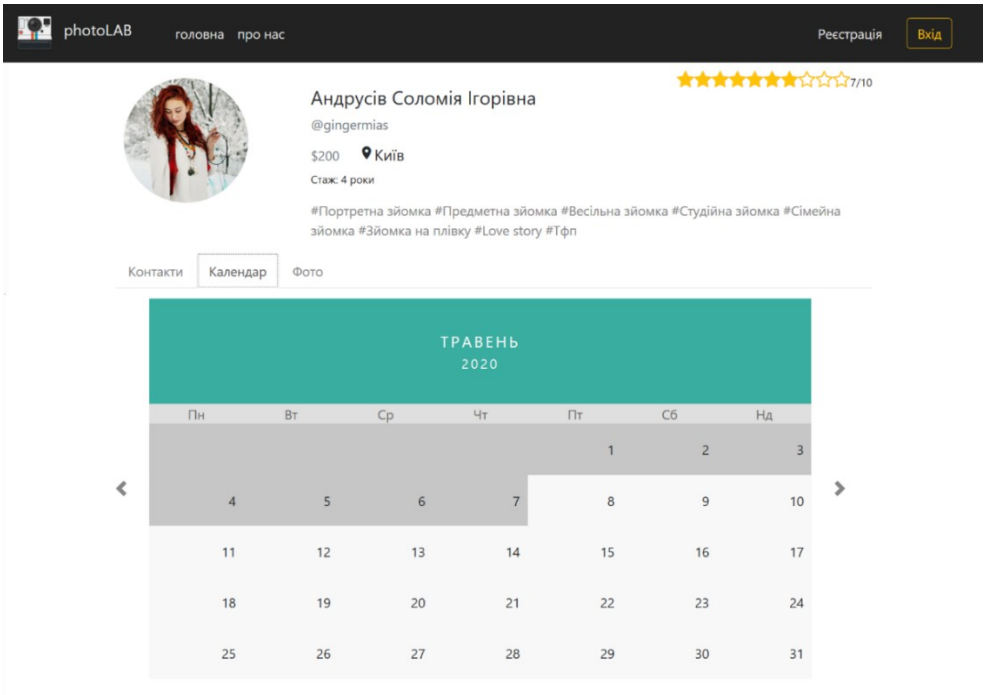


Рисунок 3.9 - Перегляд календаря

- Зареєструватись в системі(див. рис. 3.10);

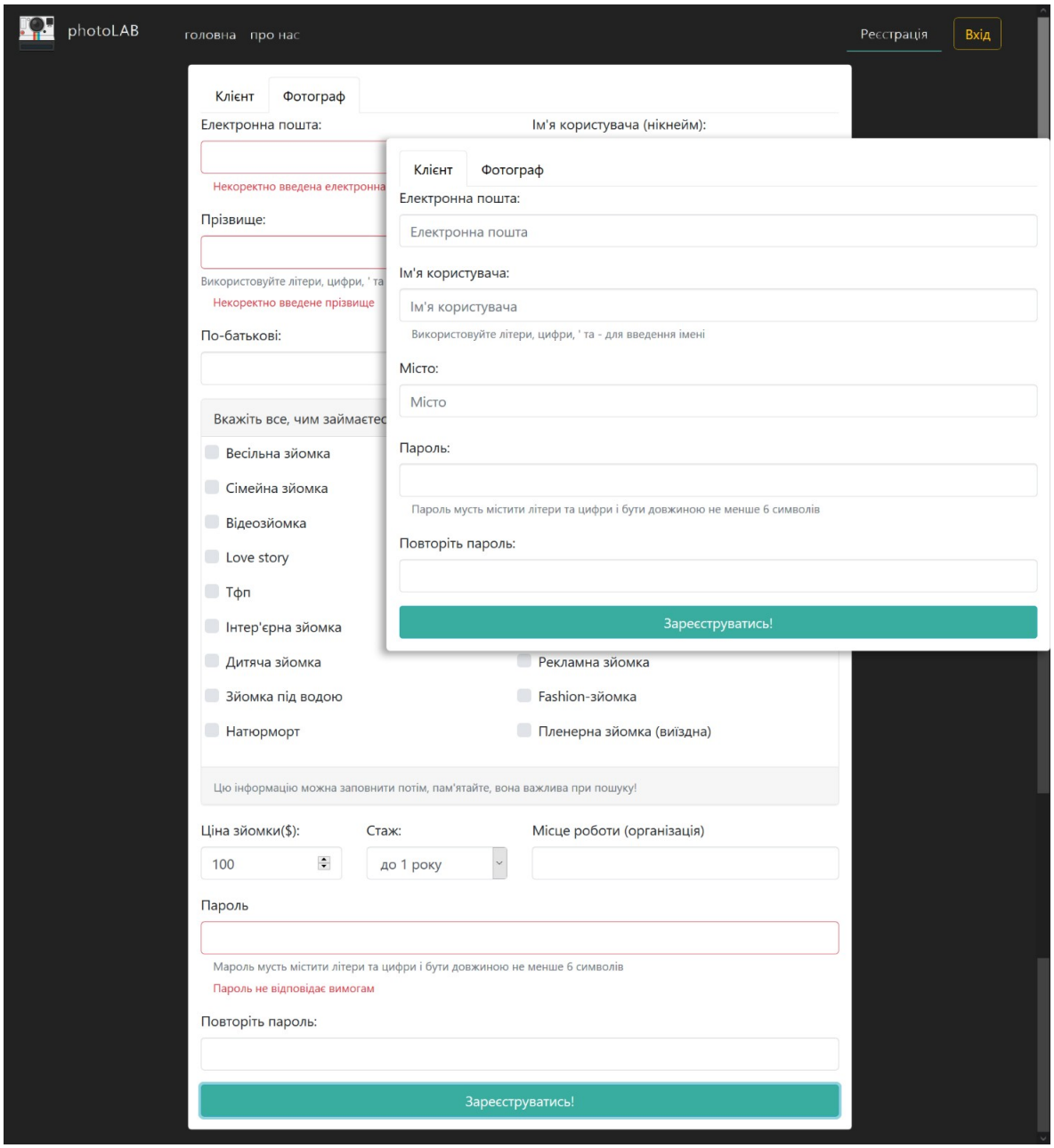


Рисунок 3.10 - Реєстрація фотографа та клієнта

- Переглядати сторінку «Про нас» з статичною інформацією про програму (див. рис. 3.11).

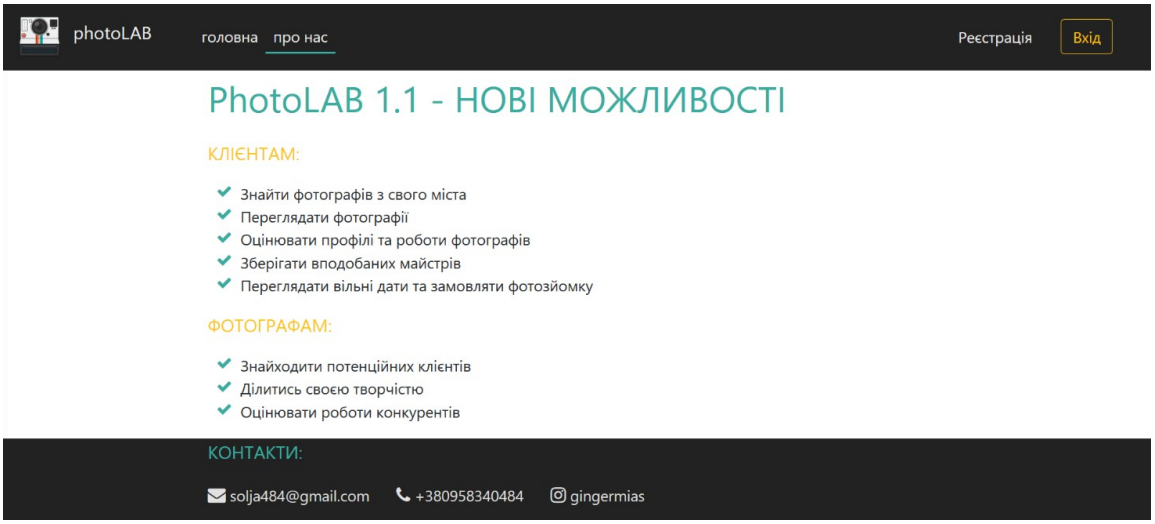


Рисунок 3.11 - Сторінка "Про нас"

Графічний інтерфейс та функціонал сайту для зареєстрованих користувачів типів «Адміністратор», «Клієнт» та «Фотограф» представлено у додатках Г, Д та Е відповідно.

ВИСНОВКИ

В ході курсової роботи було розглянуто та проаналізовано вибрану предметну область. Практичну частину було вирішено реалізувати у вигляді веб-сайту.

Досліджено найбільш популярні та широковживані технології, що використовуються при розробці веб-застосувань. Вибрано ті, що дозволяють найкращим способом реалізувати повний функціонал веб-сервісу вибраного типу для курсового проекту.

Здійснено пошук існуючих аналогів розробки. При проведенні аналізу їх роботи та функціональних можливостей визначено закономірності, що спостерігаються у всіх веб-застосуваннях такого типу та додано до функціоналу курсового проекту.

Розроблено онлайн-платформу для пошуку та замовлення професійної фотозйомки, в якій реалізовано власне замовлення зйомки та пошук фото і фотографів, а також інші корисні можливості, таких як рейтинг та зберігання вподобаних профілів.

У результаті виконання роботи створено програмний застосунок з зручним та інтуїтивно-зрозумілим інтерфейсом, в якому користувачі можуть здійснювати пошук та замовлення фотозйомки, а фотографи – ділитися своїми роботами та бути знайденими потенційними клієнтами.

Проект є унікальним в галузі комерційної фотографії в Україні. Він може використовуватися фотографами та аматорами для пошуку замовників та заробітку на своїй творчості. А також клієнтами, що не обов’язково близько знайомі з креативною індустрією, для замовлення фотозйомки, пошуку фотографа та оцінювання якості його роботи. В основному йдеться про замовлення фотографа для подій – весілля, днів народження, випускних тощо.

Проект має багато можливостей для подальшого розвитку та вдосконалення. До прикладу, можна додати засоби електронних платежів чи вдосконалити пошук фото з використанням нейронних мереж. Проте варто

пам’ятати, що фотозйомка як послуга – річ дуже індивідуальна і в більшості випадків її результати залежать від комунікації між фотографом та моделями, тобто від особистих людських якостей, а не тільки від професійних навичок. Тому повністю автоматизувати даний процес та уникнути живого спілкування між замовником та постачальником послуги неможливо.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Розпорядження про затвердження видів економічної діяльності, які належать до креативних індустрій [Електронний ресурс]. Режим доступу: https://www.kmu.gov.ua/npas/pro-zatverdzchnoyi-diyalnosti-yaki-nalezhat-do-kreativnih-industrij
2. Українська фотографія. Інтерв'ю з засновником однойменного ресурсу, Сергієм Топольницьким [Електронний ресурс]. Режим доступу: http://martatrotsiuk.com/archives/7491
3. Аналог розробки: Behance. [Електронний ресурс]. Режим доступу: https://www.behance.net/JOBLIST
4. Аналог розробки: Flickr [Електронний ресурс]. Режим доступу: https://www.flickr.com/
5. Аналог розробки: Instagram [Електронний ресурс]. Режим доступу: https://www.instagram.com
6. Аналог розробки: Olx [Електронний ресурс]. Режим доступу: https://www.olx.ua/uk/
7. Selcuk Cebi, Determining importance degrees of website design parameters based on interactions and types of websites, Decision Support Systems 54 (2013) 1031-1032 (переглянути в pdf).
8. Стаття «Жанри фотографії» [Електронний ресурс]. Режим доступу: http://fotoshpora.blogspot.com/2013/11/blog-post_582.html
9. Вікіпедія. Список статей про існуючі жанри фотографії [Електронний ресурс]. Режим доступу: https://en.wikipedia.org/wiki/Category:Photography_by_genre
10. Фотоблог. Стаття «Основні жанри фотографії як мистецтва» [Електронний ресурс]. Режим доступу: https://natalie-berta.livejournal.com/8435.html

ДОДАТКИ

Додаток А

(довідниковий)

Список видів зйомок

1. Портретна зйомка.
2. Предметна зйомка.
3. Весільна зйомка.
4. Студійна зйомка.
5. Сімейна зйомка.
6. Святкова зйомка.
7. Відеозйомка.
8. Зйомка на плівку.
9. Love story – романтична зйомка закоханих пар.
10. Модельні тести – зйомка що має на меті заповнення портфоліо для фотомоделі.
11. Tfp(Time for Photo) – вид безплатної зйомки, яку фотографи проводять з метою практичних тренувань, випробувань нової техніки, реалізації власних ідей тощо.
12. Стиль ню, еротичні зйомки.
13. Інтер'єрна зйомка.
14. Вулична зйомка(street style) – вид зйомки, що відбувається на вулицях великих міст.
15. Дитяча зйомка.
16. Рекламна зйомка.
17. Зйомка під водою.
18. Fashion-зйомка.
19. Натюрморт.
20. Пленерна зйомка – зйомка на природі з використанням естетичного освітлення, в розмовній мові називають «зйомкою на виїзд».

Додаток Б

(довідниковий)

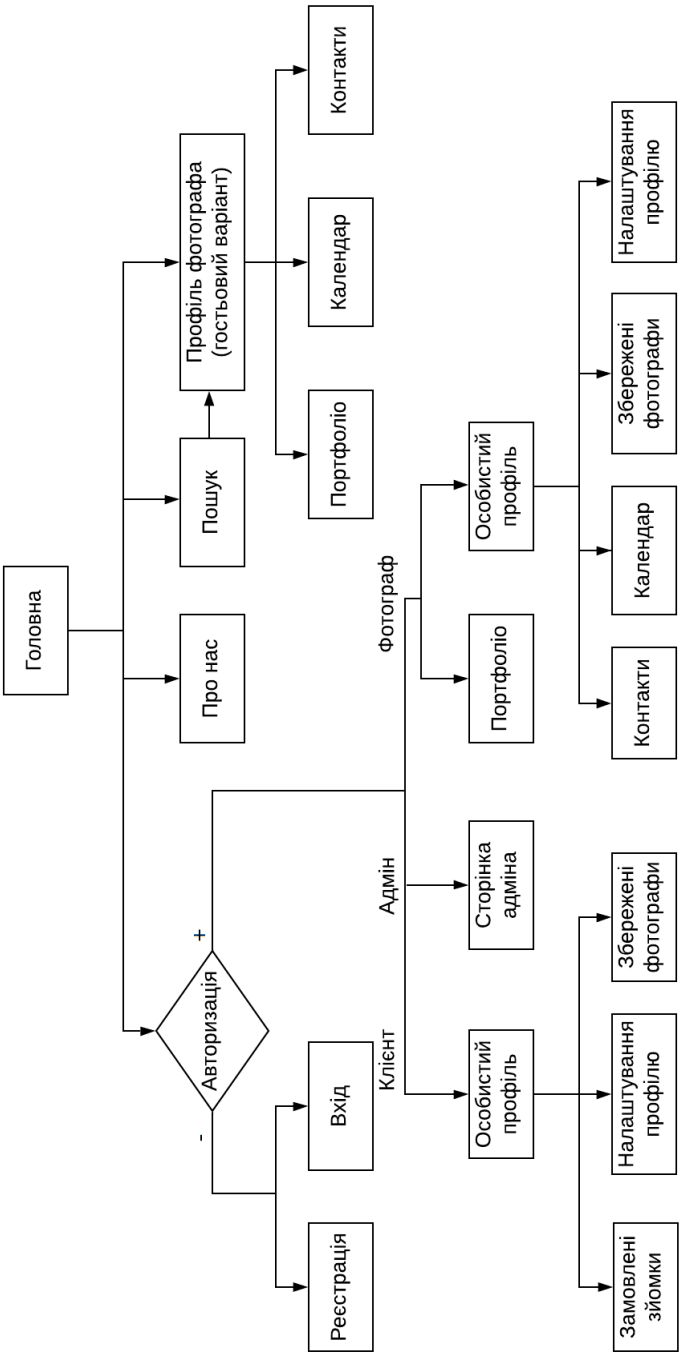


Рисунок Б.1 Схема роботи сайту

Додаток В

(довідниковий)

Діаграма моделі бази даних

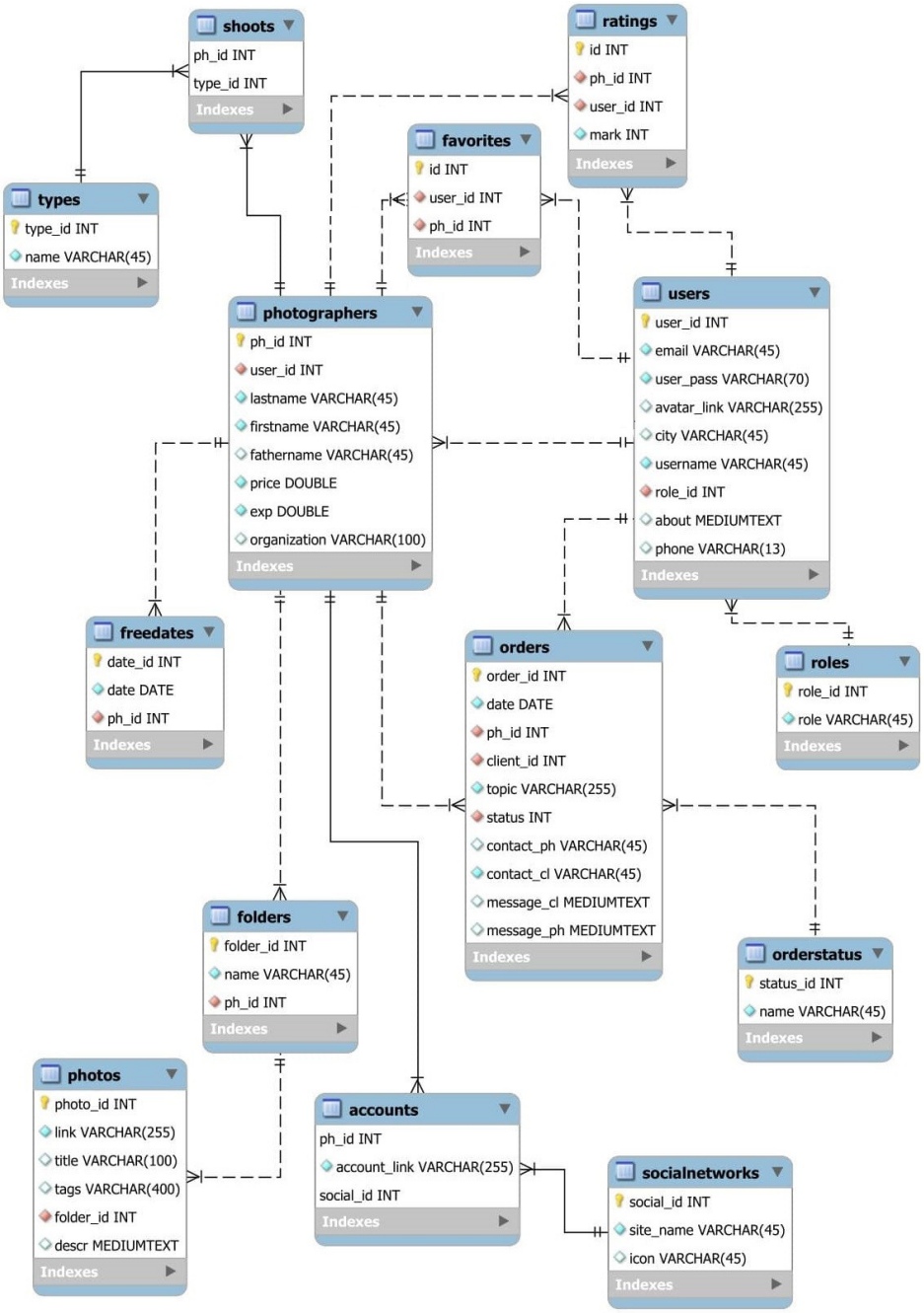


Рисунок В.1 - Схема таблиць бази даних

Додаток Г

(довідниковий)

Графічний інтерфейс користувача «Адміністратор»

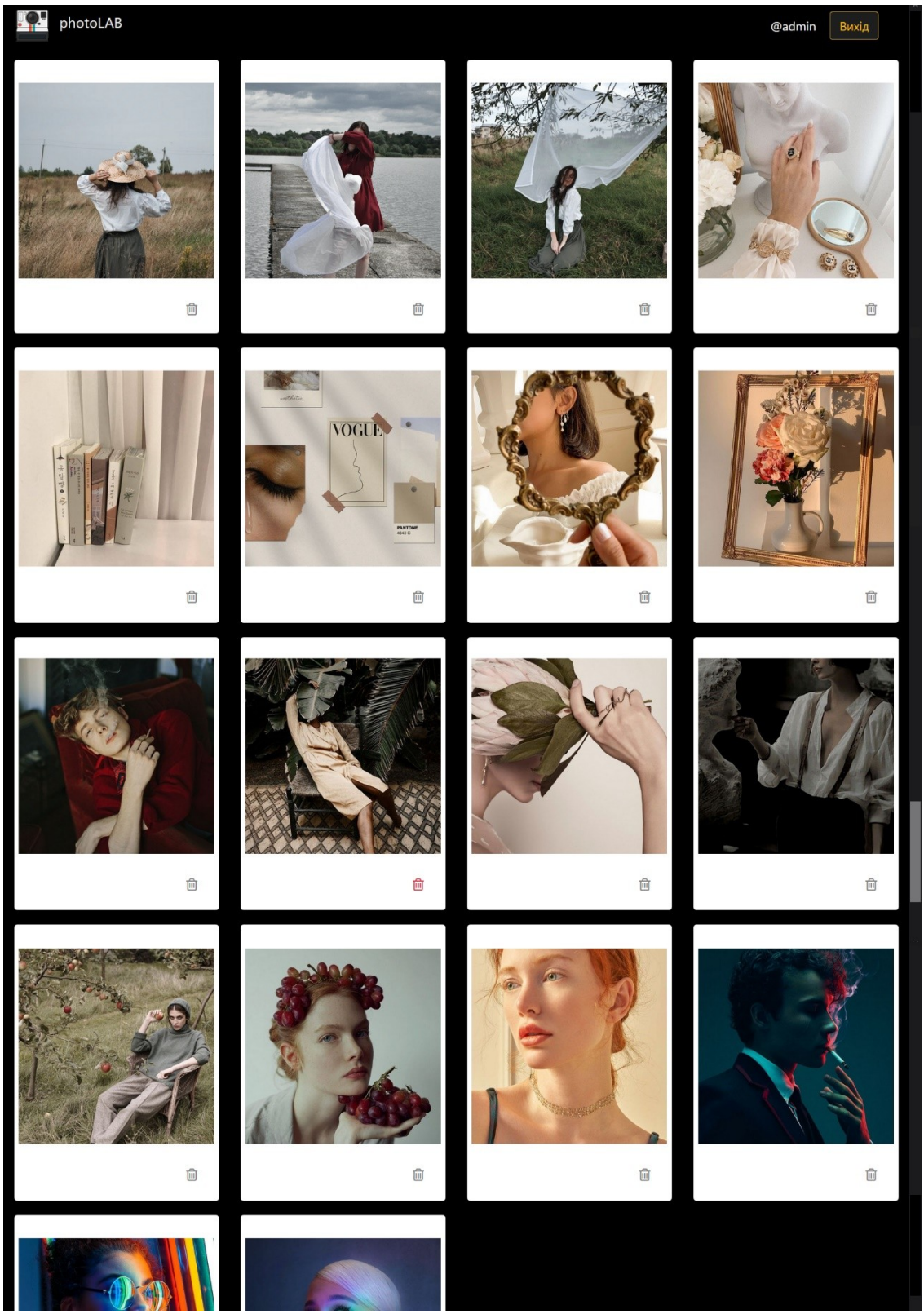


Рисунок Г.1 - Сторінка адміністратора

Додаток Д

(довідниковий)

Графічний інтерфейс користувача «Клієнт»

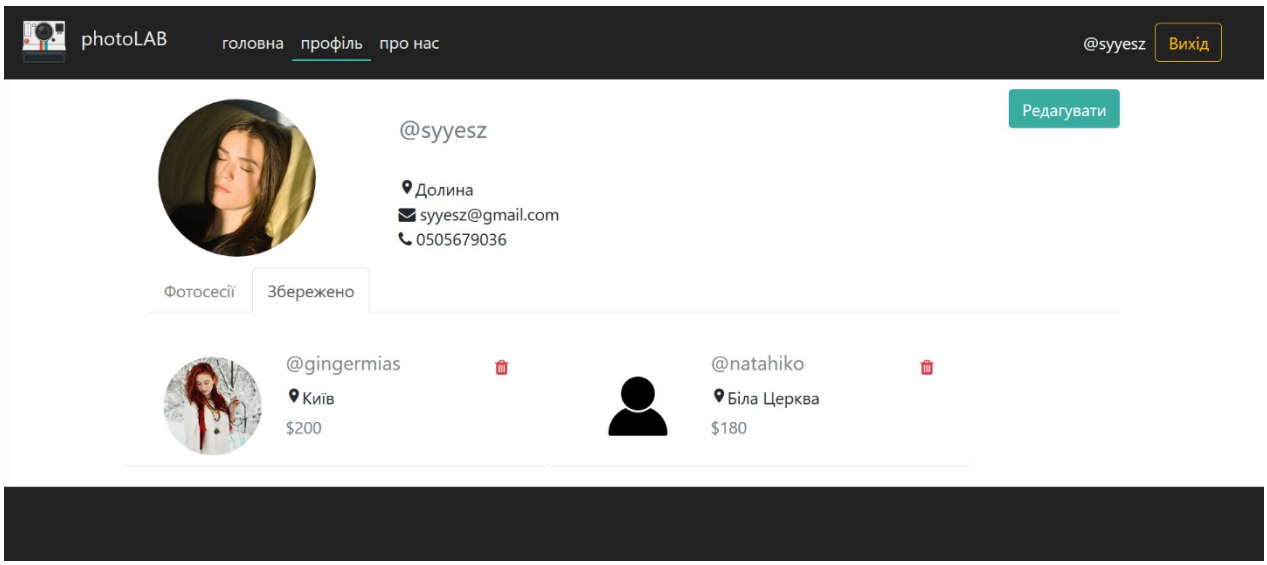


Рисунок Д.1 - Профіль клієнта. Вкладка закладок.

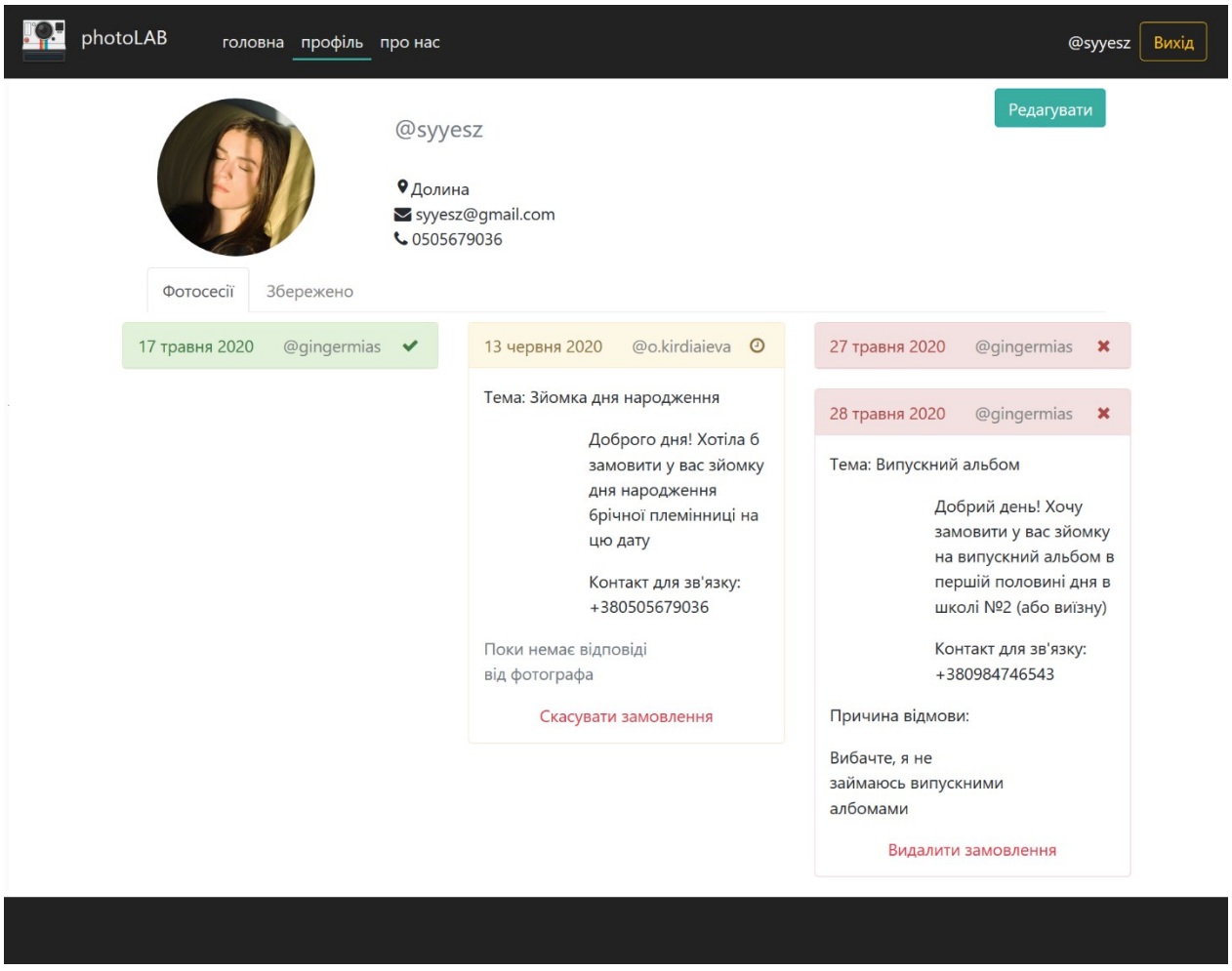


Рисунок Д.2 - Профіль клієнта. Вкладка замовлень.

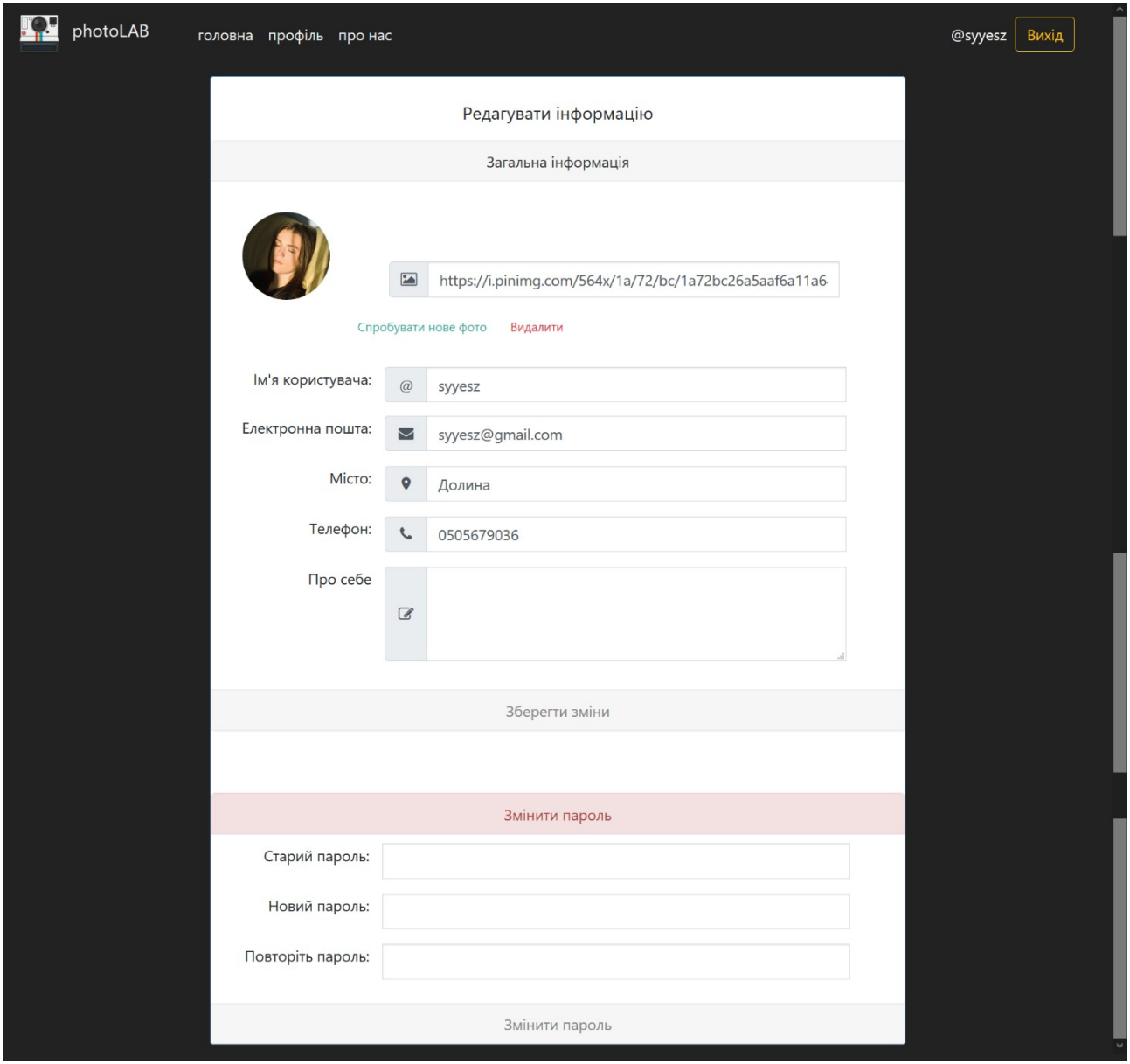

photoLAB
головна
профіль
про нас
@syyesz
Вихід
Редагувати інформацію
Загальна інформація
https://i.pinimg.com/564x/1a/72/bc/1a72bc26a5aaf6a11a6
Спробувати нове фото
Видалити
Ім'я користувача:
@
syyesz
Електронна пошта:
syyesz@gmail.com
Місто:
Долина
Телефон:
0505679036
Про себе
Зберегти зміни
Змінити пароль
Старий пароль:
Новий пароль:
Повторіть пароль:
Змінити пароль

Рисунок Д.3 - Редагування інформації клієнта.

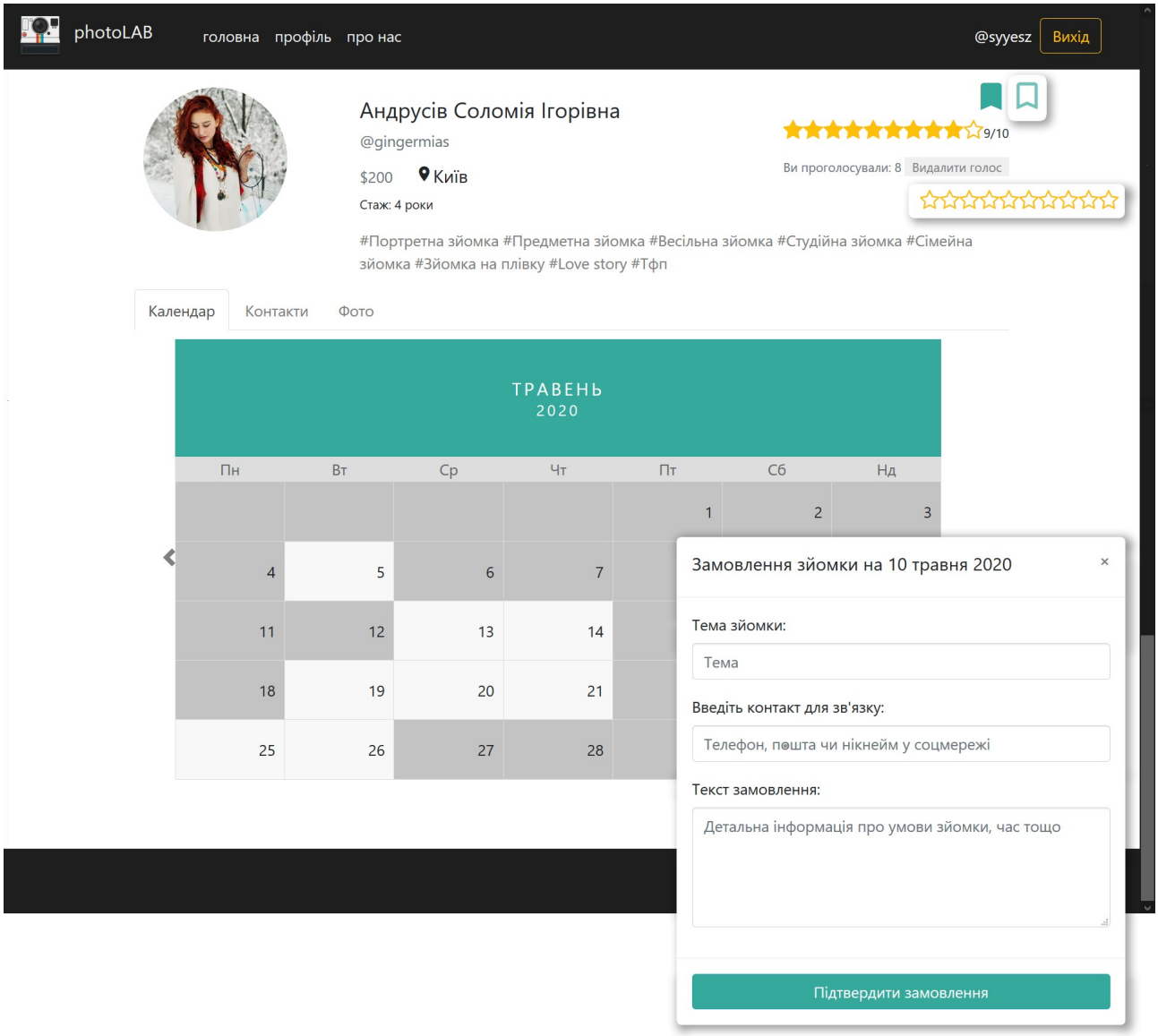


Рисунок Д.4 – Всі можливі дії користувача на профілі фотографа.

Додаток Е

(довідниковий)

Графічний інтерфейс користувача «Фотограф»

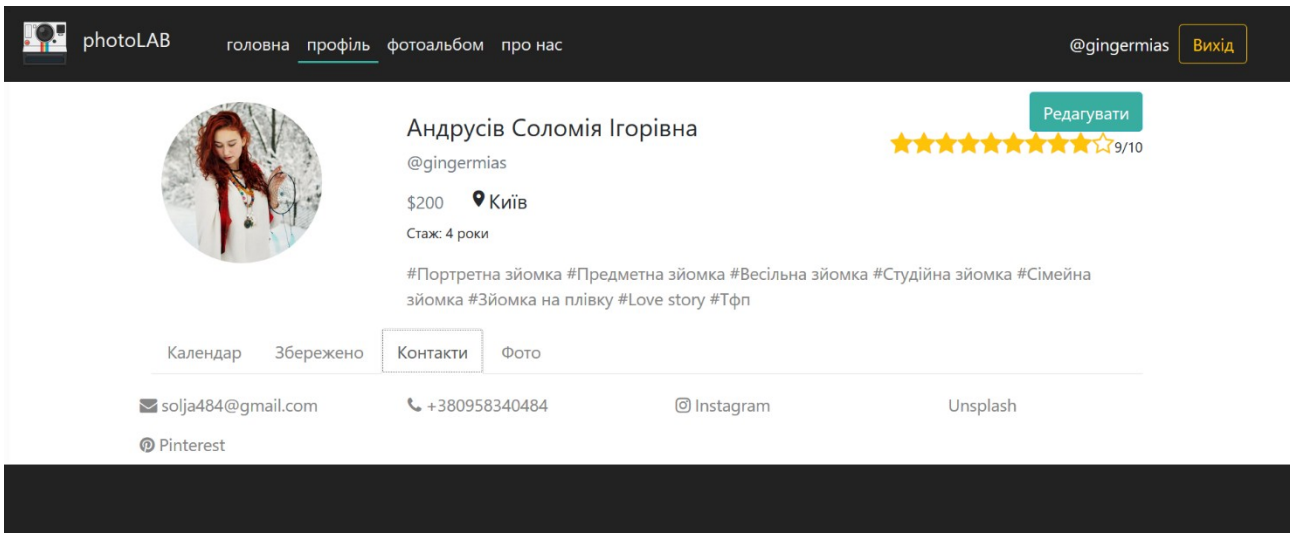


Рисунок Е.1 - Профіль фотографа. Вкладка контактів.

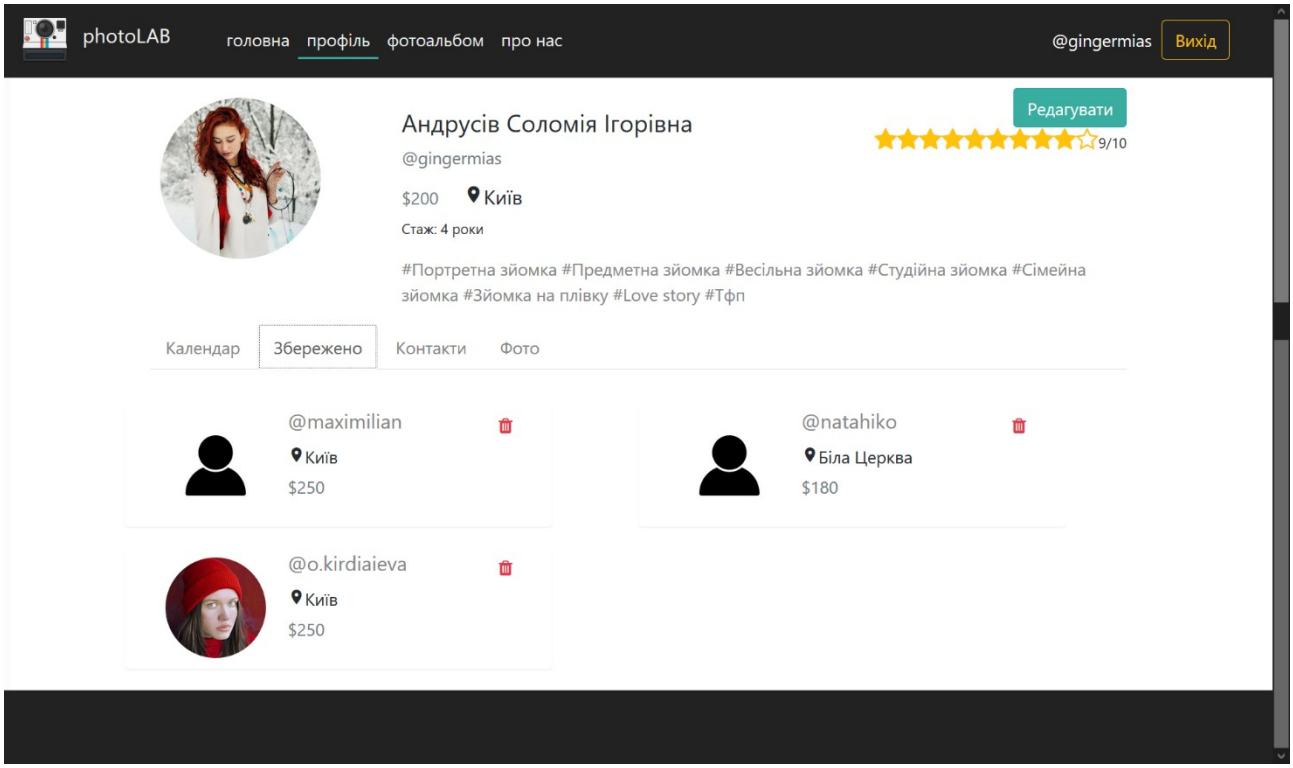


Рисунок Е-2 - Профіль фотографа. Вкладка закладок.

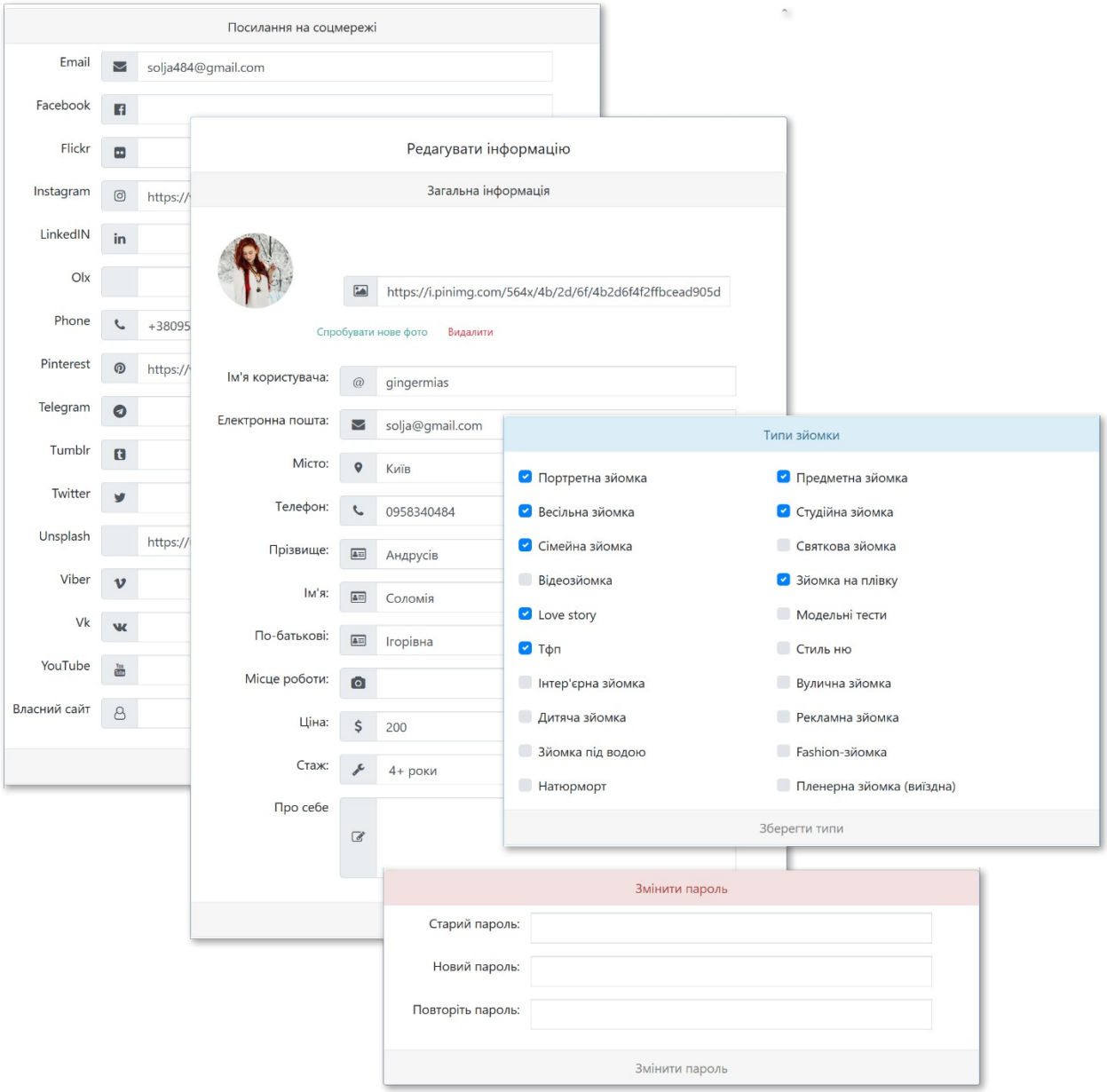


Рисунок Е.3 - Редагування особистої інформації фотографа

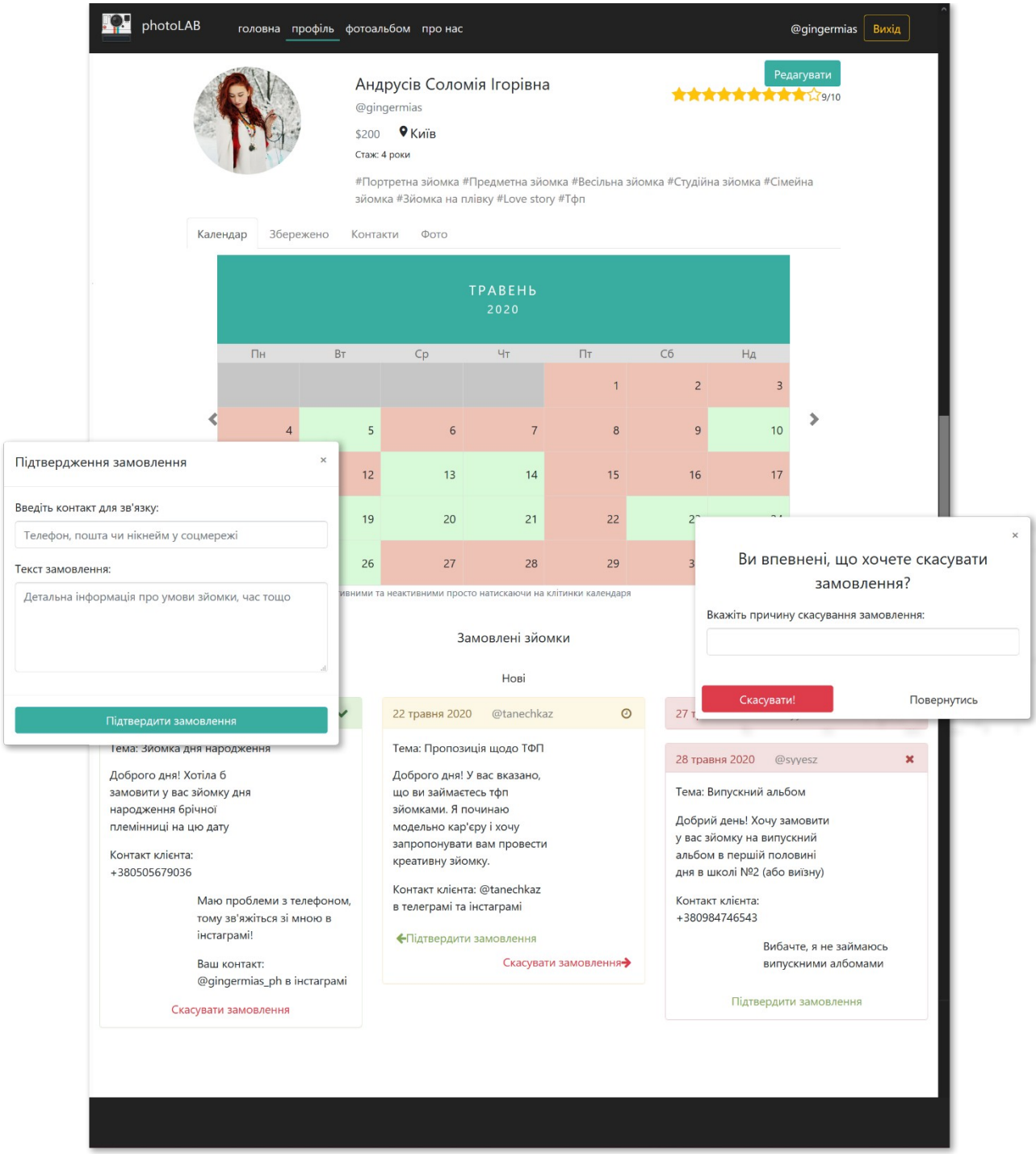


Рисунок Е.4- Сторінка замовлень

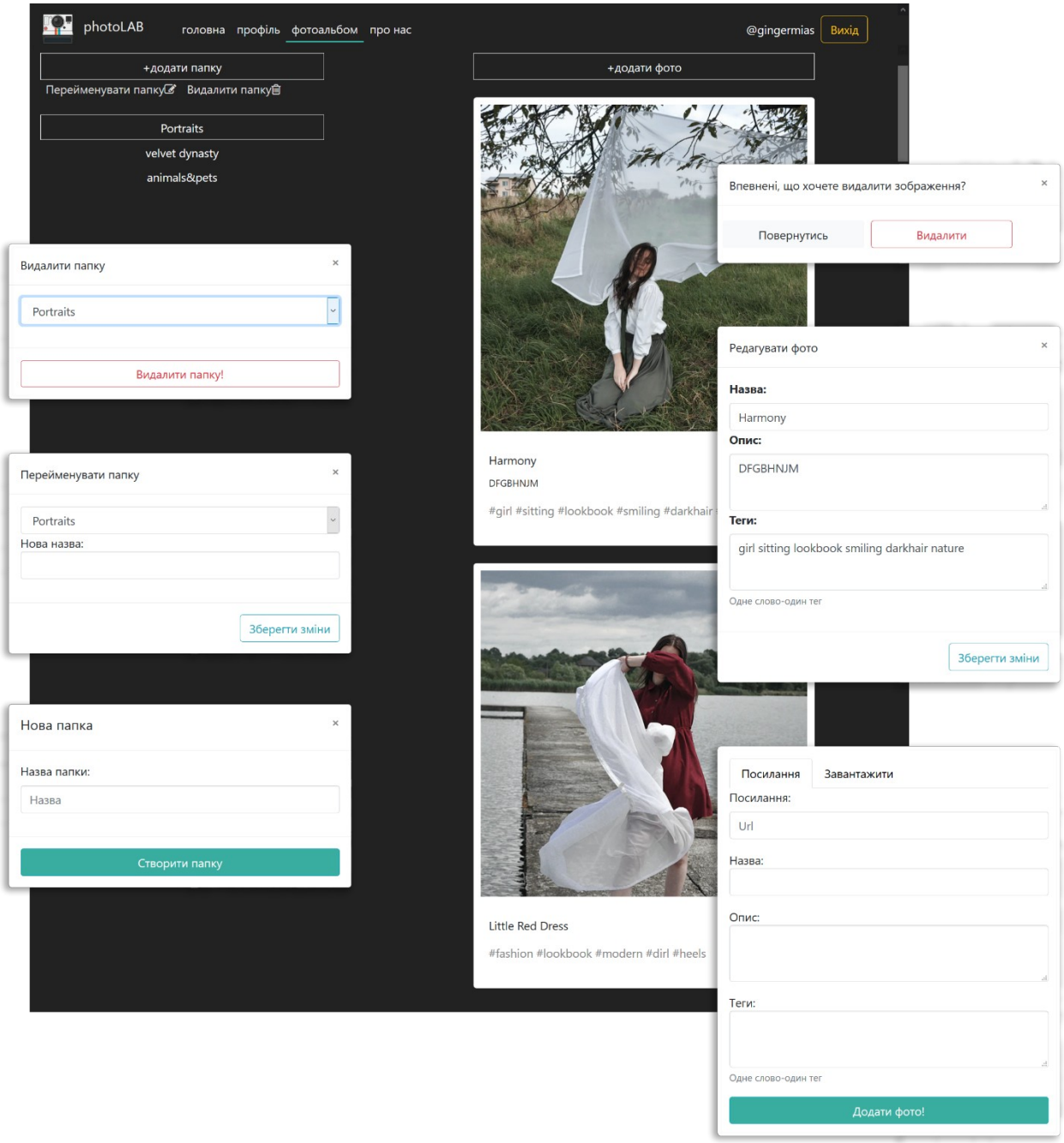


Рисунок Е.5 - Сторінка фото та організація роботи з ними

Додаток Ж

(обов’язковий)

Програмний код(серверна частина)

server.js

```
//include modules
let express = require('express');
let server = express();
let bodyParser = require('body-parser');
let config = require('./config.json');
let cookieParser = require('cookie-parser');
const mysql = require("mysql2");
let crypto = require("crypto");
let custom = require('./public/javascripts/custom_modules.js');
let types = [];

//connect pug
server.set("view engine", "pug");
server.set('views', './');

server.use(express.static(__dirname));
server.use(bodyParser.json());
server.use(bodyParser.urlencoded({extended: true}));
server.use(cookieParser());

//start server
server.listen(2606, () => {
    console.log('listening on 2606')
});

//connect to mysql database
const connection = mysql.createConnection({
    host: config.host,
    user: config.user,
    database: config.database,
    password: config.password
}).promise();
```

```
//create inverted list for search
function createInvertedList() {
    connection.query(
        "SELECT tags,photo_id as id FROM photos"
    ).then(([results, fields]) => {
        for (let r in results)
            results[r].taglist = custom.splitIntoTags(results[r].tags);
        custom.setInvertedList(results);
    })
        .catch(err => {
            console.log(err);
        });
}
createInvertedList();

//select all types of photoshoots
connection.query("SELECT name,type_id FROM types", function (err, results,
fields) {
    types = results;
});

//page with static information about site
server.get('/about', function (req, res) {
    let active = 'about';
    res.cookies = req.cookies;
    res.render(__dirname + "/views/about.pug", {
        active: active,
        config: config,
        cookies: req.cookies
    });
});

//main page of authorized and unauthorized user
server.get('/home/:username', (req, res) => {
    res.cookies = req.cookies;
    let username = req.params.username;
    if ((req.cookies.auth == 'false') && username == req.cookies.username)
```

```
        res.redirect('/');
    if (username != req.cookies.username)
        res.redirect('/guest/' + username);
    let active = 'home';
    let guest = false;
    res.cookie("user_id", req.cookies.user_id);
    connection.query("SELECT photographers.ph_id,
photographers.user_id,username,city, avatar_link, price,  exp,organization,
ROUND(AVG(mark)) AS ave FROM ( photographers INNER JOIN users ON
photographers.user_id=users.user_id)
LEFT JOIN ratings ON photographers.ph_id=ratings.ph_id BY photographers.ph_id
ORDER BY ave DESC, username ASC"
    ).then(([results2, fields2]) => {
        connection.query(
            "SELECT user_id,ph_id FROM favorites WHERE user_id=? GROUP BY
ph_id", [req.cookies.user_id]
        ).then(([results3, fields3]) => {
            connection.query(
                "SELECT link, ph_id FROM photos INNER JOIN folders ON
folders.folder_id=photos.folder_id ORDER BY ph_id ASC, photo_id DESC"
            ).then(([results4, fields4]) => {
                res.render(__dirname + '/views/home.pug',
                    {
                        types: types,
                        phinfo: results2,
                        favorites: results3,
                        photos: results4,
                        active: active,
                        guest: guest,
                        config: config,
                        cookies: res.cookies
                    });
            })
        })
            .catch(err => {
                console.log(err);
            });
    })
        .catch(err => {
            console.log(err);
```

```
        });
});
server.get('/', function (req, res) {
    let active = 'home';
    if (req.cookies.auth == 'true') {
        res.cookies = req.cookies;
        if (req.cookies.role != 'admin')
            res.redirect('/home/' + req.cookies.username);
        if (req.cookies.role == 'admin')
            res.redirect('/admin');
    }
    let noun = "";
    connection.query("SELECT photographers.ph_id,
photographers.user_id,username,city,
avatar_link,price,exp,organization,ROUND(AVG(mark)) AS ave FROM ( photographers
INNER JOIN users ON photographers.user_id=users.user_id) LEFT JOIN ratings ON
photographers.ph_id=ratings.ph_id GROUP BY photographers.ph_id ORDER BY ave
DESC, username ASC")
        .then(([results, fields]) => {
            connection.query(
                "SELECT ph_id,link FROM photos INNER JOIN folders ON
photos.folder_id=folders.folder_id ")
                .then(([results2, fields2]) => {
                    res.cookies = req.cookies;
                    res.render(__dirname + '/views/home.pug',
                        {
                            types: types,
                            phinfo: results,
                            photos: results2,
                            auth: res.cookies.auth,
                            active: active,
                            config: config,
                            favorites: res.cookies,
                            cookies: res.cookies
                        });
                })
                .catch(err => {
                    console.log(err);
                });
        })
```

```
        .catch(err => {
            console.log(err);
        });
});


//get information and render photographer profile when user visit it
server.get('/guest/:username', (req, res) => {
    let guest_username = req.params.username;
    res.cookies = req.cookies;
    let active = '';
    let guest = true;
    if (guest_username == null)
        res.redirect('/');
    connection.query(" SELECT photographers.ph_id AS ph_id,photographers.user_id
AS user_id,email,avatar_link,city, organization, username, lastname, firstname,
fathername, price, exp, organization, about, ROUND(AVG(mark)) AS ave FROM
( photographers INNER JOIN users ON photographers.user_id =users.user_id ) LEFT
JOIN ratings ON photographers.ph_id=  ratings.ph_id WHERE username = ? GROUP BY
ph_id ", [guest_username])
        .then(([results, fields]) => {
            let ph_id = results[0].ph_id;
            let info = results[0];
            if (info.ave == null)
                info.ave = 0;
            else info.ave = parseInt(info.ave, 10);
            connection.query("SELECT date FROM freedates WHERE ph_id=?",
[ph_id])
                .then(([results1, fields1]) => {
                    connection.query("SELECT name FROM types INNER JOIN shoots
ON types.type_id=shoots.type_id WHERE ph_id=?", [ph_id])
                        .then(([results2, fields2]) => {
                            connection.query("SELECT account_link,site_name,icon
FROM accounts " +
                                "INNER JOIN socialnetworks ON
socialnetworks.social_id=accounts.social_id WHERE ph_id=? ORDER BY
socialnetworks.social_id", [ph_id])
                                .then(([results3, fields3]) => {
                                    res.cookies = req.cookies;
                                    if (req.cookies.auth == 'false') {
                                        res.render(__dirname +
"/views/profileph.pug",
                                            {
```

```
                                                mytypes: results2,
                                                contacts: results3,
                                                info: results[0],
                                                active: active,
                                                config: config,
                                                guest: guest,
                                                username: req.params.username,
                                                auth: req.cookies.auth,
                                                cookies: req.cookies,
                                                dates: results1
                                            });
                                    } else {

                                        connection.query("SELECT mark FROM
ratings WHERE user_id=? AND ph_id=?  ", [req.cookies.user_id, ph_id])
                                            .then(([results4, fields4]) => {

                                                try {
                                                    info.mark =
results4[0].mark;
                                                } catch (err) {
                                                    info.mark = 0;
                                                }
                                                connection.query("SELECT * FROM
favorites WHERE user_id=? AND ph_id=?", [req.cookies.user_id, ph_id])
                                                    .then(([results5, fields5])
=> {

                                                        let favorite = true;
                                                        if (results5 == [] ||
results5[0] == undefined)
                                                            favorite = false;

                                                        res.render(__dirname +
"/views/profileph.pug",
                                                            {
                                                                mytypes:
results2,
                                                                contacts:
results3,
                                                                info: info,
```

```
                                                                active: active,
                                                                config: config,
                                                                guest: guest,
                                                                favorite:
favorite,
                                                                username:
req.params.username,
                                                                auth:
req.cookies.auth,
                                                                cookies:
req.cookies,
                                                                dates: results1
                                                            });
                                                        })
                                                        .catch(err => {
                                                            console.log(err);
                                                        });
                                                })
                                                .catch(err => {
                                                    console.log(err);
                                                });
                                        }
                                    })
                                    .catch(err => {
                                        console.log(err);
                                    });
                            })
                            .catch(err => {
                                console.log(err);
                            });
                    })
                    .catch(err => {
                        console.log(err);
                    });
            })
            .catch(err => {
                console.log(err);
            });
```

```
});

server.get('/guest/photoalbum/:username', (req, res) => {

    let guest_username = req.params.username;

    let active = '';

    let guest = true;

    res.cookies = req.cookies;

    if (guest_username == req.cookies.username)

        res.redirect('/photoalbum/' + guest_username);

    connection.query("SELECT ph_id FROM photographers INNER JOIN users on
photographers.user_id= users.user_id WHERE username=?", [guest_username])

        .then(([results, fields]) => {

            let ph_id = results[0].ph_id;

            connection.query("SELECT link,title,tags, descr, photos.folder_id
FROM photos INNER JOIN folders ON folders.folder_id=photos.folder_id  WHERE
ph_id =? ORDER BY folders.folder_id ASC, photo_id DESC", [ph_id])

                .then(([results2, fields2]) => {

                    for (p in results2)

                        if (results2[p].tags == null)

                            results2[p].taglist = [];

                        else

                            results2[p].taglist =
custom.splitIntoTags(results2[p].tags);

                    connection.query("SELECT folder_id,name FROM folders WHERE
ph_id =? GROUP BY folder_id ORDER BY folder_id", [ph_id])

                        .then(([results3, fields3]) => {

                            res.render(__dirname + "/views/photoalbum.pug", {

                                config: config,

                                guest: guest,

                                ph_username: guest_username,

                                photos: results2,

                                folders: results3,

                                active: active,

                                cookies: req.cookies


                            });

                        })

                        .catch(err => {

                            console.log(err);

                        });

                })
```

```
                .catch(err => {
                    console.log(err);
                });
        })
        .catch(err => {
            console.log(err);
        });
});

//show page with user information which he can edit
server.get('/edit/:username', (req, res) => {
    let username = req.params.username;
    if (req.cookies.username !== username)
        res.redirect('/guest/' + username);
    let active = "";
    res.cookies = req.cookies;
    if (req.cookies.role == 'client')
        connection.query("SELECT * FROM users WHERE username=?", [username])
            .then(([results, fields]) => {
                connection.query("SELECT username, email FROM users WHERE
username<>?", [username])
                    .then(([results2, fields2]) => {
                        res.render(__dirname + "/views/edit.pug", {
                            active: active,
                            auth: req.cookies.auth,
                            config: config,
                            cookies: req.cookies,
                            info: results[0],
                            users: results2
                        });
                    })
                    .catch(err => {
                        console.log(err);
                    });
            })
            .catch(err => {
                console.log(err);
            });
```

```
    else if (req.cookies.role == 'photographer')

        connection.query("SELECT * FROM users INNER JOIN photographers ON
users.user_id = photographers.user_id WHERE username=?", [username])

            .then(([results, fields]) => {

                let ph_id = results[0].ph_id;

                connection.query("SELECT username, email FROM users WHERE
username<>?", [username])

                    .then(([results2, fields2]) => {

                        connection.query("SELECT shoots.type_id, name, ph_id
FROM types INNER JOIN shoots ON types.type_id=shoots.type_id WHERE ph_id=?",
[ph_id])

                            .then(([results3, fields3]) => {

                                connection.query("SELECT * FROM accounts WHERE
ph_id=?", [ph_id])

                                    .then(([results4, fields4]) => {

                                        connection.query("SELECT * FROM
socialnetworks ORDER BY site_name")

                                            .then(([results5, fields5]) => {

                                                results[0].role =
req.cookies.role;

                                                res.render(__dirname +
"/views/edit.pug", {

                                                    active: active,

                                                    config: config,

                                                    social: results5,

                                                    accounts: results4,

                                                    ph_types: results3,

                                                    types: types,

                                                    cookies: req.cookies,

                                                    info: results[0],

                                                    users: results2

                                                });

                                            })

                                            .catch(err => {

                                                console.log(err);

                                            });

                                    })

                                    .catch(err => {

                                        console.log(err);

                                    });

                            })
```

```
                            .catch(err => {
                                console.log(err);
                            });
                    })
                    .catch(err => {
                        console.log(err);
                    });
            })
            .catch(err => {
                console.log(err);
            });
});


//show user profile and photo album(if photographer)
server.get('/photoalbum/:username', function (req, res) {
    let username = req.params.username;
    if (req.cookies.auth == 'false' && req.cookies.username != username)
res.redirect('/guest/photoalbum/' + username);


    let active = 'album';
    let guest = false;
    res.cookies = req.cookies;
    connection.query("SELECT ph_id FROM photographers INNER JOIN users on
photographers.user_id= users.user_id WHERE username=?", [username])
        .then(([results, fields]) => {
            let ph_id = results[0].ph_id;
            connection.query("SELECT photo_id, link, title,  tags, descr,
photos.folder_id FROM photos INNER JOIN folders ON folders.folder_id =
photos.folder_id WHERE ph_id =? ORDER BY folders.folder_id ASC, photo_id DESC",
[ph_id])
                .then(([results2, fields2]) => {
                    connection.query("SELECT folder_id,name FROM folders WHERE
ph_id =? GROUP BY folder_id ORDER BY folder_id", [ph_id])
                        .then(([results3, fields3]) => {
                            for (p in results2)
                                results2[p].taglist =
custom.splitIntoTags(results2[p].tags);
                            res.render(__dirname + "/views/photoalbum.pug", {
                                config: config,
                                guest: guest,
```

```
                                photos: results2,
                                folders: results3,
                                active: active,
                                cookies: req.cookies
                            });
                        })
                        .catch(err => {
                            console.log(err);
                        });
                })
                .catch(err => {
                    console.log(err);
                });
        })
        .catch(err => {
            console.log(err);
        });
});
server.get('/profile/:username', function (req, res) {
    let username = req.params.username;
    res.cookies = req.cookies;
    if (req.cookies.auth == 'false') {
        if (username == req.cookies.username)
            res.redirect('/');
        else res.redirect('/guest/' + req.cookies.username);
    }
    if (username != req.cookies.username)
        res.redirect('/guest/' + req.cookies.username);


    let active = 'profile';
    let guest = false;
    if (req.cookies.role == 'photographer') {
        connection.query(" SELECT photographers.ph_id AS
ph_id,photographers.user_id AS user_id,email,avatar_link,city, organization,
username, lastname, firstname, fathername, price, exp, organization, about,
ROUND(AVG(mark)) AS ave FROM ( photographers INNER JOIN users ON
photographers.user_id=users.user_id ) LEFT JOIN ratings ON
photographers.ph_id=ratings.ph_id WHERE username = ? GROUP BY ph_id ",
[username])
            .then(([results, fields]) => {
```

```
                let ph_id = results[0].ph_id;
                connection.query("SELECT date FROM freedates WHERE ph_id=?",
[ph_id])
                    .then(([results1, fields1]) => {
                        connection.query("SELECT name FROM types INNER JOIN
shoots ON types.type_id=shoots.type_id WHERE ph_id=?", [ph_id])
                            .then(([results2, fields2]) => {
                                if (results2 == undefined)
                                    results2 = [];
                                connection.query("SELECT
account_link,site_name,icon FROM accounts INNER JOIN socialnetworks ON
socialnetworks.social_id=accounts.social_id WHERE ph_id=? ORDER BY
socialnetworks.social_id", [ph_id])
                                    .then(([results3, fields3]) => {
                                        if (results3[0] == undefined)
                                            results3 = [];
                                        connection.query("SELECT
photographers.ph_id, username, avatar_link, price, city FROM favorites INNER
JOIN photographers ON photographers.ph_id=favorites.ph_id INNER JOIN users ON
users.user_id = photographers.user_id WHERE favorites.user_id=? ORDER BY
username", [results[0].user_id])
                                            .then(([results4, fields4]) => {
                                                if (results4[0] == undefined)
                                                    results4 = [];



                                                let info = results[0];
                                                if (info.ave == null)
                                                    info.ave = 0;
                                                else info.ave =
parseInt(info.ave, 10);
                                                connection.query("SELECT * FROM
orders INNER JOIN users ON users.user_id=orders.client_id WHERE ph_id=? ORDER BY
date", [ph_id])
                                                    .then(([results5, fields5])
=> {


                                                        res.cookies =
req.cookies;
                                                        res.cookie("user_id",
results[0].user_id);
                                                        res.render(__dirname +
"/views/profileph.pug",
                                                            {
```

```
                                                                    mytypes:
results2,
                                                                    contacts:
results3,
                                                                    favorites:
results4,
                                                                    info: info,
                                                                    active: active,
                                                                    config: config,
                                                                    guest: guest,
                                                                    cookies:
req.cookies,
                                                                    dates: results1,
                                                                    orders: results5
                                                                });

                                                        })
                                                        .catch(err => {
                                                            console.log(err);
                                                        });

                                                })
                                                .catch(err => {
                                                    console.log(err);
                                                });
                                        })
                                        .catch(err => {
                                            console.log(err);
                                        });
                                })
                                .catch(err => {
                                    console.log(err);
                                });
                        })
                        .catch(err => {
```

```
                console.log(err);
            });
    }
    else if (req.cookies.role == 'client') {
        connection.query("SELECT * FROM users WHERE  username=?", [username])
            .then(([results, fields]) => {
                connection.query("SELECT photographers.ph_id, username,
avatar_link, price, city FROM favorites INNER JOIN photographers ON
photographers.ph_id  =favorites.ph_id INNER JOIN users ON users.user_id =
photographers.user_id WHERE favorites.user_id=? ORDER BY username",
[results[0].user_id])
                    .then(([results2, fields2]) => {
                        connection.query("SELECT
order_id,orders.ph_id,client_id,username,date,topic, message_ph,message_cl,
status, contact_ph,contact_cl FROM orders INNER JOIN photographers ON
photographers.ph_id  =orders.ph_id INNER JOIN users ON photographers.user_id=
users.user_id WHERE client_id=? ORDER BY date", [results[0].user_id])
                            .then(([results3, fields3]) => {
                                res.cookies = req.cookies;
                                res.render(__dirname + "/views/profile.pug", {
                                    types: types,
                                    info: results[0],
                                    favorites: results2,
                                    active: active,
                                    config: config,
                                    cookies: req.cookies,
                                    orders: results3
                                });
                            })
                            .catch(err => {
                                console.log(err);
                            });
                    })
                    .catch(err => {
                        console.log(err);
                    });
            })
            .catch(err => {
                console.log(err);
            });
    } else if (req.cookies.role == 'admin') {
```

```
        res.cookies = req.cookies;
        res.redirect('/admin');
    }

});

//show admin page
server.get('/admin', function (req, res) {
    connection.query("SELECT * FROM photos")
        .then(([results, fields]) => {
            res.render(__dirname + "/views/admin.pug", {
                photos: results,
                config: config,
                cookies: req.cookies
            });
        })
        .catch(err => {
            console.log(err);
        });
});

//log in and log out
server.post('/login', (req, res) => {
    const login = req.body.login;
    let pass = req.body.password;
    connection.query("SELECT * FROM users INNER JOIN roles on users.role_id =
roles.role_id WHERE  email=?", [login])
        .then(([results, fields]) => {
            if (results == [])
                res.send({'success': 'no'});
            if (hashing(pass) == results[0].user_pass) {
                res.cookie("role", results[0].role);
                res.cookie("auth", 'true');
                res.cookie("username", results[0].username);
                res.cookie("user_id", results[0].user_id);
                let href = '/';
                if (results[0].role == 'admin')
                    href += 'admin';
```

```
                else href += 'home/' + results[0].username;
                res.send({"success": "yes", "href": href})
            } else {
                console.log("incorrect password");
                res.send({'success': 'no'});
            }
        })
        .catch(err => {
            console.log('no user with this login');
            console.log(err);
            res.send({'success': 'no'});
        });
});
server.post('/logout', function (req, res) {
    res.cookie("auth", 'false');
    res.cookie("username", "");
    res.cookie("user_id", "");
    res.cookie("role", "");
    console.log("logged out");
    res.redirect('/');
});


//registration
server.post('/register', (req, res) => {
    let pass = hashing(req.body.password);
    let username = req.body.username;
    connection.query("INSERT INTO users(username,email,user_pass,city,role_id)
VALUES(?,?,?,?,1)", [username, req.body.email, pass, req.body.city])
        .then(([results, fields]) => {
            res.cookie('role', "client");
            res.cookie('auth', 'true');
            res.cookie('user_id', results.insertId);
            res.cookie('username', username);
            let href = '/profile/' + username;
            res.send({"success": "yes", "href": href});
        })
        .catch(err => {
            console.log(err);
```

```
            res.send({"success": "no"});
        });


});
server.post('/registerph', (req, res) => {
    let pass = hashing(req.body.password);
    let username = req.body.username;
    connection.query("INSERT INTO users(username,email,user_pass,city,role_id)
VALUES(?,?,?,?,2)", [username, req.body.email, pass, req.body.city])
        .then(([results, fields]) => {
            let user_id = results.insertId;
            connection.query("INSERT INTO
photographers(user_id,lastname,firstname,fathername,price,exp,organization) " +
                "VALUES(?,?,?,?,?,?,?)", [user_id, req.body.lastname,
req.body.firstname, req.body.fathername, req.body.price,
                req.body.experience, req.body.organization])
                .then(([results2, fields2]) => {
                    let ph_id = results2.insertId;
                    let types = req.body.types;
                    if (types != []) {
                        for (let i in types)
                            types[i].push(ph_id)
                        connection.query("INSERT INTO shoots(type_id,ph_id)
VALUES ?", [types])
                            .then(([results3, fields3]) => {
                                res.cookie('role', 'photographer');
                                res.cookie('auth', 'true');
                                res.cookie('user_id', user_id);
                                res.cookie('username', username);
                                let href = '/profile/' + username;
                                res.send({"success": "yes", "href": href});
                            })
                            .catch(err => {
                                console.log(err);
                                res.send({"success": "no"});
                            });
                    } else {
                        res.cookie('role', 'photographer');
                        res.cookie('auth', 'true');
```

```
                        res.cookie('user_id', user_id);
                        res.cookie('username', username);
                        let href = '/profile/' + username;
                        res.send({"success": "yes", "href": href});
                    }
                })
                .catch(err => {
                    console.log(err);
                    res.send({"success": "no"});
                });
        })
        .catch(err => {
            console.log(err);
            res.send({"success": "no"});
        });


});
server.get('/registry', function (req, res) {
    let active = 'reg';
    res.cookies = req.cookies;
    connection.query(
        "SELECT * FROM users"
    ).then(([results, fields]) => {
        res.render(__dirname + "/views/registration.pug", {
            types: types,
            active: active,
            config: config,
            users: results
        });
    })
        .catch(err => {
            console.log(err);
        });


});
```

```
//edit profile information
server.post('/changepassword', (req, res) => {
    res.cookies = req.cookies;
    let pass = hashing(req.body.user_pass);
    connection.query("UPDATE users SET user_pass=? WHERE user_id=?", [pass,
req.body.user_id])
        .then(([results, fields]) => {
            res.send({"success": "yes"});
        })
        .catch(err => {
            console.log(err);
            res.send({"success": "no"});
        });
});
server.post('/editclient', (req, res) => {
    connection.query("UPDATE users SET username=?, email=?,
city=?,avatar_link=?,phone=?,about=? WHERE user_id=?",
        [req.body.username, req.body.email, req.body.city, req.body.avatar_link,
req.body.phone, req.body.about, req.body.user_id])
        .then(([results, fields]) => {


            res.cookies = req.cookies;
            res.cookie("username", req.body.username);
            if (req.body.role == 'photographer')
                connection.query("UPDATE photographers SET user_id=?,
firstname=?, lastname=?,fathername=?,price=?,exp=?,organization=? WHERE
ph_id=?",
                    [req.body.user_id, req.body.firstname, req.body.lastname,
req.body.fathername, req.body.price, req.body.experience, req.body.organization,
req.body.ph_id])
                    .then(([results, fields]) => {
                        res.cookie('username', req.body.username);
                        res.send({'success': 'yes'});
                    })
                    .catch(err => {
                        console.log(err);
                        res.send({'success': 'no'});
                    });
            else {
                console.log("Successfully updated user");
```

```
                res.send({'success': 'yes'});
            }
        })
        .catch(err => {
            console.log(err);
            res.send({'success': 'no'});
        });
});
server.post('/editaccounts', (req, res) => {
    res.cookies = req.cookies;
    let values = req.body.values;
    connection.query("DELETE FROM accounts WHERE ph_id=?", [req.body.ph_id])
        .then(([results, fields]) => {
            connection.query("INSERT INTO accounts(social_id,account_link,ph_id)
VALUES ?", [values])
                .then(([results, fields]) => {
                    res.send({"success": "yes"});
                })
                .catch(err => {
                    console.log(err);
                    res.send({"success": "no"});
                });
        })
        .catch(err => {
            console.log(err);
            res.send({"success": "no"});
        });
});
server.post('/edittypes', (req, res) => {
    res.cookies = req.cookies;
    let values = req.body.values;
    connection.query("DELETE FROM shoots WHERE ph_id=?", [req.body.ph_id])
        .then(([results, fields]) => {
            connection.query("INSERT INTO shoots(type_id,ph_id) VALUES ?",
[values])
                .then(([results, fields]) => {
                    res.send({"success": "yes"});
                })
```

```
                .catch(err => {
                    console.log(err);
                    res.send({"success": "no"});
                });
        })
        .catch(err => {
            console.log(err);
            res.send({"success": "no"});
        });
});


//filter and search functions
server.post('/filter', (req, res) => {
    res.cookies = req.cookies;
    let sel_user = "SELECT photographers.ph_id,
photographers.user_id,username,city,avatar_link,price,exp,organization,ROUND(AVG
(mark)) AS ave" +
        " FROM ( photographers INNER JOIN users ON
photographers.user_id=users.user_id) " +
        " LEFT JOIN ratings ON photographers.ph_id=ratings.ph_id ";
    sel_user += " WHERE price BETWEEN ? AND ? ";
    if (req.body.city != '0')
        sel_user += " AND city=? ";
    else
        sel_user += "AND city <> ?";
    if (req.body.years != '0')
        sel_user += " AND exp=? ";
    else
        sel_user += " AND exp<> ?";
    if (req.body.type != '0')
        sel_user += " AND photographers.ph_id IN (SELECT ph_id FROM shoots WHERE
type_id=?) ";
    connection.query(sel_user + " GROUP BY photographers.ph_id ORDER BY ave DESC
", [req.body.min_price, req.body.max_price, req.body.city, req.body.years,
req.body.type])
        .then(([results, fields]) => {
            let phs = [];
            for (let i in results)
                phs.push(results[i].ph_id);
```

```
            connection.query("SELECT ph_id,link FROM photos INNER JOIN folders
ON photos.folder_id=folders.folder_id WHERE ph_id IN (?)", [phs])
                .then(([results3, fields3]) => {
                    if (req.cookies.auth) {
                        connection.query("SELECT ph_id, user_id FROM favorites
WHERE user_id=? AND ph_id IN (?)", [req.cookies.user_id, phs])
                            .then(([results4, fields4]) => {
                                res.send({
                                    'success': "yes",
                                    ph_info: results,
                                    cookies: req.cookies,
                                    photos: results3,
                                    favorites: results4
                                });
                            })
                            .catch(err => {
                                res.send({'success': "no"});
                            });
                    }
                    else
                        res.send({
                            'success': "yes",
                            ph_info: results,
                            cookies: req.cookies,
                            photos: results3,
                            favorites: []
                        });
                })
                .catch(err => {
                    console.log(err);
                    res.send({'success': "no"});
                });
        })
        .catch(err => {
            console.log(err);
            res.send({'success': "no"});
        });
});
```

```
server.post('/searchph', (req, res) => {
    res.cookies = req.cookies;
    let name = req.body.name;
    connection.query(
        "SELECT photographers.ph_id,
photographers.user_id,username,city,avatar_link,price,exp,organization,ROUND(AVG
(mark)) AS ave" +
        " FROM ( photographers INNER JOIN users ON
photographers.user_id=users.user_id) " +
        " LEFT JOIN ratings ON photographers.ph_id=ratings.ph_id " +
        " WHERE username = ? OR firstname=? OR lastname=? GROUP BY
photographers.ph_id ORDER BY ave DESC ", [name, name, name])
        .then(([results, fields]) => {
            let phs = [];
            for (let i in results)
                phs.push(results[i].ph_id);
            connection.query("SELECT ph_id,link FROM photos INNER JOIN folders
ON photos.folder_id=folders.folder_id WHERE ph_id IN (?)", [phs])
                .then(([results3, fields3]) => {
                    if (req.cookies.auth) {
                        connection.query("SELECT ph_id, user_id FROM favorites
WHERE user_id=? AND ph_id IN (?)", [req.cookies.user_id, phs])
                            .then(([results4, fields4]) => {
                                res.send({
                                    'success': "yes",
                                    ph_info: results,
                                    cookies: req.cookies,
                                    photos: results3,
                                    favorites: results4
                                });
                            })
                            .catch(err => {
                                res.send({'success': "no"});
                            });
                    }
                    else
                        res.send({
                            'success': "yes",
                            ph_info: results,
                            cookies: req.cookies,
```

```
                            photos: results3,
                            favorites: []
                        });
                })
                .catch(err => {
                    res.send({'success': "no"});
                });
        })
        .catch(err => {
            res.send({'success': "no"});
        });
});
server.get('/search', function (req, res) {
    let request = custom.splitIntoTags(req.query.q);
    let photos = custom.search(request);
    if (photos[0] === undefined)
        res.render(__dirname + "/views/search.pug", {
            photos: photos,
            config: config,
            cookies: req.cookies,
            request: req.query.q
        });
    else {
        connection.query(" SELECT photo_id, username, link,title,tags, descr,
photos.folder_id " +
            "FROM ((photos INNER JOIN folders ON
folders.folder_id=photos.folder_id )" +
            " INNER JOIN photographers ON folders.ph_id=photographers.ph_id)
INNER JOIN users ON photographers.user_id=users.user_id " +
            " WHERE photo_id IN (?) ", [photos])
            .then(([results2, fields2]) => {
                for (p in results2)
                    results2[p].taglist =
custom.splitIntoTags(results2[p].tags);
                res.render(__dirname + "/views/search.pug", {
                    photos: results2,
                    config: config,
                    cookies: req.cookies,
                    request: req.query.q
```

```
                });
            })
            .catch(err => {
                console.log(err);
            });
    }
});


//return list of cities to filter
server.get('/getcities', function (req, res) {
    connection.query("SELECT city FROM users GROUP BY city", function (err,
results, fields) {
        res.cookies = req.cookies;
        if (err) {
            console.log(err);
            res.send(err);
        }
        res.send(results);
    });
});


//add and delete rows from table orders
server.post('/order', (req, res) => {
    res.cookies = req.cookies;
    connection.query("INSERT INTO
orders(ph_id,client_id,date,topic,contact_cl,message_cl,status)
VALUES(?,?,?,?,?,?,2) ",
        [req.body.ph_id, req.body.user_id, req.body.date, req.body.topic,
req.body.contact_cl, req.body.message_cl])
        .then(([results, fields]) => {
            connection.query("DELETE FROM freedates WHERE ph_id=? AND date=?",
[req.body.ph_id, req.body.date])
                .then(([results2, fields2]) => {
                    res.send({"success": "yes"});
                })
                .catch(err => {
                    console.log(err);
                    res.send({"success": "no", "error": err});
                });
        })
```

```
        .catch(err => {
            console.log(err);
            res.send({"success": "no", "error": err});
        });
});
server.post('/deleteorder', (req, res) => {
    res.cookies = req.cookies;
    connection.query("DELETE FROM orders WHERE order_id=?", [req.body.order_id])
        .then(([results, fields]) => {
            connection.query("INSERT INTO freedates(ph_id,date) VALUES(?,?)",
[req.body.ph_id, req.body.date])
                .then(([results, fields]) => {
                    res.send({"success": "yes"});
                })
                .catch(err => {
                    console.log(err);
                    res.send({"success": "no"});
                });
        })
        .catch(err => {
            console.log(err);
            res.send({"success": "no"});
        });
});


//change order status and update information
server.post('/cancel', (req, res) => {
    connection.query("UPDATE orders SET status=0,message_ph=? WHERE order_id=?",
[req.body.message_ph, req.body.order_id])
        .then(([results, fields]) => {
            res.cookies = req.cookies;
            res.send({"success": "yes"});
        })
        .catch(err => {
            console.log(err);
            res.send({"error": err});
        });
});
```

```
server.post('/approve', (req, res) => {

    connection.query("UPDATE orders SET status=1,message_ph=?, contact_ph=?
WHERE order_id=?", [req.body.message_ph, req.body.contact_ph,
req.body.order_id])

        .then(([results, fields]) => {

            res.cookies = req.cookies;

            res.send({"success": "yes"});

        })

        .catch(err => {

            console.log(err);

            res.send({"error": err});

        });

});


//photographer make cells in calendar free to order or vise versa

server.post('/removeactive', (req, res) => {

    res.cookies = req.cookies;

    connection.query("DELETE FROM freedates WHERE ph_id=? AND date=?",
[req.body.ph_id, req.body.date])

        .then(([results, fields]) => {

            res.send({"success": "yes"});

        })

        .catch(err => {

            console.log(err);

            res.send({"error": err});

        });

});

server.post('/setactive', (req, res) => {

    res.cookies = req.cookies;

    connection.query("INSERT INTO freedates(ph_id,date) VALUES(?,?)",
[req.body.ph_id, req.body.date])

        .then(([results, fields]) => {

            res.send({"success": "yes"});

        })

        .catch(err => {

            console.log(err);

            res.send({"error": err});

        });

});
```

```
//add ph profile to and delete it from favorites
server.post('/addfavorite', (req, res) => {
    res.cookies = req.cookies;
    connection.query("INSERT INTO favorites(user_id, ph_id) VALUES(?, ?)",
[req.body.user_id, req.body.ph_id])
        .then(([results, fields]) => {
            res.send({"success": "yes"});
        })
        .catch(err => {
            console.log(err);
            res.send({"success": "no", "error": err});
        });
});
server.post('/delfavorite', (req, res) => {
    res.cookies = req.cookies;
    connection.query("DELETE FROM favorites WHERE user_id=? AND ph_id=?",
[req.body.user_id, req.body.ph_id])
        .then(([results, fields]) => {
            res.send(results);
        })
        .catch(err => {
            console.log(err);
            res.send(err);
        });
});


//create, update and delete rows in table folders
server.post('/addfolder', (req, res) => {
    if (req.cookies.auth)
        connection.query("SELECT ph_id FROM photographers INNER JOIN users on
photographers.user_id= users.user_id WHERE username=?", [req.cookies.username])
            .then(([results, fields]) => {
                connection.query("INSERT INTO folders(ph_id,name) VALUES(?, ?)",
[results[0].ph_id, req.body.name])
                    .then(([results2, fields2]) => {
                        res.cookies = req.cookies;
                        res.send({'folder_id': results2.insertId});
                    })
```

```
                    .catch(err => {
                        console.log(err);
                        res.send(err);
                    });
            })
            .catch(err => {
                console.log(err);
                res.send(err);
            });
});
server.post('/editfolder', (req, res) => {


    connection.query("UPDATE folders SET name=? WHERE folder_id=?",
[req.body.name, req.body.folder_id])
        .then(([results, fields]) => {
            res.cookies = req.cookies;
            res.send(results);
        })
        .catch(err => {
            console.log(err);
            res.send(err);


        });
});
server.post('/deletefolder', (req, res) => {
    connection.query("SELECT photo_id as id FROM photos WHERE folder_id=?",
[req.body.folder_id])
        .then(([results, fields]) => {
            custom.deleteFromInvertedList(results);
            connection.query("DELETE FROM folders WHERE folder_id=?",
[req.body.folder_id])
                .then(([results1, fields1]) => {
                    res.cookies = req.cookies;
                    res.send(results1);
                })
                .catch(err => {
                    console.log(err);
                    res.send(err);
                });
```

```
        })
        .catch(err => {
            console.log(err);
            res.send(err);
        });
});


//create, update and delete rows in table photos
server.post('/addphoto', (req, res) => {
    if (req.cookies.auth)
        connection.query("INSERT INTO photos(folder_id,link,title,descr,tags)
VALUES(?, ?, ?, ?, ?)", [req.body.folder_id, req.body.link, req.body.title,
req.body.descr, req.body.tags])
            .then(([results, fields]) => {
                custom.updateInvertedList([{"id": results.insertId, "taglist":
custom.splitIntoTags(req.body.tags)}]);
                res.cookies = req.cookies;
                res.send({'success': 'yes', 'photo_id': results.insertId});
            })
            .catch(err => {
                console.log(err);
                res.send({'success': 'no', 'err': err});
            });
});
server.post('/deletephoto', (req, res) => {
    connection.query("DELETE FROM photos WHERE photo_id=?", [req.body.photo_id])
        .then(([results, fields]) => {
            custom.deleteFromInvertedList([{"id": req.body.photo_id}]);
            res.cookies = req.cookies;
            res.send(results);
        })
        .catch(err => {
            console.log(err);
            res.send(err);
        });
});
server.post('/editphoto', (req, res) => {
    connection.query("UPDATE photos SET title=?, descr=?, tags=? WHERE
photo_id=?", [req.body.title, req.body.descr, req.body.tags, req.body.photo_id])
```

```
        .then(([results2, fields2]) => {
            res.cookies = req.cookies;
            let taglist = custom.splitIntoTags(req.body.tags);
            custom.deleteFromInvertedList([{"id":
req.body.photo_id}]).updateInvertedList([{
                "id": req.body.photo_id,
                "taglist": taglist
            }]);
            res.send(taglist);
        })
        .catch(err => {
            console.log(err);
            res.send(err);
        });
});


//set and delete mark for ph profile
server.post('/unvote', (req, res) => {
    res.cookies = req.cookies;
    connection.query("DELETE FROM ratings WHERE user_id=? AND ph_id=?",
[req.body.user_id, req.body.ph_id])
        .then(([results, fields]) => {
            connection.query("SELECT ROUND(AVG(mark)) as ave FROM ratings WHERE
ph_id=?", [req.body.ph_id])
                .then(([results2, fields2]) => {
                    if (results2[0].avg == null)
                        results2[0].avg == 0;
                    results2[0].cookies = req.cookies;
                    res.cookies = req.cookies;
                    res.send(results2[0]);
                })
                .catch(err => {
                    console.log(err);
                    res.send(err);
                });
        })
        .catch(err => {
            console.log(err);
            res.send(err);
```

```
        });
});
server.post('/vote', (req, res) => {
    res.cookies = req.cookies;
    connection.query("INSERT INTO ratings(user_id, ph_id,mark) VALUES(?, ?, ?)",
[req.body.user_id, req.body.ph_id, req.body.mark])
        .then(([results, fields]) => {
            connection.query("SELECT ROUND(AVG(mark)) as ave FROM ratings WHERE
ph_id=?", [req.body.ph_id])
                .then(([results2, fields2]) => {
                    if (results2[0].avg == null || results2[0] == undefined)
                        results2[0].avg == 0;
                    results2[0].cookies = req.cookies;
                    res.cookies = req.cookies;
                    res.send(results2[0]);
                })
                .catch(err => {
                    console.log(err);
                    res.send(err);
                });
        })
        .catch(err => {
            console.log(err);
            res.send(err);
        });
});

//get raw password and return hashed
server.post('/hashpass', (req, res) => {
    res.cookies = req.cookies;
    res.send({"pass": hashing(req.body.pass)});
});

//function used to hash password
function hashing(raw) {
    let hash = crypto.createHmac("sha256", "password")
        .update(config.salt + raw).digest("hex");
    for (let i = 0; i < 5; i++) {
```

```
        hash = crypto.createHmac("sha256", "password")
            .update(config.salt + hash).digest("hex");
    }
    return hash;
}
```

custom_modules.js

```
let invertedList = new Map();
module.exports = {
    getInvertedList: function () {
        return invertedList;
    },
    setInvertedList: function (photos) {
        //create array of all tags
        let temp = [];
        for (photo of photos)
            for (tag of photo.taglist)
                temp.push(tag);
        //sorted and with unique values
        let tags = insertionSort([...new Set(temp)]);
        invertedList = new Map();
        //put them as keys in map with value = [all ids of photos which have
this tag]
        for (let tag of tags) {
            let values = [];
            for (photo of photos)
                for (t of photo.taglist)
                    if (t == tag)
                        values.push(photo.id);
            invertedList.set(tag, values);
        }
    },
    updateInvertedList: function (photo) {
        //if photo is updated, add new tags or photo id to existed tags
        for (tag of photo.taglist)
            if (invertedList.get(tag) != null)
                if (!(invertedList.get(tag)).includes(photo.id))
```

```
                    invertedList.set(tag, invertedList.get(tag).push(photo.id));
                else
                    invertedList.set(tag, photo.id);
        return invertedList;
    },
    deleteFromInvertedList: function (deleted) {
        //if photo is deleted its id is deleted from inverted list too
        for (photo of deleted)
            for (tag of invertedList)
                if (tag[1].includes(photo.id))
                    tag[1] = tag[1].splice(photo.id, 1);
        return invertedList;
    },
    search: function (request) {
        return search(request);
    },
    splitIntoTags: function (str) {
        return splitIntoTags(str);
    }
};


//search photo by tags
function search(request) {
    let temp = [];
    //find all photos which have these tags
    for (let word of request) {
        let photos = invertedList.get(word);
        if (photos != null && photos != undefined)
            for (let i of photos)
                temp.push(i);
    }


    temp = insertionSort(temp);
    let prev = [], values = [], frequency = [];
    for (let i of  temp) {
        if (i !== prev) {
```

```
            values.push([i]);
            frequency.push(1);
        } else
            frequency[frequency.length - 1]++;
        prev = i;
    }
    //sort them by frequency
    values = insertionSortTwoArrays(frequency, values);
    return values;
}

//simple insertion sort function
const insertionSort = (array) => {
    for (let i = 1; i < array.length; i++) {
        let j = i - 1;
        let temp = array[i];
        while (j >= 0 && (array[j] > temp)) {
            array[j + 1] = array[j];
            j--;
        }
        array[j + 1] = temp;
    }
    return array;
};

//insertion sort to sort values by their frequency
const insertionSortTwoArrays = (array, values) => {
    for (let i = 1; i < array.length; i++) {
        let j = i - 1;
        let temp = array[i];
        let temp2 = values[i];
        while (j >= 0 && (array[j] < temp)) {
            array[j + 1] = array[j];
            values[j + 1] = values[j];
            j--;
        }
        array[j + 1] = temp;
        values[j + 1] = temp2;
```

```
    }
    return values;
};

//split a string into tags array
function splitIntoTags(str) {
    if (str == null || str == "" || str == undefined)
        return [];
    let tagList = [];
    let tags = str.split(/\s,'".-?!/);
    for (let tag of tags)
        tagList.push(tag);
    return tagList;
}
```

Додаток З

(обов’язковий)

Програмний код(клієнтська частина)

navbar.pug

```
include PUG-Bootstrap/_bootstrap

html

    head

        title= config.title

        include views/links

    body

        nav(id='menu' class="navbar navbar-expand-md rounded-0 menu mg-0 font-17
navbar-fixed" style="background: #1e1e1e")

            button(class="navbar-toggler btn btn-outline-light mr-auto"
type="button" data-toggle="collapse" data-target="#mynavbar")

                span(class="navbar-toggler-icon")

            img(class="logo" src="../public/images/icon.png")

            a(class="navbar-brand disabled brand" style="color: #efefef!
important;") photoLAB

            div(class="collapse navbar-collapse ", id="mynavbar")

                div(class="navbar-nav expand my-navbar font-17")

                    if(cookies.role!='admin')

                        if(active=='home')

                            a(class="nav-item nav-link menu-active" active
style="color: #efefef" href="/home/"+cookies.username) головна

                        else

                            a(class="nav-item nav-link" active style="color:
#efefef" href="/home/"+cookies.username) головна


                        if(active=='profile')

                            a(class="nav-item nav-link menu-active"
style="color: #efefef" href="/profile/"+cookies.username) профіль

                        else

                            a(class="nav-item nav-link" style="color: #efefef"
href="/profile/"+cookies.username) профіль

                        if(cookies.role=='photographer')

                            if(active == 'album')

                                a(class="nav-item nav-link menu-active"
style="color: #efefef" href="/photoalbum/"+cookies.username) фотоальбом
```

```
                            else
                                a(class="nav-item nav-link" style="color:
#efefef" href="/photoalbum/"+cookies.username) фотоальбом
                        if(active=='about')
                            a(class="nav-item nav-link menu-active"
style="color: #efefef" href='/about') про нас
                        else
                            a(class="nav-item nav-link" style="color: #efefef"
href='/about') про нас
                p(class='nav-text float-rightt mg-10 text-25 white')=
'@'+cookies.username
                button(class="btn btn-outline-warning m-l-15 float-rightt",
type='button', data-toggle="modal", data-target='#exit_modal') Вихід


include views/modals
include views/scripts
```

photoalbum.pug

```
if(cookies.auth=='false')
    include ../navbar_unsign
else
    include ../navbar
.row(class="pd-0 mg-0")
    .col-md-5
        if(guest)
            a(class='link1 btn btn-outline-light sharp-corners' href=`/guest/$
{ph_username}`)
                i(class='fa fa-arrow-left')
                span= `до профілю @${ph_username}`


      div(class="nav sticky-top flex-column nav-pills" id="folders_container"
      role="tablist" aria-orientation="vertical")
            if(!guest)
                  button(class="btn btn-block col-md-10 btn-outline-light sharp-
                  corners" type='button', data-toggle="modal", data-
                  target='#add_folder_modal') +додати папку
            div
                  button(id='edit_folder', class="link2 bg-transparent border-
                  none", data-toggle="modal", data-target='#edit_folder_modal')
                        span Перейменувати папку
                        i(class="fa fa-edit ")
```

```
            button(id='delete_folder', class="link-hover-red  bg-
            transparent border-none",  data-toggle="modal", data-
            target='#delete_folder_modal')

                  span Видалити папку

                  i(class="fa fa-trash-o ")

      div(class="empty_div")

      each f in folders

            if(f == folders[0])

                  a(class="link1 col-md-10 folder-link active" id=`folder$
                  {f.folder_id}-tab` data-toggle="pill"  href=`#folder$
                  {f.folder_id}`role="tab" aria-controls= href = `folder$
                  {f.folder_id}` aria-selected="true"
                  onclick=`setCurrentFolder($
                  {JSON.stringify(f.folder_id)})`)= f.name

            script.

                  setCurrentFolder(!{JSON.stringify(f.folder_id)});

            else

                  a(class="link1 col-md-10  folder-link" id=`folder$
                  {f.folder_id}-tab` data-toggle="pill"  href=`#folder$
                  {f.folder_id}`role="tab" aria-controls= href = `folder$
                  {f.folder_id}` aria-selected="false"
                  onclick=`setCurrentFolder($
                  {JSON.stringify(f.folder_id)})`)= f.name

.col-md-1

.col-md-5

      div(class="tab-content" id='photos_container')

      if(!guest)

            button(id='add_photo_button' class="btn btn-block btn-outline-
            light sharp-corners" type='button', data-toggle="modal", data-
            target='#add_photo_modal') +додати фото

      br

      script.

            let folders0 = !{JSON.stringify(folders)};

            if (folders0[0] == undefined)

                  $('#add_photo_button').hide();

            if (getCurrentFolder() == "")

                  setCurrentFolder(folders0[0].folder_id);

      each f in folders

            if(f.folder_id === folders[0].folder_id)

                  div(class="tab-pane show active" id=`folder$
                  {f.folder_id}` role="tabpanel" aria-labelledby=`folder$
                  {f.folder_id}-tab`)

                  each photo in photos

                        if(photo.folder_id === f.folder_id)
```

```
                                    div(class='card ' id=`photo$
                                    {photo.photo_id}`)
                                          div(class='img-container')
                                                div(class="circle-avatar "
                                                style=`background-image:url($
                                                {photo.link})`)
                                          div(class='card-body')
                                                if(!guest)
                                                            button(class='link-
hover-red2 bg-transparent float-right text-20 border-none' data-toggle="modal",
data-target='#delete_photo_modal'
onclick=`$("#delete_photo_button").attr("onclick","deletePhoto($
{photo.photo_id})")`)
                                                            i(class="fa text-20
                                                            fa-trash-o ")
                                                            button(class='link1
bg-transparent float-right text-20 border-none' id=`editphotobutton$
{photo.photo_id}` data-toggle="modal", data-target='#edit_photo_modal'
onclick=`fillEditPhotoModal(${JSON.stringify(photo)})`)
                                                            i(class="fa text-20
                                                            fa-edit ")
                                                      h5(class='card-title'
id=`photo_title${photo.photo_id}`)= photo.title
                                                      p(class='text-14 text-
break' id=`photo_descr${photo.photo_id}`)= photo.descr
                                                      p(class="text-14 text-muted
text-break" id=`photo_tags${photo.photo_id}`)
                                                            each tag in
photo.taglist
                                                                  a(class='link1'
href=`/search?q=${tag}`)= '#' + tag + ' '
                                                                  br
                  else
                      div(class="tab-pane fade" id=`folder${f.folder_id}`
role="tabpanel" aria-labelledby=`folder${f.folder_id}-tab`)
                          each photo in photos
                              if(photo.folder_id === f.folder_id)
                                  div(class='card' id=`photo${photo.photo_id}`)
                                      div(class='img-container')
                                          div(class="circle-avatar "
style=`background-image:url(${photo.link})`)
                                      div(class='card-body')
                                          if((!guest)&&cookies.auth==true)
                                              button(class='link-hover-red2 bg-
transparent float-right text-20 border-none' data-toggle="modal", data-
target='#delete_photo_modal'
```

```
                                                onclick=`$
("#delete_photo_button").attr("onclick","deletePhoto(${photo.photo_id})")`)
                                                    i(class="fa text-20 fa-trash-o
")
                                                button(class='link1 bg-transparent
float-right text-20 border-none' data-toggle="modal", data-
target='#edit_photo_modal'
                                                onclick=`fillEditPhotoModal($
{JSON.stringify(photo)})` id=`editphotobutton${photo.photo_id}`)
                                                    i(class="fa text-20 fa-edit ")
                                            h5(class='card-title' id=`photo_title$
{photo.photo_id}`)= photo.title
                                            p(class='text-14 text-break'
id=`photo_descr${photo.photo_id}`)= photo.descr
                                            p(class="text-14 text-muted text-break"
id=`photo_tags${photo.photo_id}`)
                                                each tag in photo.taglist
                                                    a(class='link1' href=`/search?
q=${tag}`)= '#' + tag + ' '
                                    br


    .col-md-1
```

requests.js

```
//all files here are ajax-requests to server to manipulate with data
function login() {
    $("#login_failed").remove();
    let data = {
        "login": $("#login_login").val(),
        "password": $("#login_pass").val()
    };
    $.ajax({
        url: 'http://localhost:2606/login',
        type: 'post',
        dataType: 'json',
        data: JSON.stringify(data),
        contentType: 'application/json',
        success: function (data2) {
            if (data2.success == 'yes')
                window.location.replace(data2.href);
            else
```

```
                $("#entry_form").append("<div class='panel' id='login_failed'>"
+
                    "<p class='red'>Неправильний логін або пароль</p></div>");
        },
        error: function (data2) {
            console.log(data2.error);
            $("#entry_form").append("<div class='panel' id='login_failed'>" +
                "<p class='red'>Неправильний логін або пароль</p></div>");
        }
    });
}
function regClient(users) {
    $("#invalid_reg_un").text("Це поле не може бути пустим");
    $("#invalid_reg_email").text("Некоректно введена електронна пошта");
    $("#reg_failed").remove();
    let a = validEmail("reg_email");
    let b = validEmpty("reg_username");
    let c = validNamesF("reg_city");
    if (!(a && b && c))
        return false;
    if (!validExistedUsername($("#reg_username").val(), users)) {
        $("#invalid_reg_un").text("Користувач з таким іменем уже існує!");
        $("#reg_username").removeClass("is-valid").addClass("is-invalid");
        return false;
    }
    if (!validExistedEmail($("#reg_email").val(), users)) {
        $("#invalid_reg_email").text("Користувач з такою поштою уже існує!");
        $("#reg_email").removeClass("is-valid").addClass("is-invalid");
        return false;
    }
    if (!validPassword("reg_password"))
        return false;
    let pass = $("#reg_password").val();
    let pass2 = $("#reg_password2").val();
    validPassword("reg_password2");
    if (pass != pass2)
        return false;
    let data = {
```

```
        "password": pass,
        "username": $("#reg_username").val(),
        "email": $("#reg_email").val(),
        "city": $("#reg_city").val()
    };

    $.ajax({
        url: 'http://localhost:2606/register',
        type: 'post',
        dataType: 'json',
        data: JSON.stringify(data),
        contentType: 'application/json',
        success: function (data2) {
            removeValid('registry_form');
            clearForm('registry_form');
            if (data2.success == "yes")
                window.location.replace(data2.href);
            else
                $("#registry_form").append("<div class='panel'
id='login_failed'>" +
                    "<p class='red'>Вибачте,виникли проблеми при
реєстрації</p></div>");
        },
        error: function (data2) {
            console.log(data2.error);
            $("#registry_form").append("<div class='panel' id='reg_failed'>" +
                "<p class='red'>Вибачте,виникли проблеми при
реєстрації</p></div>");
        }
    });
}
function regPh(users, types) {
    $("#invalid_regph_un").text("Це поле не може бути пустим");
    $("#invalid_regph_email").text("Некоректно введена електронна пошта");
    $("#reg_failed").remove();
    let a = validEmail("ph_reg_email");
    let b = validEmpty("ph_reg_username");
    let c = validNames("ph_reg_lastname");
    let d = validNames("ph_reg_firstname");
```

```
let e = validNamesF("ph_reg_fathername");
let f = validNamesF("ph_reg_city");
if (!(a && b && c && d && e && f))
    return false;
const emailSel = $("#ph_reg_email");
const userSel = $("#ph_reg_username");
if (!validExistedUsername(userSel.val(), users)) {
    $("#invalid_regph_un").text("Користувач з таким іменем уже існує!");
    userSel.removeClass("is-valid").addClass("is-invalid");
    return false;
}
if (!validExistedEmail(emailSel.val(), users)) {
    $("#invalid_regph_email").text("Користувач з такою поштою уже існує!");
    emailSel.removeClass("is-valid").addClass("is-invalid");
    return false;
}
if (!validPassword("ph_reg_password"))
    return false;
let pass = $("#ph_reg_password").val();
let pass2 = $("#ph_reg_password2").val();
validPassword("ph_reg_password2");
if (pass != pass2)
    return false;
let values = [];
for (t of types)
    if ($('#set_type' + t.type_id).is(":checked"))
        values.push([t.type_id]);
let data = {
    "password": pass,
    "username": userSel.val(),
    "email": emailSel.val(),
    "city": $("#ph_reg_city").val(),
    "lastname": $("#ph_reg_lastname").val(),
    "firstname": $("#ph_reg_firstname").val(),
    "fathername": $("#ph_reg_fathername").val(),
    "price": $("#ph_reg_price").val(),
    "experience": $("#ph_reg_experience").val(),
    "organization": $("#ph_reg_org").val(),
```

```
        "types": values
    };
    $.ajax({
        url: 'http://localhost:2606/registerph',
        type: 'post',
        dataType: 'json',
        data: JSON.stringify(data),
        contentType: 'application/json',
        success: function (data2) {
            removeValid('registry_ph_form');
            if (data2.success == "yes")
                window.location.replace(data2.href);
            else
                $("#registry_ph_form").append("<div class='panel'
id='login_failed'>" +
                    "<p class='red'>Вибачте,виникли проблеми при
реєстрації</p></div>");
        },
        error: function (data2) {
            console.log(data2.error);
            $("#registry_ph_form").append("<div class='panel' id='reg_failed'>"
+
                "<p class='red'>Вибачте,виникли проблеми при
реєстрації</p></div>");
        }
    });
}


function addToFavorites(ph_id, user_id) {
    let data = {
        "ph_id": ph_id,
        "user_id": user_id
    };
    $.ajax({
        url: 'http://localhost:2606/addfavorite',
        type: 'post',
        dataType: 'json',
        data: JSON.stringify(data),
        contentType: 'application/json',
```

```
        success: function (data2) {
            if (data2.success == 'yes')
                $("#fav" + ph_id).empty().append(`<i class=' text-40 material-
icons '>&#xe8e6;</i>`).attr("onclick", "deleteFromFavorites(" + ph_id + "," +
user_id + ")");
        },
        error: function (data2) {
            console.log(data2.error);
        }
    });
}
function deleteFromFavorites(ph_id, user_id, profile = false) {
    let data = {
        "ph_id": ph_id,
        "user_id": user_id
    };
    $.ajax({
        url: 'http://localhost:2606/delfavorite',
        type: 'post',
        dataType: 'json',
        data: JSON.stringify(data),
        contentType: 'application/json',
        success: function (data2) {
            if (profile) {
                $("#favpanel" + ph_id).remove();
            } else {
                $("#fav" + ph_id).empty().append(`<i class=' text-40 material-
icons '>&#xe8e7;</i>`).attr("onclick", "addToFavorites(" + ph_id + "," + user_id
+ ")");
            }
        },
        error: function (data2) {
            console.log(data2.error);
        }
    });
}
//set & get current folder to local storage
function setCurrentFolder(folder_id) {
    localStorage.setItem("folder", folder_id);
```

```
}
function getCurrentFolder() {
    return localStorage.getItem("folder");
}
function addFolder() {
    if (!validEmpty("addfoldertitle"))
        return;
    let name = $("#addfoldertitle").val();
    let data = {
        'name': name
    };
    $.ajax({
        url: 'http://localhost:2606/addfolder',
        type: 'post',
        dataType: 'json',
        data: JSON.stringify(data),
        contentType: 'application/json',
        success: function (data2) {
            removeValid("add_folder_form");
            clearForm("add_folder_form");
            $("#folders_container").append("<a class='link1 text-center col-md-
10 folder-link' id='folder" + data2.folder_id + "-tab' " +
                "data-toggle='pill'  href='#folder" + data2.folder_id + "'
role='tab' aria-controls='#folder" + data2.folder_id + "' " +
                "aria-selected='false' onclick='setCurrentFolder(" +
data2.folder_id + ")'>" + name + "</a>");
            $("#photos_container").append("<div class='tab-pane fade'
id='folder" + data2.folder_id + "' role='tabpanel' " +
                "aria-labelledby='folder" + data2.folder_id + "-tab'></div>" +
                "<br>");
            $("#edit_folder_select").append("<option value=" + data2.folder_id +
" id='editfolderoption" + data2.folder_id + "'>" + name + "</option>");
            $("#delete_folder_select").append("<option value=" + data2.folder_id
+ " id='delfolderoption" + data2.folder_id + "'>" + name + "</option>");
            $('#add_photo_button').show();
            $("#add_folder_modal .close").click();
        },
        error: function (data2) {
            console.log(data2.error);
        }
```

```
    });
}
function deleteFolder() {
    let folder_id = $("#delete_folder_select").val();
    let data = {
        'folder_id': folder_id
    };
    $.ajax({
        url: 'http://localhost:2606/deletefolder',
        type: 'post',
        dataType: 'json',
        data: JSON.stringify(data),
        contentType: 'application/json',
        success: function (data2) {
            $("#folder" + folder_id + "-tab").remove();
            $("#folder" + folder_id).remove();
            $("#delfolderoption" + folder_id).remove();
            $("#editfolderoption" + folder_id).remove();
            $("#delete_folder_modal .close").click();
        },
        error: function (data2) {
            console.log(data2.error);
        }
    });
}
function editFolder() {
    let folder_id = $("#edit_folder_select").val();
    if (!validEmpty("edit_folder_caption"))
        return;
    let name = $("#edit_folder_caption").val();
    let data = {
        'folder_id': folder_id,
        'name': name
    };
    $.ajax({
        url: 'http://localhost:2606/editfolder',
        type: 'post',
        dataType: 'json',
```

```
        data: JSON.stringify(data),
        contentType: 'application/json',
        success: function (data2) {
            removeValid("edit_folder_form");
            clearForm("edit_folder_form");
            $("#folder" + folder_id + "-tab").text(name);
            $("#editfolderoption" + folder_id).text(name);
            $("#delfolderoption" + folder_id).text(name);
            $("#edit_folder_modal .close").click();
        },
        error: function (data2) {
            console.log(data2.error);
        }
    });
}
function addPhotoLink() {
    let folder_id = getCurrentFolder();
    if (!(validEmpty("add_photolink_link") &&validTags("add_photolink_tags")))
        return false;
    let data = {
        "folder_id": folder_id,
        "title": $("#add_photolink_title").val(),
        "descr": $("#add_photolink_descr").val(),
        "tags": $("#add_photolink_tags").val().toLowerCase(),
        "link": $("#add_photolink_link").val()
    };
    $.ajax({
        url: 'http://localhost:2606/addphoto',
        type: 'post',
        dataType: 'json',
        data: JSON.stringify(data),
        contentType: 'application/json',
        success: function (data2) {
            if (data2.success == 'yes') {
                removeValid("add_photolink_form");
                clearForm("add_photolink_form");
                data.photo_id = data2.photo_id;
                let taglist = splitIntoTags(data.tags);
```

```
                data.taglist = taglist;

                $("#folder" + folder_id).prepend("<div class='card' id='photo" +
data2.photo_id + "'>" +

                    "<div class='img-container'><div class='circle-avatar'
style='background-image:url(" + data.link + ")'></div></div>" +

                    "<div class='card-body'>" +

                    "<button class='link-hover-red2 bg-transparent float-right
text-20 border-none' data-toggle='modal' " +

                    "data-target='#delete_photo_modal'
onclick='fillDeletePhotoOnclick(" + data2.photo_id + ")'>" +

                    "<i class='fa text-20 fa-trash-o '></i></button>" +

                    "<button class='link1 bg-transparent float-right text-20
border-none' data-toggle='modal'  data-target='#edit_photo_modal'" +

                    " onclick='fillEditPhotoModal(" + JSON.stringify(data) + ")'
id='editphotobutton" + data2.photo_id + "'>" +

                    "<i class='fa text-20 fa-edit'></i></button>" +

                    "<h5 class='card-title' id='photo_title" + data2.photo_id +
"'>" + data.title + "</h5>" +

                    "<p class='text-14 text-break' id='photo_descr" +
data2.photo_id + "'>" + data.descr + "</p>" +

                    " <p class='text-14 text-muted text-break' id='photo_tags" +
data2.photo_id + "'></p>" +

                    "</div></div><br>");

                $("#add_photo_modal .close").click();

                    let pattern=[/\s/];

                for (tag of taglist)

                    if(!tag.match(pattern))

                    $("#photo_tags" + data2.photo_id).append("<a class='link1'
href='/search?q=" + tag + "'>#" + tag + " "+"</a>");

            } else {

                $("#add_photo_modal").append("<div id='photo_fail' class='panel
panel-danger'><div class='panel-heading'>" +

                    "Виникла помилка при додавані фото, спробуйте ще раз</div></
div>");

                setTimeout(() => {

                    $("#photo_fail").remove();

                    $("#add_photo_modal .close").click();

                }, 15000);

            }

        },

        error: function (data2) {

            console.log(data2.error);
```

```
            $("#add_photo_modal").append("<div id='photo_fail' class='panel
panel-danger'><div class='panel-heading'>" +
                "Виникла помилка при додавані фото, спробуйте ще
раз</div></div>");
            setTimeout(() => {
                $("#photo_fail").remove();
                $("#add_photo_modal .close").click();
            }, 15000);
        }
    });
}
function deletePhoto(photo_id) {



    let data = {
        'photo_id': photo_id
    };
    $.ajax({
        url: 'http://localhost:2606/deletephoto',
        type: 'post',
        dataType: 'json',
        data: JSON.stringify(data),
        contentType: 'application/json',
        success: function (data2) {
            $("#photo" + photo_id).empty().remove();
            $("#delete_photo_modal .close").click();
        },
        error: function (data2) {
            console.log(data2.error);
        }
    });
}
function editPhoto(photo_id) {
    if (!validTags("edit_photo_tags"))
        return;
    let data = {
        "photo_id": photo_id,
        "title": $("#edit_photo_title").val(),
```

```
        "descr": $("#edit_photo_descr").val(),
        "tags": $("#edit_photo_tags").val().toLowerCase()
    };



    $.ajax({
        url: 'http://localhost:2606/editphoto',
        type: 'post',
        dataType: 'json',
        data: JSON.stringify(data),
        contentType: 'application/json',
        success: function (data2) {
            removeValid("edit_photo_form");
            $("#photo_title" + photo_id).text(data.title);
            $("#photo_descr" + photo_id).text(data.descr);
            $("#photo_tags" + photo_id).empty();
            for (tag of data2)
                $("#photo_tags" + photo_id).append("<a class='link1'
href='search?q=" + tag + "'> #" + tag + "</a>");
            data.taglist = data2;
            $("#editphotobutton" + photo_id).attr('onclick',
'fillEditPhotoModal(' + data + ')');
            $("#edit_photo_modal .close").click();
        },
        error: function (data2) {
            console.log(data2.error);
        }
    });
}
function fillEditPhotoModal(photo) {



    $("#edit_photo_button").attr("onclick", "editPhoto(" + photo.photo_id +
")");
    if (photo.descr != null)
        $("#edit_photo_descr").val(photo.descr);
    if (photo.title != null)
        $("#edit_photo_title").val(photo.title);
    if (photo.tags != null)
```

```
        $("#edit_photo_tags").val(photo.tags);
}
function fillDeletePhotoOnclick(photo_id) {
    $('#delete_photo_button').attr('onclick', 'deletePhoto(' + photo_id + ')')
}
function splitIntoTags(str) {
    let tags = [];
    let stringArray = str.split(/\s/);
    for (let s of stringArray)
        tags.push(s);
    return tags;
}
function fillCities() {
    $.ajax({
        url: 'http://localhost:2606/getcities',
        method: 'get',
        success: function (data) {
            for (let i of data)
                if (i.city != null && i.city != "")
                    $("#filter_city").append("<option value='" + i.city + "'>" +
i.city + "</option>");
        }
    });

}

function vote(mark, user_id, ph_id) {
    for (let j = 0; j < mark; j++) {
        $("#rating_star" + j).removeClass("fa-star-o").addClass("fa-
star").hover(function () {
            for (let j = 0; j < mark; j++) {
                $("#rating_star" + j).removeClass("fa-star-o").addClass("fa-
star");
            }
        }, function () {
            for (let j = 0; j < mark; j++) {
                $("#rating_star" + j).removeClass("fa-star-o").addClass("fa-
star");
            }
```

```
        });
    }
    console.log(user_id + " vote " + ph_id);
    let data = {
        "user_id": user_id,
        "ph_id": ph_id,
        "mark": mark
    };
    $.ajax({
        url: 'http://localhost:2606/vote',
        type: 'post',
        dataType: 'json',
        data: JSON.stringify(data),
        contentType: 'application/json',
        success: function (data2) {
            let info = {
                "ave": data2.ave,
                "mark": mark,
                "ph_id": ph_id
            };
            setRatings(true, info, data2.cookies);
        },
        error: function (data2) {
            console.log(data2.error);
        }
    });
}
function unvote(user_id, ph_id) {
    console.log(user_id + " unvote " + ph_id);
    let data = {
        "user_id": user_id,
        "ph_id": ph_id
    };
    $.ajax({
        url: 'http://localhost:2606/unvote',
        type: 'post',
        dataType: 'json',
        data: JSON.stringify(data),
```

```
        contentType: 'application/json',
        success: function (data2) {
            let info = {
                "ave": data2.ave,
                "mark": 0,
                "ph_id": ph_id
            };
            setRatings(true, info, data2.cookies);
        },
        error: function (data2) {
            console.log(data2.error);
        }
    });
}
function makeOrder(ph_id, user_id, month, day, year) {
    let date_sql = year + '-' + month + '-' + day;
    let a = validEmpty("make_order_topic");
    let b = validEmpty("make_order_contact");
    if(!(a&&b)) return false;
    let topic=$("#make_order_topic").val();
    let data = {
        "ph_id": ph_id,
        "user_id": user_id,
        "date": date_sql,
        "topic": topic,
        "contact_cl": $("#make_order_contact").val(),
        "message_cl": $("#make_order_text").val()
    };
    $.ajax({
        url: 'http://localhost:2606/order',
        type: 'post',
        dataType: 'json',
        data: JSON.stringify(data),
        contentType: 'application/json',
        success: function (data2) {
            if(data2.success=="yes"){
```

```
                $("#day"+day+"-"+month).prop("onclick",
null).off("click").removeClass("free").addClass("busy").removeAttr('data-
target');

                $("#make_order_form").prepend("<div class='panel panel-success'
id='order_success'>" +

                    "<div class='panel-heading'><icon class='fa
fa-check'></icon>Замовлення відправлено фотографу! Очікуйте відповіді
</div></div>");

                setTimeout(() => {

                    $("#order_success").remove();

                    clearForm("make_order_form");

                    removeValid("make_order_form");

                    $("#make_order_modal .close").click();

                }, 15000);

            }else{

                $("#make_order_form").prepend("<div class='panel panel-danger'
id='order_failed'>" +

                    "<div class='panel-heading'>Вибачте, виникла помилка
</div></div>");

                setTimeout(() => {

                    $("#order_failed").remove();

                    $("#make_order_modal .close").click();

                }, 15000);

            }

        },

        error: function (data2) {

            $("#make_order_form").prepend("<div class='panel panel-danger'
id='order_failed'>" +

                "<div class='panel-heading'>Вибачте, виникла
помилка</div></div>");

            setTimeout(() => {

                $("#order_failed").remove();

                $("#make_order_form .close").click();

            }, 15000);

        }

    });

}

function cancelOrder(order_id,date,ph_id){

    let date_sql=date.split('T');

    let data= {

        "order_id": order_id,
```

```
        "date": date_sql[0],
        "ph_id":ph_id
    };
    $.ajax({
        url: 'http://localhost:2606/deleteorder',
        type: 'post',
        dataType: 'json',
        data: JSON.stringify(data),
        contentType: 'application/json',
        success: function (data2) {
            if(data2.success=="yes"){
                $("#order"+order_id).prepend("<div class='panel panel-success'
id='order_success'>" +
                    "<div class='panel-heading'><icon class='fa
fa-check'></icon>Замовлення успішно скасовано </div></div>");
                setTimeout(() => {
                    $("#order"+order_id).remove();
                }, 15000);
            }else{
                $("#order"+order_id).prepend("<div class='panel panel-danger'
id='order_failed'>" +
                    "<div class='panel-heading'>Не вдалося скасувати замовлення,
спробуйте ще раз пізніше</div></div>");
                setTimeout(() => {
                    $("#order_failed").remove();
                    $("#make_order_modal .close").click();
                }, 15000);
            }
        },
        error: function (data2) {
            $("#order"+order_id).prepend("<div class='panel panel-danger'
id='order_failed'>" +
                "<div class='panel-heading'>Не вдалося скасувати замовлення,
спробуйте ще раз пізніше</div></div>");
            setTimeout(() => {
                $("#order_failed").remove();
                $("#make_order_form .close").click();
            }, 15000);
        }
    });
```

```
}



function approve(order){
    if(!validEmpty("approve_order_contact")) return false;
    let order_id=order.order_id;
    let data= {
        "order_id": order_id,
        "message_ph":$("#approve_order_text").val(),
        "contact_ph":$("#approve_order_contact").val()
    };
        let d = order.date.split(/[-T:]/);
    let date=d[2] + ' ' + months[parseInt(d[1], 10)].genitive + " " + d[0] + "
";
    $.ajax({
        url: 'http://localhost:2606/approve',
        type: 'post',
        dataType: 'json',
        data: JSON.stringify(data),
        contentType: 'application/json',
        success: function (data2) {
            if(data2.success=="yes"){
                order.message_ph=$("#approve_order_text").val();
                order.contact_ph=$("#approve_order_contact").val();
                $("#order"+order_id).remove();
                $("#approved_orders").append("<div class='panel panel-success
mg-b-20'  id='order"+order_id+"'>" +
                    "<div class='panel-heading pointer' data-toggle='collapse'
href='#ordercard"+order_id+"' aria-expanded='true'>" +
                    "<span class='pd-r-25'
id='date_order"+order_id+"'>"+date+"</span>" +
                    "<a class='link1'
href='/guest/"+order.username+"'>@"+order.username+"</a>" +
                    "<i class='fa fa-check mg-5 float-right'></i>" +
                    "</div>" +
                    "<div class='panel-body collapse show in'
id='ordercard"+order_id+"' aria-expanded='true' >" +
                    "<p>Тема: "+order.topic+"</p>" +
                    "<div class='row'>" +
                    "<div class='col-md-8'>" +
```

```
                    "<p class='text-break'>"+order.message_cl+"</p><p>Контакт клієнта:"+order.contact_cl+"</p></div>" +
                    "<div class='col-md-4'></div><div class='col-md-4'></div>" +
                    "<div class='col-md-8'><p class='text-break'>"+order.message_ph+"</p><p>Ваш контакт:"+order.contact_ph+"</p></div>" +
                    "<div class='col-md-12'><button class='border-none btn-block bg-transparent red pointer' data-toggle='modal'" +
                    " onclick='setCancelOrderOnclick("+JSON.stringify(order)+")' data-target='#cancel_order_modal'>Скасувати замовлення</button></div>" +
                    "</div>" +
                    "</div>" +
                    "</div>");
                $("#approve_order_modal .close").click();
                removeValid("approve_order_form");
                clearForm("approve_order_form");
            }
        },
        error: function (data2) {
            console.log(data2.error);
        }
    });
}
function cancel(order){
    console.log("I'm in canceling");
    if(!validEmpty("cancel_order_reason")) return false;
    let order_id=order.order_id;
    let data= {
        "order_id": order_id,
        "message_ph":$("#cancel_order_reason").val()
    };
    let d = order.date.split(/[-T:]/);
    let date=d[2] + ' ' + months[parseInt(d[1], 10)].genitive + " " + d[0] + " ";
    $.ajax({
        url: 'http://localhost:2606/cancel',
        type: 'post',
        dataType: 'json',
        data: JSON.stringify(data),
        contentType: 'application/json',
        success: function (data2) {
```

```
            if(data2.success=="yes"){
                $("#order"+order_id).remove();
               order.message_ph=$("#cancel_order_reason").val();
                $("#cancelled_orders").append("<div class='panel panel-danger
mg-b-20' id='order"+order_id+"'>" +
                    "<div class='panel-heading pointer' data-toggle='collapse'
href='#ordercard"+order_id+"' aria-expanded='true'>" +
                    "<span class='pd-r-25'
id='date_order"+order_id+"'>"+date+"</span>" +
                    "<a class='link1'
href='/guest/"+order.username+"'>@"+order.username+"</a>" +
                    "<i class='fa fa-times mg-5 float-right'></i>" +
                    "</div>" +
                    "<div class='panel-body collapse show in'
id='ordercard"+order_id+"' aria-expanded='true' >" +
                    "<p>Тема: "+order.topic+"</p>" +
                    "<div class='row'>" +
                    "<div class='col-md-8'>" +
                    "<p class='text-break'>"+order.message_cl+"</p><p>Контакт
клієнта:"+order.contact_cl+"</p></div>" +
                    "<div class='col-md-4'></div><div class='col-md-4'></div>" +
                    "<div class='col-md-8'><p class='text-break'> Причина
відмови: <br>"+order.message_ph+"</p></div>" +
                    "<div class='col-md-12'><button class='border-none btn-block
bg-transparent light-green pointer' data-toggle='modal'" +
                    " onclick='setApproveOrderOnclick("+JSON.stringify(order)+")
' data-target='#approve_order_modal'>Підтвердити замовлення</button></div>" +
                    "</div></div></div>");
                $("#cancel_order_modal .close").click();
                removeValid("cancel_order_form");
                clearForm("cancel_order_form");
            }
        },
        error: function (data2) {
            console.log(data2.error);
        }
    });
}
function setSqlDate(month, day, year){
    let date = year + "-";
    if (month < 10)
```

```
        date += "0" + month+"-";
    else date += month+"-";
    if (day < 10)
        date += "0" + day;
    else date += day;
    return date;
}
function setBusy(ph_id, month, day, year) {
    let data={
        "ph_id":ph_id,
        "date":setSqlDate(month, day, year)
    };
    $.ajax({
        url: 'http://localhost:2606/removeactive',
        type: 'post',
        dataType: 'json',
        data: JSON.stringify(data),
        contentType: 'application/json',
        success: function (data2) {
            if(data2.success=="yes")
                $("#day" + day + "-" +
month).addClass("inactive").removeClass("active-date").attr("onclick",
"setFree(" + ph_id + "," + month + "," + day + "," + year + ")");
        },
        error: function (data2) {
        console.log(data2.error)
        }
    });
}
function setFree(ph_id, month, day, year) {
    let data={
        "ph_id":ph_id,
        "date":setSqlDate(month, day, year)
    };
    $.ajax({
        url: 'http://localhost:2606/setactive',
        type: 'post',
        dataType: 'json',
```

```
        data: JSON.stringify(data),
        contentType: 'application/json',
        success: function (data2) {
            if(data2.success=="yes")
                $("#day" + day + "-" + month).addClass("active-
date").removeClass("inactive").attr("onclick", "setBusy(" + ph_id + "," + month
+ "," + day + "," + year + ")");
        },
        error: function (data2) {
            console.log(data2.error)
        }
    });
  }
```

validation.js

```
//function for validation email with regex
function validEmail(str) {
    const selector = $("#" + str);
    let name = selector.val();
    if (name.length < 1 || name.length > 100) {
        selector.removeClass('is-valid');
        selector.addClass('is-invalid');
        return false;
    }
    let pattern = /^(([^<>()[\]\\.,;:\s@\"]+(\.[^<>()[\]\\.,;:\s@\"]+)*)|(\".
+\"))@((([0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+
[a-zA-Z]{2,}))$/;
    if (name.match(pattern)) {
        selector.removeClass('is-invalid');
        selector.addClass('is-valid');
        return true;
    } else {
        selector.removeClass('is-valid');
        selector.addClass('is-invalid');
        return false;
    }
}
//function for validation names with regex which match only cyrillic latin - and
'
function validNames(str) {
```

```
    const selector = $("#" + str);
    let name = selector.val();

    if (name.length < 2 || name.length > 45) {
        selector.removeClass('is-valid');
        selector.addClass('is-invalid');
        return false;
    }
    let letters = /^[a-zA-Za-яA-ЯiIїЇьЫэЭёЁъЪ\-']+$/;
    if (name.match(letters)) {
        selector.removeClass('is-invalid');
        selector.addClass('is-valid');
        return true;
    } else {
        selector.removeClass('is-valid');
        selector.addClass('is-invalid');
        return false;
    }
}
//function for validation names with regex which match only cyrillic latin - and
' and let empty value
function validNamesF(str) {
    const selector = $("#" + str);
    let name = selector.val();
    if (name == "") {
        selector.removeClass('is-valid');
        selector.removeClass('is-invalid');
        return true
    } else return validNames(str);
}
//function for validation password with regex which match only cyrillic latin -
and _ and more than 5 symbols
function validPassword(str) {
    const selector = $("#" + str);
    let pass = selector.val();
    if (pass.length < 6) {
        selector.removeClass('is-valid');
        selector.addClass('is-invalid');
```

```
        return false;
    }
    let pattern = /^[a-zA-Z0-9а-яА-ЯіІїЇ\-_]+$/;
    if (pass.match(pattern)) {
        selector.removeClass('is-invalid');
        selector.addClass('is-valid');
        return true;
    }
    selector.removeClass('is-valid');
    selector.addClass('is-invalid');
    return false;
}
//function for validation phone with regex - let digits and (123) 123 1234, 123-123-1234
function validPhone(str) {
    const selector = $("#" + str);
    let phone = selector.val();
    if (phone == "") {
        selector.removeClass('is-valid');
        selector.removeClass('is-invalid');
        return true
    }
    let pattern = /^\d+$/;
    let pattern2 = /^(\()?\d{3}(\))?(-|\s)?\d{3}(-|\s)\d{4}$/;
    if (phone.match(pattern) && phone.length == 10 || phone.match(pattern2)) {
        selector.removeClass('is-invalid');
        selector.addClass('is-valid');
        return true
    }
    selector.removeClass('is-valid');
    selector.addClass('is-invalid');
    return false;
}
//function for validation empty fields with regex
function validEmpty(str) {
    const selector = $("#" + str);
    let val = selector.val();
    let pattern = /^\s+$/;
```

```
    if (val == ""||val.match(pattern)) {
        selector.removeClass('is-valid');
        selector.addClass('is-invalid');
        return false;
    }
    selector.removeClass('is-invalid');
    selector.addClass('is-valid');
    return true;
}
//function for validation tags - let only letters and numbers
function validTags(str) {
    const selector = $("#" + str);
    let name = selector.val();
    if (name===="") {
        selector.removeClass('is-invalid');
        selector.removeClass('is-valid');
        return true;
    }
    let letters = /^[a-zA-Z0-9а-яА-ЯiIїЇ\s]+$/;
    if (name.match(letters)) {
        selector.removeClass('is-invalid');
        selector.addClass('is-valid');
        return true;
    } else {
        selector.removeClass('is-valid');
        selector.addClass('is-invalid');
        return false;
    }
}
//function which check if new username match any existed one
function validExistedUsername(str,users) {
    for(let u of users)
        if(str==u.username)
            return false;
    return true;
}
//function which check if new email match any existed one
function validExistedEmail(str,users) {
```

```
    for(let u of users)
        if(str==u.email)
            return false;
    return true;
}
//function which remove valid and invalid bootstrap classes
function removeValid(str) {
    $("form#" + str + " :input").each(function () {
        $(this).removeClass('is-valid');
    });
}
//function which clear all input values
function clearForm(str) {
    $("form#" + str + " :input").each(function () {
        $(this).val('');
    });
}
```