НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет
Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:
Завідувач кафедри
_____ Олександр Коваль
«___»_____2021 р.

Дипломна робота

на здобуття ступеня бакалавра
спеціальності 121 «Інженерія програмного забезпечення»
освітня програма «Інженерія програмного забезпечення розподілених систем»

на тему: «Web - система тегування датасетів зображень»

Виконав:
студент IV курсу, групи ТІ—71
Кузьменчук Дмитро Олександрович _____

Керівник:
к.т.н., доцент,
Шалденко Олексій Вікторович _____

Консультант:

_____

Рецензент:

_____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент (—ка) _____

Київ – 2021 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

спеціальності 121 «Інженерія програмного забезпечення»

освітня програма «Інженерія програмного забезпечення розподілених систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр Коваль
(підпис)
”___”_____2021р.

ЗАВДАННЯ
на дипломну роботу студенту

Кузьменчуку Дмитру Олександровичу
(прізвище, ім’я, по батькові)

1. Тема роботи Web – система тегування датасетів зображень

керівник роботи Шалденко Олексій Вікторович, к.т.н., доцент
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”____” травня 2021р. № _____

2. Строк подання студентом роботи _____

3. Вихідні дані до роботи _мови програмування Python та JavaScript, API системи для тегування датасетів

4.Зміст розрахунково—пояснювальної записки (перелік питань, які потрібно розробити) Провести аналіз існуючих систем в сфері тегування датасетів зображень. Розробити дизайн користувацького інтерфейсу. Визначити необхідні можливості для графічного редактора при тегуванні датасетів зображень. Обрати стек технологій який дозволить реалізувати всі можливості графічного редактора та розгорнути систему в хмарному сервісі. Розробити діаграму прецедентів. Спроектувати архітектуру бази даних та обрати сервер бази даних. Розробити всі необхідні модулі необхідні для функціонування системи. Поєднати модулі разом та провести тестування системи.

5. Перелік ілюстративного матеріалу Інтерфейс системи для тегування зображень ImageTagger, інтерфейс системи для тегування зображень LabelMe, Діаграма прецедентів, Діаграма взаємодії, Діаграма бази даних, Інтерфейс сторінки з проектами, Інтерфейс сторінки для створення проектів, Інтерфейс сторінки для створення робіт, Діалог завантаження тегованого датасету, Інтерфейс сторінки для виконання робіт

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання ”___”_____202___ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи		
2.	Вивчення та аналіз задачі		
3.	Розробка архітектури та загальної структури системи		
4.	Розробка структур окремих підсистем		
5.	Програмна реалізація системи		
6.	Оформлення пояснювальної записки		
7.	Захист програмного продукту	12.05.2021	
8.	Передзахист	25.05.2021	
9.	Захист	14.06.2021	

Студент _____ ___Кузьменчук Д. О.
(підпис) (прізвище та ініціали,)

Керівник роботи _____ _Шалденко О. В.
(підпис) (прізвище та ініціали,)

Анотація

Дипломна робота викладена на 60 сторінках, вона містить 32 рисунки, 3 додатки та 20 бібліографічних посилань.

Метою роботи є створення Веб—системи тегування датасетів зображень для подальшого використання при створенні тегованих датасетів які можуть застосовуватись при тренуванні моделей штучного інтелекту в сфері computer vision.

В ході виконання дипломної роботи було спроектовано та реалізовано систему яка дозволяє користувачу створювати та тегувати датасети які складаються з довільної кількості зображень будь—якого формату. Було створено графічний редактор який дозволяє відмічати об’єкти на зображеннях та маніпулювати ними за допомогою інтерфейсу з яким можна взаємодіяти шляхом натискання відповідних кнопок, а також «гарячих» клавіш на клавіатурі. Після створення продукту було проведено тестування системи з реальними користувачами на базі проведення практики.

Ключові слова: тегування, датасет, модель штучного інтелекту, computer vision, графічний редактор.

Abstract

The graduate research work is presented on 60 pages, it contains 32 figures, 3 appendices and 20 bibliographic references.

The aim of the work is to create a Web system for tagging image datasets for further usage in creating tagged datasets that can be used in the training of artificial intelligence models in the field of computer vision.

In the course of the work a system that allows the user to create and tag datasets consisting of any number of images of any format was designed and implemented. A graphical editor that allows you to mark objects in images and manipulate them using an interface which can be interacted with by pressing the appropriate buttons, as well as "hot" keys on the keyboard has been designed. After creating the product, the system was tested with real users on the practice basis.

Keywords: tagging, dataset, artificial intelligence model, computer vision, graphic editor.

ЗМІСТ

ВСТУП

На сьогоднішній день швидкими темпами розвивається штучний інтелект. Невід'ємним атрибутом для навчання моделі є датасет. Датасет може містити в собі найрізноманітніші дані такі як статистика про середню кількість опадів, кредитну історію людини, покупки товарів в інтернет магазині, контакти в соціальних мережах та багато багато іншого.

Наразі обчислювальні можливості комп'ютерів мають змогу обробляти як окремі зображення так і навіть відео які передаються в режимі реального часу. Проблемою для тренування таких систем є те, що без сторонньої допомоги неможливо автоматично оцінити результат роботи моделі і в подальшому його покращувати, тому в ході даної роботи було розроблено систему тегування датасетів зображень під назвою Oculum.

Дана задача є дуже актуальною тому, що завжди буде існувати потреба в створенні датасетів на яких щось відмічено, і ми не знаємо які об'єкти потрібно буде визначати в майбутньому.

Саме тому створення подібної системи яка дозволить будь-яким користувачам займатись тегуванням зображень є дуже важливою темою.

З використанням системи будь-який розробник або група розробників може перетворити набір сирих необроблених даних в систематизований та структурований датасет, при цьому можна як займатись цим власноруч так і залучати сторонніх людей.

Процес тегування передбачає нанесення простих геометричних фігур поверх зображень, тому з такою роботою може впоратись будь-яка людина не залежно від рівня технічної освіти та навичок до програмування. Тому при потребі можна за короткий час знайти людей, можливо в якості волонтерів, а може й на платній основі.

Будучи невід'ємною частиною в процесі навчання моделей датасет є основою для розвитку штучного інтелекту. Саме тому побудова нових систем з гнучким та інтуітивно зрозумілим інтерфейсом є дуже актуальною темою.

У першому розділі розглянуто постановку задачі для побудови системи для тегування датасетів зображень та визначено найважливіші цілі.

У другому розділі розглянуто переваги та недоліки існуючих систем призначених для тегування датасетів зображень.

У третьому розділі викладено можливості інструментів обраних для реалізації системи, а також причини їх вибору.

У четвертому розділі розкривається архітектура побудованої системи а також способи взаємодії з нею.

У п’ятому розділі детально розповідається про взаємодію користувача з інтерфейсом побудованої системи, а також про правила та обмеження.

1. ЗАДАЧА РОЗРОБКИ ВЕБ-СИСТЕМИ ТЕГУВАННЯ ДАТАСЕТІВ ЗОБРАЖЕНЬ

При розробці системи Oculum було поставлено наступні задачі:

— проаналізувати доступну літературу на базі практики;

— дослідити та порівняти існуючі системи з аналогічним призначенням;

— розробити модель роботи системи;

— визначити основні методи роботи з графікою для тегування датасетів зображень.

— вибрати стек технологій який дозволить реалізувати поставлену ціль;

— розробити алгоритми для розмічання елементів на зображеннях та їх збереження в форматі .json;

— розгорнути систему в хмарному сервісі та протестувати її;

— провести аналіз коду і виправити можливі недоліки;

— протестувати готову систему за допомогою реальних користувачів на базі практики.

При розробці графічного редактора для тегування зображень було визначено наступний перелік необхідних функцій та можливостей:

— виділяти області та ключові точки;

— рухати області та ключові точки;

— змінювати розміри областей;

— видаляти області та ключові точки;

— рухати зображення при цьому зберігаючи відносне положення елементів таких як області і ключові точки;

— збільшувати зображення при цьому зберігаючи положення центру і змінюючи положення елементів таких як ключові точки та області з огляду на те що виділена зона має інше положення та більші лінійні розміри а також збільшувати пропорційно виділені області;

— зменшувати зображення зі збереженням центру та відносного положення інших елементів подібно до попереднього пункту;

— вказувати яка емоція відповідає обличчю на виділеній області а також вказувати на який елемент обличчя вказує ключова точка.

Створений програмний продукт повинен відповідати наступному ряду вимог та характеристик:

— запускатися на будь—якому пристрої на якому встановлено браузер з підтримкою HTML, CSS та JavaScript;

— мати зручний графічний редактор і інтуїтивно зрозумілим інтерфейсом;

— мати підказки для користувача які говорять про призначення тих чи інших елементів інтерфейсу в разі якщо щось було не зрозуміло;

— підтримувати роботу з будь—якими типами зображень;

— динамічно реагувати на дії користувача.

Основними цілями роботи системи є:

— Створення тегованих датасетів з можливістю відмічати на них зони з обличчям, ключові точки лиця та емоції на обличчях з ціллю подальшого використання при навчанні моделей в такому розділі машинного навчання як Computer Vision.

— Залучення людей для швидкого розмічання датасетів.

2. ОПИС ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ ПРИ РОЗВ'ЯЗАННІ ЗАДАЧІ — ТЕГУВАННЯ ДАТАСЕТІВ ЗОБРАЖЕНЬ

Темою моєї дипломної роботи є тегування датасетів зображень. Датасет — це певний набір впорядкованих та структурованих даних у моєму випадку зображень.

Тегування — процес додавання тегів (інформації про те що зображено в цілому, або про якусь конкретну область зображення) до кожної складової датасету. Зображення — це набір пікселів, тому, щоб додати до нього теги необхідно розмітити на ньому певну область або певні ключові точки. З якою метою це все робиться? Це необхідно в процесі тренування моделей моделей штучного інтелекту для того щоб покращувати точність роботи з кожною ітерацією.

Для втілення вище сказаного необхідно програмний продукт, який:

1) дає можливість відмічати необхідні області та ключові точки;

2) дозволяє отримувати результати тегування в форматі який можна використовувати для подальшого тренування моделей;

3) працює з будь—якими форматами зображень.

Розглянемо системи призначені для виконання подібних задач, проаналізуємо їхні слабкі та сильні сторони.

2.1 Інструмент для тегування зображень ImageTagger

На рисунку 2.1 зображено інтерфейс головної сторінки представленої системи. Тут відображено інформацію про перелік доступних датасетів (область під номером 1), перелік команд та засоби для їх створення (область під номером 2), Форма для створення нового датасету (область під номером 3).

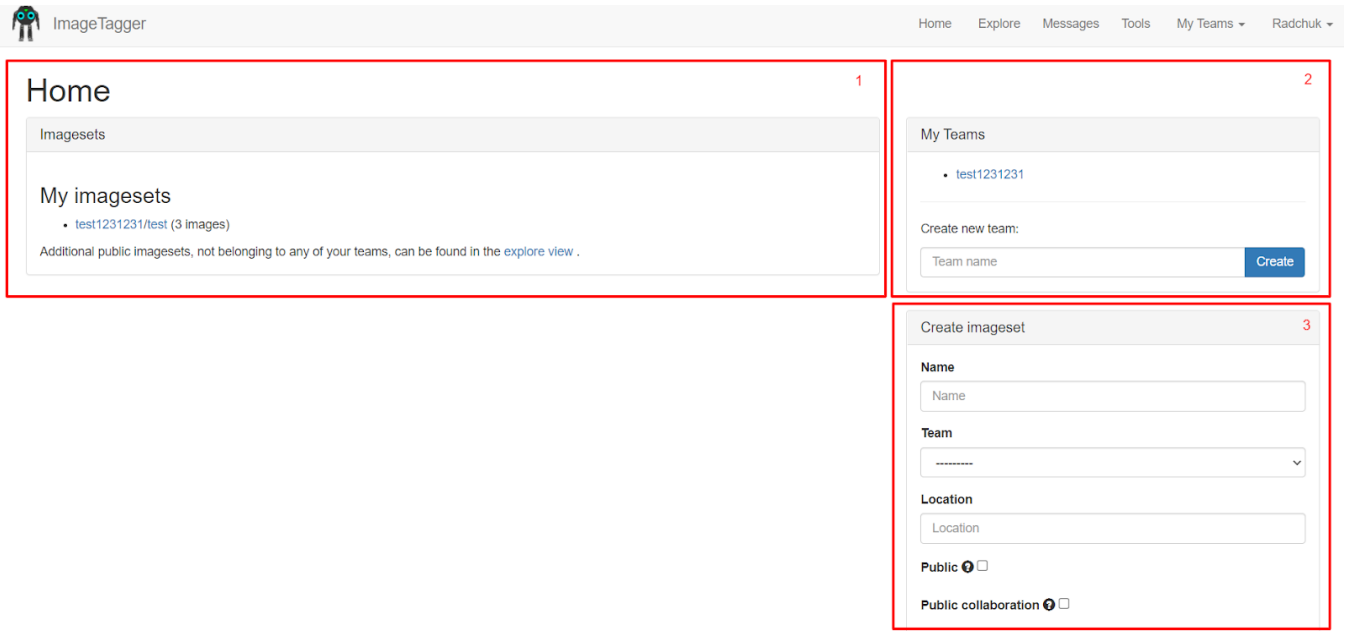

Рисунок 2.1 — Головна сторінка програмного продукту ImageTagger.

При створенні датасету можна вказати ряд параметрів за якими в подальшому можна буде ідедентифікувати цей датасет а саме: його назва, назва команди яка його створила і відповідає за тегування, з яким географічним місцем пов’язаний створений датасет а також можна вказати параметри доступу, тобто відкрити доступ всім бажаючим або обмежити доступ лише для обмеженої групи довірених людей.

На рисунку 2.2 можна побачити сторінку для перегляду інформації про датасет. Серед інформації є відомості про те якого розміру датасет (скільки зображень в нього входить), інформацію про те скільки зображень вже було протеговано. Також система надає можливість завантажити початковий датасет в форматі архіву з розширенням .zip або завантажити датасет безпосередньо у вказану директорію з використанням файлу зі скриптом використовуючи інтерфейс командного рядка. Скрипт написаний на мові програмування Python, тому перед його використання потрібно попередньо встановити цю мову собі на машину. Сам файл зі скриптом можна завантажити на окремій сторінці в межах цього сайту.

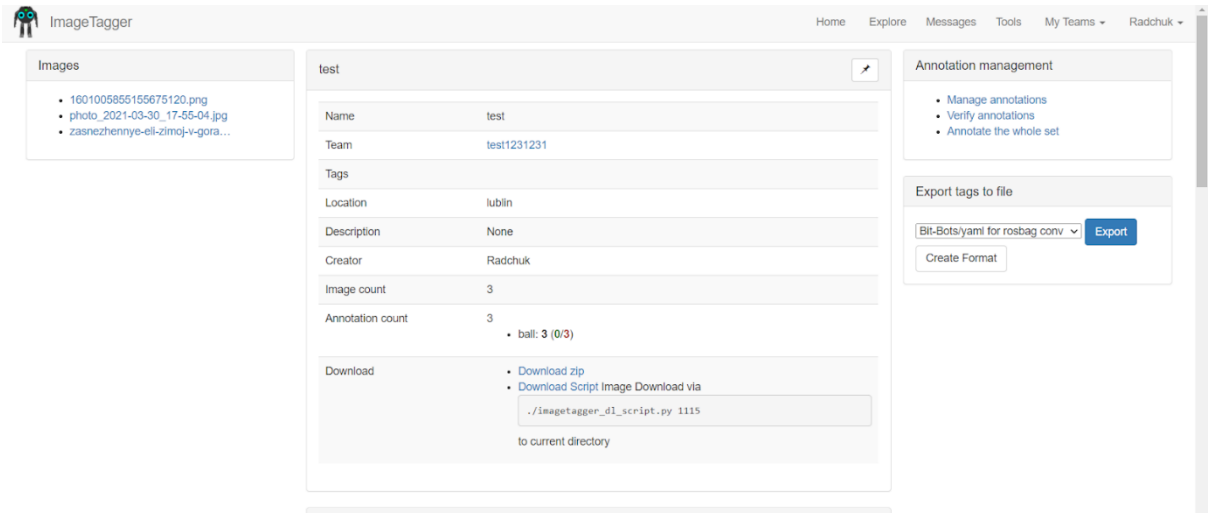

Рисунок 2.2 — Сторінка з інформацією про датасету ImageTagger

На рисунку 2.3 можна побачити інтерфейс сторінки для редагування налаштувань пов’язаних з певним датасетом. Система дозволяє встановити такі параметри як пріоритет, що можна використовувати для організації порядку виконання завдань. Тип анотацій дозволяє вибрати один з варіантів для тегування залежно від того які саме речі потрібно відмітити на зображеннях. Також є набір кнопок призначених для завантаження нових картинок до датасету та управління процесом завантаження. Також можна переглядати статус завантаження кожного окремого зображення.

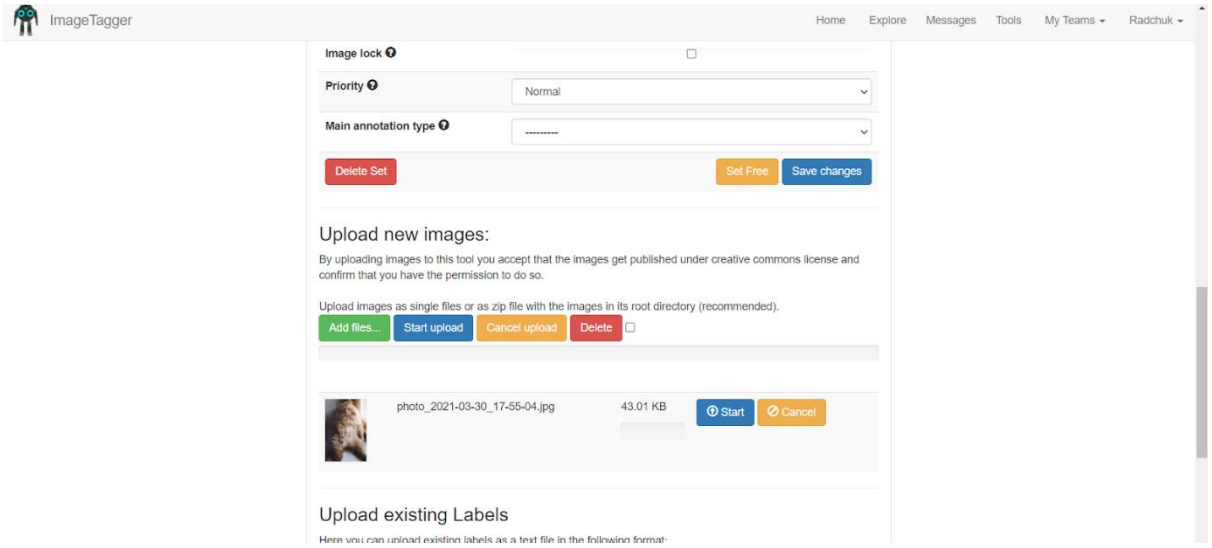

Рисунок 2.3 — Редагування датасету ImageTagger

На рисунку 2.4 зображено інтерфейс сторінки для тегування датасету зліва знаходиться панель на якій виводиться список зображень для яких відсутні теги а також можливість фільтрувати їх за назвою. По середині знаходиться одне з зображень а під ним форма для редагування інформації про анотацію. Зліва знаходиться панель з випадаючим списком анотацій, а під ним кнопки для збереження та очищення анотацій а також для переходу між зображеннями.

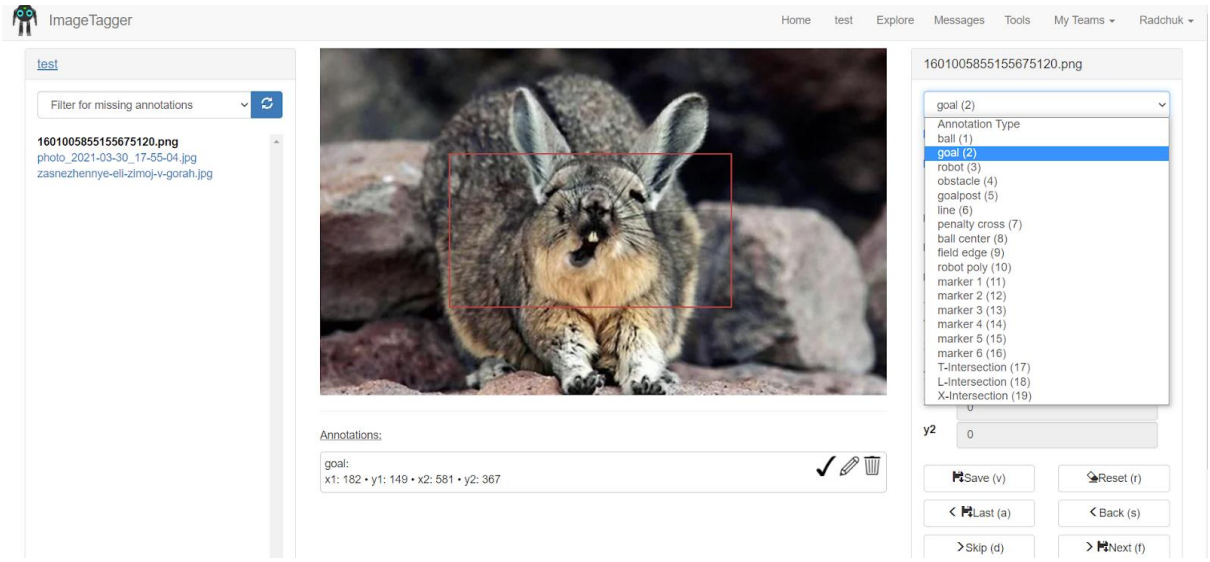

Рисунок 2.4 — Тегування датасету ImageTagger

Переваги: Застосунок достатньо простий і надає можливість відразу завантажити цілий датасет зображень, тегувати його та завантажувати результати роботи у декількох різних форматах. Є можливість обмежити доступ до проекту зробивши його приватним або відкритим, а також можна створювати команди і таким чином організовувати колективну роботу. Проект уже розгорнутий на сайті тому кожен хто має посилання може зайти на нього та працювати не маючи технічної освіти чи навичок програмування. Ще одним з плюсів системи є відкритий код.

Недоліки: Список анотації реалізований за допомогою випадаючого списку, який налічує достатньо багато різноманітних тегів з якими буде дуже важко розібратися новому користувачу. У випадку коли зображень багато, переходити між ними незручно. Анотації потрібно зберігати вручну, після створення кожної з них,

необхідно використовувати функцію збереження, ця операція збільшує час, потрібний для виконання задачі.

2.2 Інструмент для тегування зображень LabelMe

На рисунку 2.5 зображено інтерфейс сторінки для тегуваня датасетів в системі LabelMe. Система надає такі можливості як: навігація між зображеннями в датасеті, відміна попередньої дії, відміна поверненої дії, збільшення зображення, зменшення зображення та видалення тегів. Як видно з зображення процес додавання тегів складається з нанесення на картинку ряду точок які об’єднують між собою в порядку додавання і створюють замкнений контур всередині якого міститься певний предмет особа або щось інше. Після виділення області таким чином потрібно вказати що саме знаходиться в ній а також можна змінити колір для виділення.

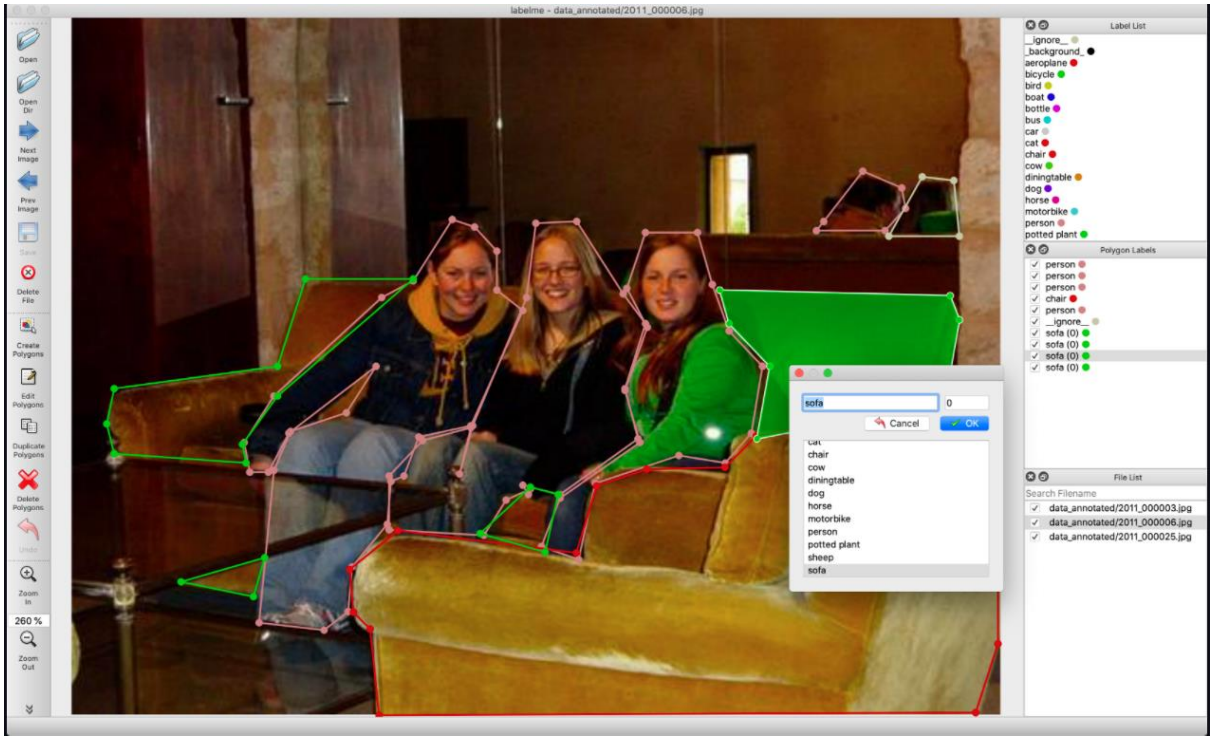

Рисунок 2.5 — Тегування картинок LabelMe

Система надає можливість обробляти відео, але для того щоб перетворити відео у набір зображень, необхідно виконати з командного рядка де запущено програму

команду в коді. (python convert_video_to_images your_video_file.*). Помістити отримані зображення в окрему директорію і завантажити використовуючи застосунок. Для завантаження датасету потрібно також виконати наступну команду з командного рядка де запущено програму (/labelme2voc.py annotated_data labels.txt).

Переваги: Застосунок має достатньо зрозумілий та простий інтерфейс який дозволяє створювати датасет з набору зображень, або відео файлів які було розділено на кадри, тегувати його та отримувати результати роботи у вигляді файлу з розширенням json. Продукт має відкритий початковий код.

Недоліки: Попри те що можна працювати з відео файлами, операції над ними потрібно виконувати з інтерфейсу командного рядка що не завжди можливо для звичайних користувачів які не мають профільної технічної освіти та навичок в програмуванні. Анотації необхідно завантажувати використовуючи командний рядок. Створення тегів обмежено лише однією можливістю такою як виділення областей. Дизайн виглядає застарілим. Проект потрібно самостійно розгортати. Для того щоб розгорнути проект, потрібно вміти користуватись такими технологіями як docker, python та postgres.

3. ЗАСОБИ РОЗРОБКИ

3.1 Мова програмування Java Script

Інструмент Java Script (JS) – це високорівнева мова програмування яка відповідає специфікації ECMAScript. JS є інтерпретованою і мультипарадигмною мовою. Основними рисами є динамічна типізація, синтаксис з фігурними дужками, функції як об’єкти першого класу, автоматичне управління ресурсами такими як пам’ять та прототипне програмування.

В сукупності з HTML та CSS, JS є одною з головних технологій у Веб програмуванні. Більш ніж 97% усіх вебсайтів у світі використовують JS на стороні клієнта для того щоб зробити сторінки інтерактивними. На сьогоднішній момент всі популярні браузери мають механізм для виконання JS коду на пристрої користувача.

Будучи мультипарадигмною JS підтримує керований подіями, функціональний та імперативний стилі програмування. JS має API для роботи з текстовими даними, датами і часом, регулярними виразами, стандартними структурами даних а також об’єктною моделлю документу [8].

3.2 Бібліотека для побудови інтерфейсів React

Інструмент React – це декларативна, гнучка та ефективна бібліотека JavaScript для побудови користувацьких інтерфейсів. React дозволяє будувати комплексні інтерфейси з малих та ізольованих один від одного фрагментів коду які називаються компонентами. Це дозволяє декомпозувати інтерфейс і тим самим спрощує розробку.

React дозволяє швидко і зручно створити інтерактивний інтерфейс. Можна розробити дизайн у відповідності ко кожного стану застосунку, і React сам буде реагувати на зміни у стані і обновляти та перемальовувати лише ті компоненти стан яких змінився [9].

3.3 Пакувальник статичних модулів Webpack

Інструмент Webpack – це пакувальник статичний модулів з відкритим кодом який призначений для сучасних JavaScript застосунків. Під час обробки проекту webpack будує граф залежностей який відображає всі необхідні модулі і як результат генерує один або більше пакетів. За замовчуванням webpack розуміє тільки файли з розширенням .js та .json. За допомогою відповідних завантажувачів можна дозволити обробляти й інші типи файлів конвертуючи їх у валідні модулі які можуть використовуватись застосунком, та додати їх до графу залежностей [10, 11].

3.4 Регістр програмного забезпечення npm

Інструмент npm – це найбільший у світі регістр програмного забезпечення. Його використовують розробники по всьому світу для того щоб ділитися бібліотеками з відкритим кодом, також багато організацій використовує npm для управління приватною розробкою. npm складається з трьох окремих компонентів:

— вебсайт;

— інтерфейс командного рядка (CLI);

— регістр.

Вебсайт використовується для того щоб знаходити необхідні бібліотеки, управляти профілями, доступом та іншими аспектами. На приклад ви можете створити організацію для управління доступом до публічних та приватних пакетів.

Інтерфейс командного рядка — це найбільш вживаний розробниками інструмент призначений для установки, видалення та обновлення пакетів.

Регістр – це величезна публічна база даних з програмним забезпеченням та супроводжуючою його мета інформацією написані на JavaScript [12].

3.5 Мова розмітки HTML

Інструмент HTML – це мова розмітки гіпертексту яка використовується для розмічання веб сторінок у всесвітній мережі інтернету. Елементи HTML є будівельними блоками з яких створюється все що ми бачимо на будь—якій сторінці у браузері. Синтаксис передбачає використання тегів які обгорнуті в кутові дужки. Теги можуть бути одинарні (коли в них нічого немає) та парними. HTML дозволяє вкладати теги один в одного надаючи розробнику гнучкий інструмент [15, 16, 17, 20]. HTML надає засоби для створення таких важливих елементів як:

— заголовки, абзаци, таблиці, нумеровані та марковані списки, цитати та інше;

— посилання для переходу на інші сторінки та для завантаження інформації;

— створення форм для вводу текстової інформації та завантаження файлів;

— включення картинок, звуку, відео а також інших медіа об’єктів;

3.6 Каскадні таблиці стилів CSS

Інструмент CSS – це каскадні таблиці стилів які використовуються розробниками для задання елементам веб сторінки написаним на HTML таких характеристик як ширина, висота, колір, відступи, рамки, положення відносно інших елементів та багато багато інших [18, 19, 20]. CSS дозволяє задати стилі для бажаних елементів вказавши селектор тегу або створивши селектор класу. Стилі для кожного селектора записують в фігурних дужках, для того щоб задати стиль потрібно вказати його назву та значення після двокрапки. Після кожного стилю потрібно ставити крапку з комою. Також CSS дозволяє задавати стилі для компонентів при виникненні певних подій таких як: наведення курсору на кнопку, зміна стану посилання після переходу за ним та інші. Особливістю CSS є те, що стилі батьківського елементу можуть бути перевизначені при повторному присвоєнні такого само стилю дочірньому елементу.

3.7 Бібліотека для роботи з графікою Fabric.js

Інструмент Fabric.js – це потужна бібліотека з відкритим кодом написана на JavaScript і призначена для роботи з HTML5 та canvas [13, 14]. Бібліотека включає в себе об’єктну модель яка дозволяє працювати з об’єктами більш зручним чином ніж при аналогічній роботі з canvas. Зручність полягає в тому, що маючи посилання на екземпляр об’єкту ми можемо змінювати його як захочемо і при цьому Fabric.js бере на себе всю роботу по перемальовуванні, також слід відмітити, що при зміні одного об’єкту інші не будуть перемальовуватись. Fabric.js надає 7 типів базових фігур:

— fabric.Circle
— fabric.Ellipse
— fabric.Line
— fabric.Polygon
— fabric.Polyline
— fabric.Rect
— fabric.Triangle

3.8 Високорівневий веб—фреймворк Django

Інструмент Django — це високорівневий веб—фреймворк написаний на такій мові програмування як Python [1]. Він який дозволяє за короткий термін створювати безпечні та такі в які можна швидко вносити зміни не боячись зруйнувати всю архітектуру веб—сайти. створений розробниками які мають великий досвід роботи та високу кваліфікацію, Django бере на себе більшу частину рутинних завдань які з’являються в процесі веб—розробки, тому розробники можуть направити всі свої сили на написання бізнес логіки і при цьому не потрібно винаходити велосипед. Цей фреймворк має відкритий початковий код, а також процвітаючу та активну спільноту яка налічує десятки тисяч людей зі всього світу в яких можна при необхідності отримати консультацію на відповідних сайтах. Фреймворк має прекрасну

документацію яка описує всі аспекти роботи з ним та величезну кількість варіантів підтримки як безкоштовної так і на платній основі. Цей фреймворк — є основою на якій побудовано бекенд моєї системи. Розглянемо головні принципи які утворюють філософію програмування з використанням Django:

— Завершеність(Complete): Django слідує принципу "Батареї в комплекті" (Batteries included) і надає базову, готову реалізацію для всього що розробники зазвичай можуть реалізувати "стандартно". Так як все, що потрібно розробнику, є складовою частиною одного загального "продукту", все це безперервно працює разом, підпорядковуючись послідовним принципам дизайну та підкріплюється великою кількістю детальної та змістовної документації яка дозволяє швидко у всьому розібратися.

— Універсальність (Versatile): Django може застосовуватися (і застосовується) для побудови практично будь—якого виду веб—сайтів — від систем керування матеріалами будь—якого типу та сайтів з довідковою інформацією до соціальних мереж, інтернет магазинів та сайтів з актуальними новинами. Він може підтримувати будь—який формат на клієнтській стороні і доставляти інформацію практично у всіх форматах (серед яких HTML, JSON, RSS—канали, XML та багато інших). Хоча фреймворк надає величезну кількість опцій для роботи (наприклад, підключення до найпопулярніших баз даних та засобів шаблонування тощо), може виникнути така ситуація коли стандартних засобів недостатньо, в такому випадку є можливість розширити базовий функціонал створивши свій власний компонент який буде задовольняти специфічні потреби.

— Безпека (Secure): Django застерігає розробників від вчинення ряду типових помилок які можуть виникнути в процесі розробки програмного забезпечення та призводити до вразливості системи в плані безпеки, надаючи каркас, розроблений для самостійного захисту системи який спрямовує дії розробників у правильне русло. Наприклад, Django надає можливість безпечно управляти конфіденційною інформацією про облікові записи користувачів та відповідну інформацію необхідну для авторизації, Django вирішує таку типову помилку як зберігання інформації про сесію у файлах cookie, де її можуть викрасти зловмисники (замість цього в файлах

cookie зберігається ключ, а реальні дані знаходяться в базі даних застосунку) або пряме зберігання оригінальних паролів замість їх хешу. Django автоматично надає захист від основних зловмисницьких операцій, таких як SQL ін’єкції, сценарії між сайтами, фальсифікація запитів між сайтами.

— Масштабованість (Scalable): Django використовує архітектуру яка базується на застосуванні компонентів "нічого спільного" (shared—nothing) (всі компоненти розробляються паралельно один від одного, це зменшує складність системи за рахунок декомпозиції і дозволяє швидше вносити зміни в систему та за потреби навіть замінити певний модуль цілком). Декомпозиція системи в набір пов’язаних між собою компонентів означає, що система може реагувати на зростання трафіку, шляхом збільшення кількості обладнання на тому рівні де це потрібно, наприклад: сервери баз даних, сервери кешування або сервери застосунків. Є приклади сайтів які були написані на Django та успішно масштабували свої потужності для того щоб задовольнити зростаюче навантаження спричинене зростанням кількості користувачів та відповідно їх активністю (деякими прикладами можуть бути такі сайти як, Instagram чи Disqus).

— Портативність(Portable): Django створений на одній з найбільш популярних мов програмування такій як Python, що підтримується на всіх популярних в наш час платформах. Тому це відкриває можливості для розгортання своєї системи на різних операційних системах таких як Linux, Mac OS X та Windows. Також, Django охоче підтримується всіма великими провайдерами веб—хостингу, які зазвичай надають шаблони деякої інфраструктуру та документації призначеної для розгортання сайтів Django.

3.9 Інструмент для створення REST запитів DRF

Схеми API — це цінний інструмент, що дає можливість користуватися широким спектром функцій серед яких, генерація довідкової інформації або управління динамічними бібліотеками клієнтів, які мають можливість працювати з

вашим API. Також Django REST Framework бере на себе відповідальність за генерацію схем OpenAPI [2].

3.9 Інтерпретована мова програмування Python

Інструмент Python — інтерпретована, високорівнева мова програмування, що дозволяє виконувати широкий спектр задач в різних. Філософія програмування з використанням Python наголошує на читабельності коду за допомогою значного використання відступів. Його мовні конструкції, а також об’єктно—орієнтований підхід мають на меті допомогти програмістам писати чіткий, логічний код для малих та великих проектів. Попри те, що Python програє в швидкості багатьом мовам зі статичною типізацією, але він має зручний синтаксис який відомий тим, що багато виразів можна написати в одному рядку коду тим самим спрощуючи процес розробки і зменшуючи необхідну кількість часу для розробки [7].

4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Для організації розробки системи при проектуванні було створено наступні діаграми:

— діаграма прецедентів Use case;

— діаграма бази даних;

— діаграма взаємодії.

4.1 Діаграма прецедентів

На рисунку 4.1 видно що в системі є дві ролі для користувачів, а саме:

— адміністратор;

— звичайний користувач.

Система надає два варіанти для створення робіт:

— з набору зображень;

— з відео шляхом його розбиття на кадри між якими однаковий часовий інтервал (в секундах).

При розміченні картинок в роботі доступно три опції:

— виділення області з обличчям;

— розмітка ключових точок на обличчі;

— вказання емоцій на обличчі.

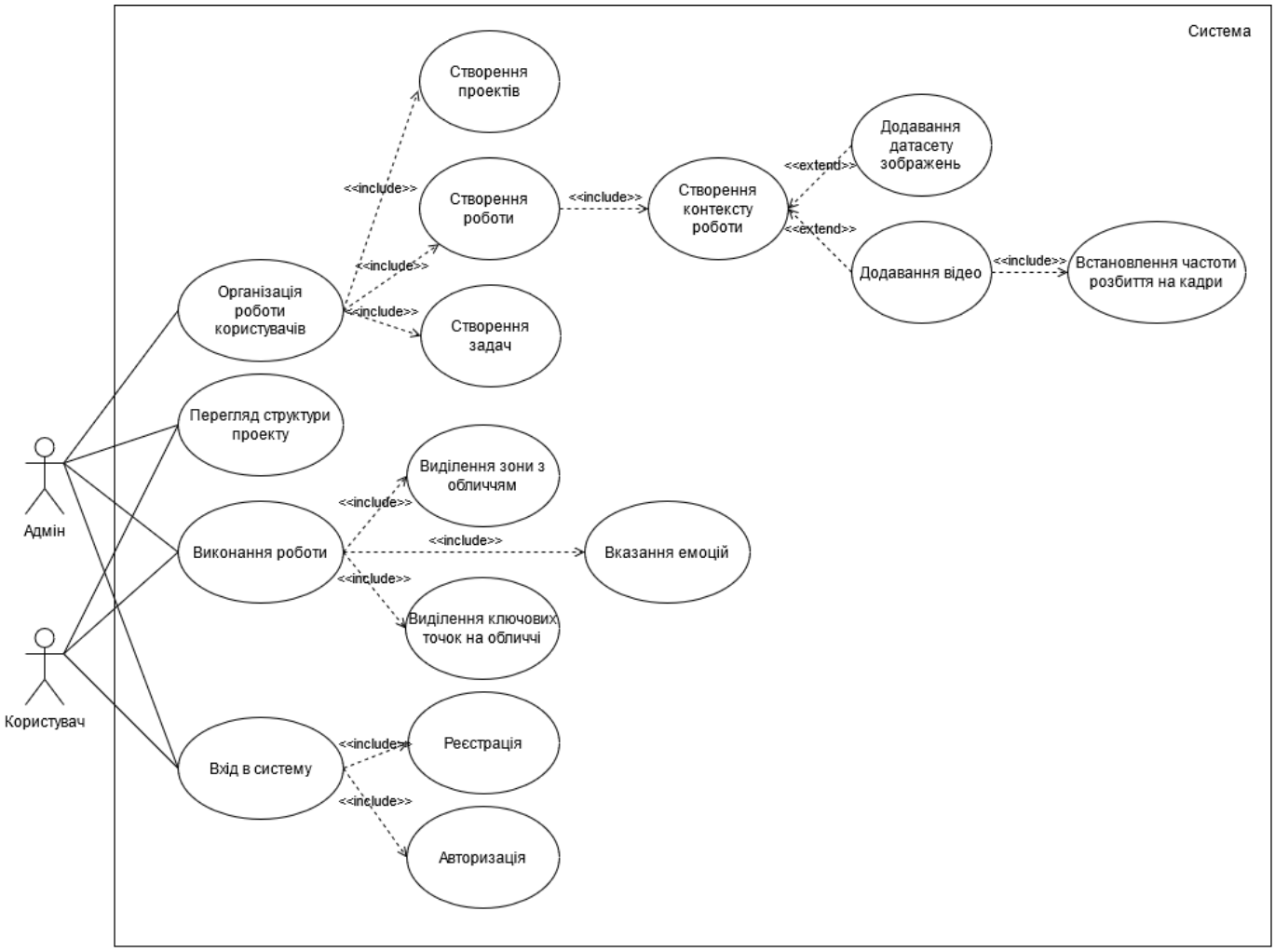

Рисунок 4.1 – Діаграма прецедентів

Також можна помітити що звичайні користувачі на відміну від адміністраторів не володіють такими повноваженнями як створення проектів, задач та робіт а також видаленням проектів задач та робіт. До спільних можливостей належать такі дії як перегляд проектів задач та робіт а також виконання робіт. Слід зауважити що при реєстрації нового користувача в системі не можливо відразу створити адміністратора такими виключними повноваженнями володіє тільки системний адміністратор який створюється з інтерфейсу командного рядка де запущено бекенд написаний на Django і доступ до якого мають безпосередньо тільки розробники системи. Таким чином тільки системний адміністратор може відразу створити іншого звичайного адміністратора або підвищити вже існуючого користувача до цієї ролі.

4.2 Діаграма взаємодії

На рисунку 4.2 зображено наступні сценарії взаємодії з системою:

— створення проектів, задач та робіт;

— видалення проектів, задач та робіт;

— завантаження тегованого датасету для задачі;

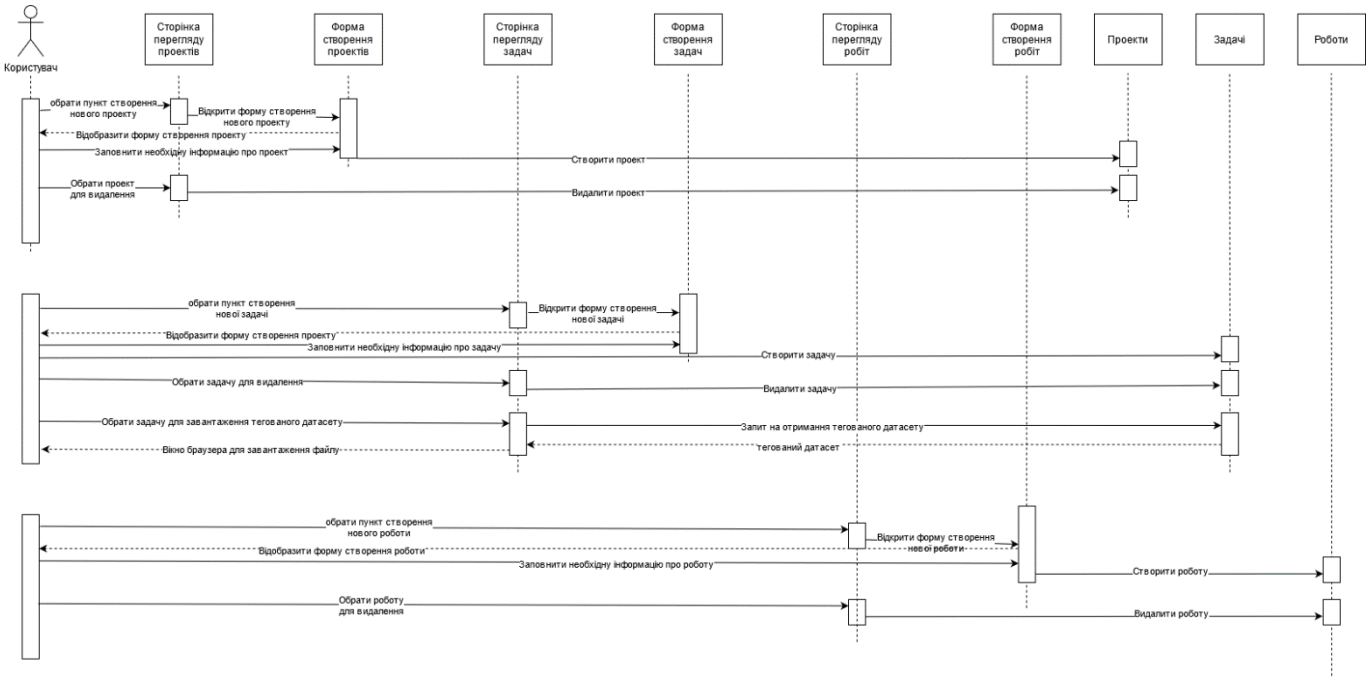

Рисунок 4.2 – Діаграма взаємодії

На рисунку 4.2 показано що кожен структурний елемент такий як проект, задача чи робота прив’язані до окремої моделі в базі даних і взаємодіяти з ними можна з окремих сторінок при цьому звичайні користувачі можуть лише переглядати їх, а дії такі як створення, видалення чи завантаження датасету можуть виконувати тільки адміністратори. Для того щоб створити проект задачу чи роботу необхідно натиснути відповідну кнопку та перейти на сторінку для створення після чого заповнити всі необхідні поля для створення і відправити запит. Для того щоб видалити проект задачу чи роботу потрібно відкрити випадаюче меню з діями розташоване в нижньому лівому куті картки та натиснути кнопку видалити. Після успішного видалення відповідний проект задача чи робота будуть вилучені зі списку для

перегляду. Також слід зауважити, що тегований датасет можна завантажити лише на рівні задачі у тому випадку коли хоча б одна з картинок що належить до робіт які входять до даної задачі вже була протегована.

На рисунку 4.3 зображено схему для входу користувача в додаток та схему для отримання робіт.

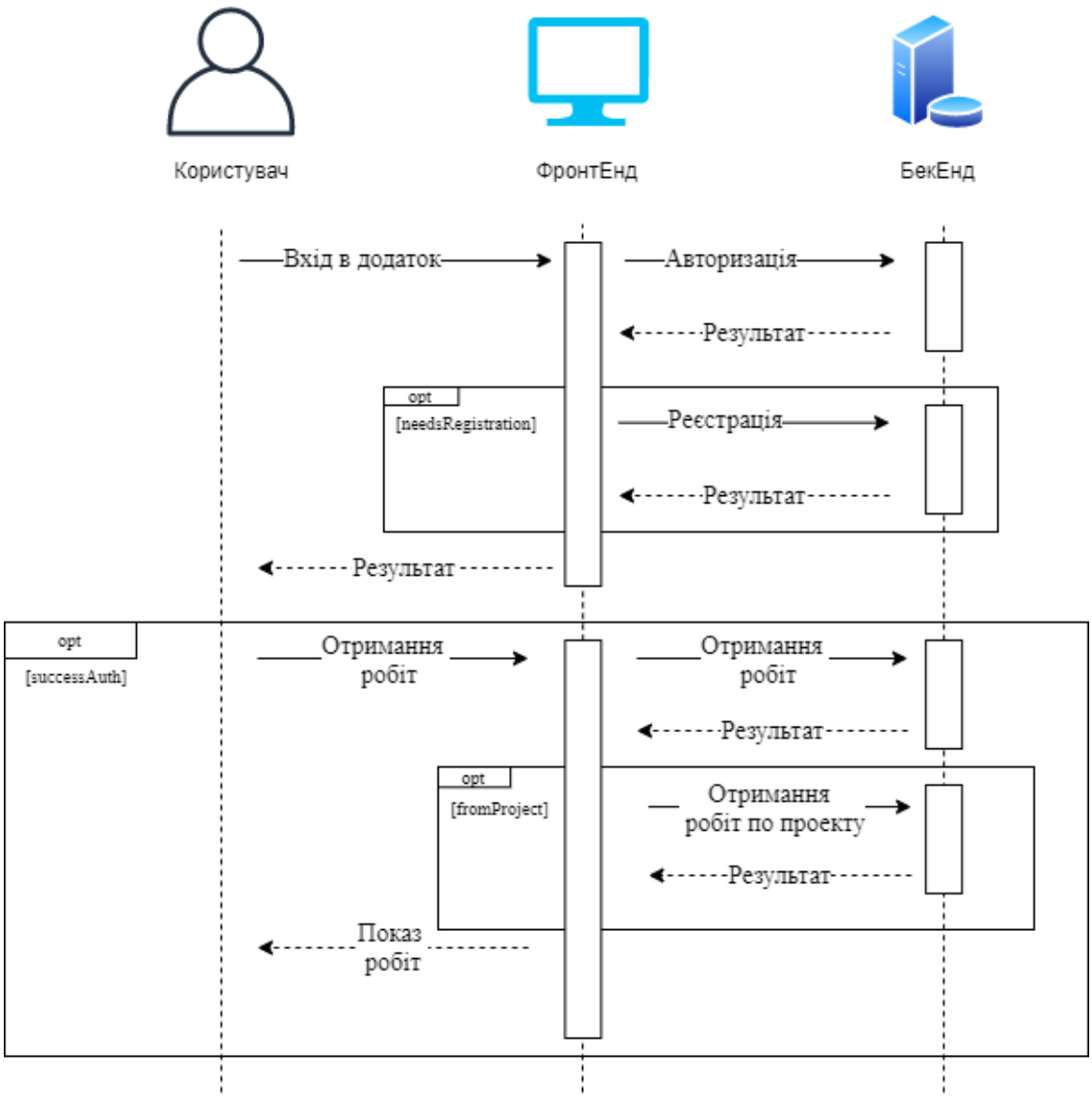

Рисунок 4.3 – Діаграма взаємодії(2)

На рисункуН 4.3 видно, що всі користувачі незалежно від ролі повинні авторизуватися при вході в додаток. У випадку коли користувач не має облікового запису він може створити новий на сторінці реєстрації. Після входу в додаток

користувач може виконувати будь—які дії які дозволені для його ролі або нічого не робити.

4.3 Діаграма бази даних

На рисунку 4.4 зображено діаграму бази даних системи Oculum на якій представлено всі сутності а також зв'язки між ними. В системі використовується реляційна база даних Postgres [3].

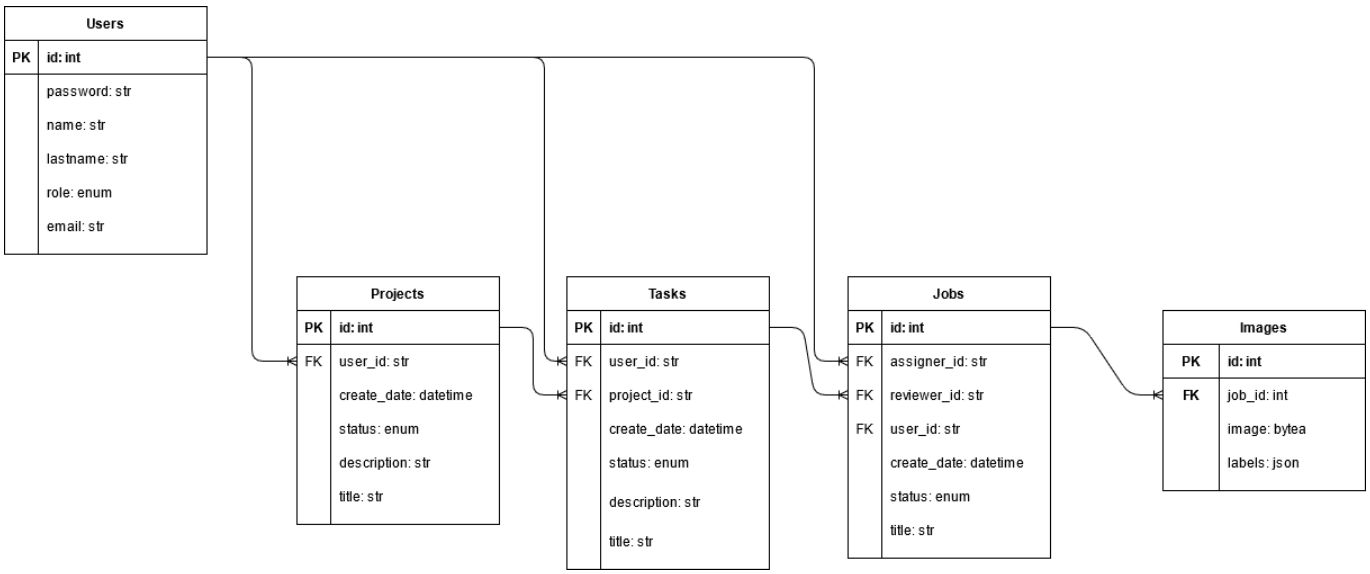

Рисунок 4.4 – Діаграма бази даних

В системі присутні наступні сутності:

— користувач;

— проект – структура найвищого рівня яка може описувати якийсь реальний проект чи стартап який потребує тегований датасет для своїх потреб;

— задача – структура яка містить в собі певну кількість робіт і для якої доступна функція завантаження датасету;

— робота – структура яка містить в собі певний набір зображень які може розмітити користувач за відносно невеликий відрізок часу;

— картинка – складова частина роботи яка може містити інформацію про елементи які відмітив користувач.

Такі сутності як проекти задачі та роботи мають набір спільних полів які описують загальні характеристики да них відносяться:

— Ідентифікатор який представляє собою автоінкрементне цілочисельне беззнакове поле розміром 4 байти тобто 32 біти за допомогою якого можна зберігати 4294967296 унікальних записів.

— Дата створення вказує на те коли саме було створено проект, задачу чи роботу. Дата зберігається в форматі TIMESTAMP.

— Статус відображає стан завершеності складових проекту, задачі чи роботи. Статус може набувати трьох різних значень таких як: створено, в процесі та виконано. Проект набуває статусу виконано лише в тому випадку коли всі задачі які до нього належать були успішно виконані, при цьому якщо в проекті немає жодного завдання він не буде рахуватися виконаним, а буде знаходитись у статусі створено. Задача переходить у стан виконано лише у тому випадку коли всі роботи які належать до неї було виконано, також подібно до проектів задача не може бути виконаною коли в ній нема робіт. Робота переходить у стан виконано лише в тому випадку коли всі зображення з яких вона складається були протеговані та результати було збережено в базі даних застосунку. Проект набуває статусу в процесі у тому випадку коли хоча б одна з задач було успішно виконана. Задача переходить у статус в процесі коли було успішно виконано хоча б одну з робіт. Робота отримує статус в процесі в тому випадку коли хоча б одна з картинок була протегована і результати були збережені в базі даних застосунку.

— Опис є текстовою інформацією що може давати короткий опис тій чи іншій сутності. У випадку коли це проект там можуть міститись відомості про замовника такі як ім’я, прізвище, контактні дані організацію яку він представляє, напрямок роботи або щось інше. У коли це завдання там можуть міститись вимоги до розмічання зображень. У разі коли це робота там можуть міститись відомості про терміни виконання.

— Назва є рядком що називає проект завдання чи роботу. Слід зауважити що назва не є унікальним полем і за потреби може дублюватися з іншим уже створеним проектом завданням чи роботою.

— Автор є зовнішнім ключом який вказує хто саме з користувачів системи створив проект завдання чи роботу.

Всі завдання в системі мають зовнішній ключ який вказує до якого проекту вони належать. У випадку якщо проект було видалено всі належні до нього завдання будуть видалені за каскадом.

Всі роботи в системі мають зовнішній ключ який вказує на завдання до яких вони відносяться і у випадку якщо завдання було видалено всі роботи належні до нього будуть також видалені за каскадом. Також між роботами та проектами і завданнями є така значна відмінність як те, що завдання мають обов’язкові поля такі як ідентифікатор користувача на якого призначено завдання та ідентифікатор користувача який повинен перевірити правильність виконання завдання. Ідентифікатор користувача на якого було призначено завдання служить для того щоб відображати для звичайних користувачів лише ті роботи які вони повинні виконати і ні які інші. Ідентифікатор користувача який повинен перевіряти роботи служить для того щоб вказати адміністратора який повинен перевірити коректність виконання. За замовчуванням користувачем який повинен перевіряти роботи є той адміністратор хто створив ту чи іншу роботу, при бажанні користувача відповідального за перевірку роботи можна змінити використовуючи адміністраторську панель Django.

Кожна робота складається як мінімум з одного зображення. Всі зображення мають унікальний ідентифікатор, поле з масивом байтів в якому зберігається саме зображення та поле в форматі json в якому зберігається інформація про теги які відмітив користувач. У випадку коли робота до якої належали зображення була видаленою, зображення також видаляються за каскадом.

Користувач є сутністю яка містить в собі такі поля як логін та пароль що зберігаються в зашифрованому виді щоб ніхто не зміг їх скомпрометувати, також в ній зберігається інформація яка ідентифікує особу користувача це ім’я та прізвище.

4.4 Запити до API

Для роботи системи на бекенді було створено ряд ендпоінтів які відповідають за наступний ряд дій:

— авторизація та аутентифікація;

— створення сутностей;

— видалення сутностей;

— модифікація сутностей;

— отримання інформації.

На рисунку 4.5 зображено структуру запиту для отримання токену доступу. Для того щоб отримати токен необхідно передати два обов’язкових параметри таких як логін та пароль.

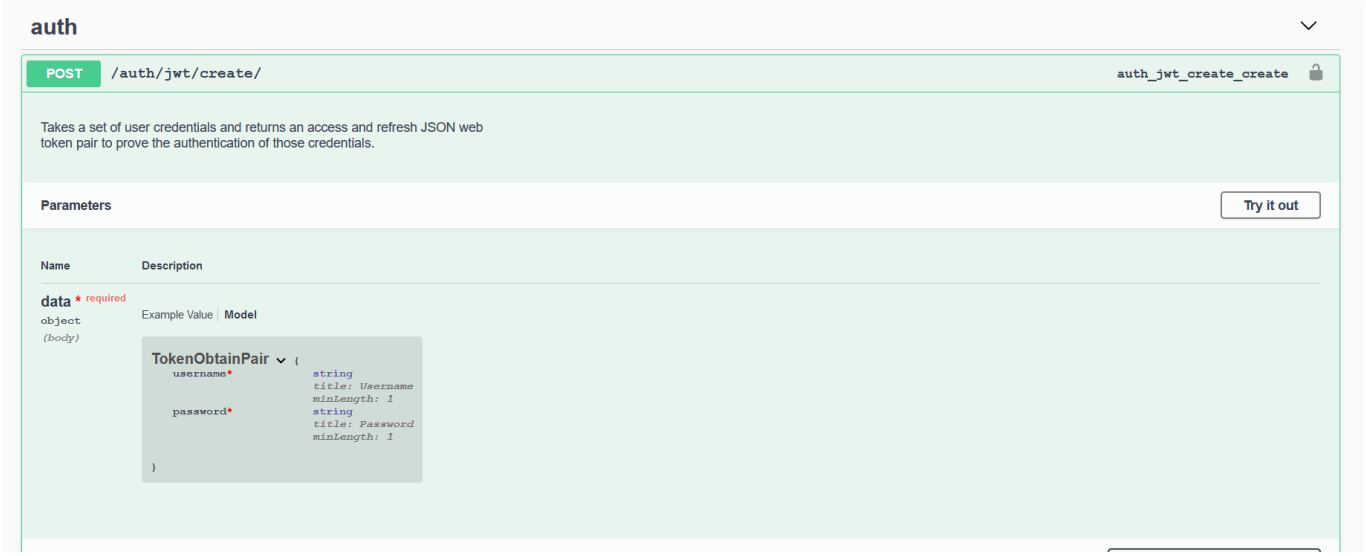

Рисунок 4.5 – Структура запиту для отримання токену доступу

В разі якщо запит був успішним у відповіді повернеться токен доступу термін життя якого складає 86400 секунд тобто один день, а також токен для поновлення токену доступу в якого термін життя складає 172800 секунд що дорівнює двом дням. У разі якщо термін життя токену доступу сплив, а токен для поновлення ще дійсний

то буде відправлено запит на поновлення. Таким чином не потрібні постійно турбувати користувача і змушувати його авторизуватися.

На рисунку 4.6 зображено структуру запиту для поновлення токену доступу, він приймає один єдиний параметр такий як токен для поновлення токену доступу і в разі якщо він дійсний повертає новий токен доступу та новий токен для поновлення.

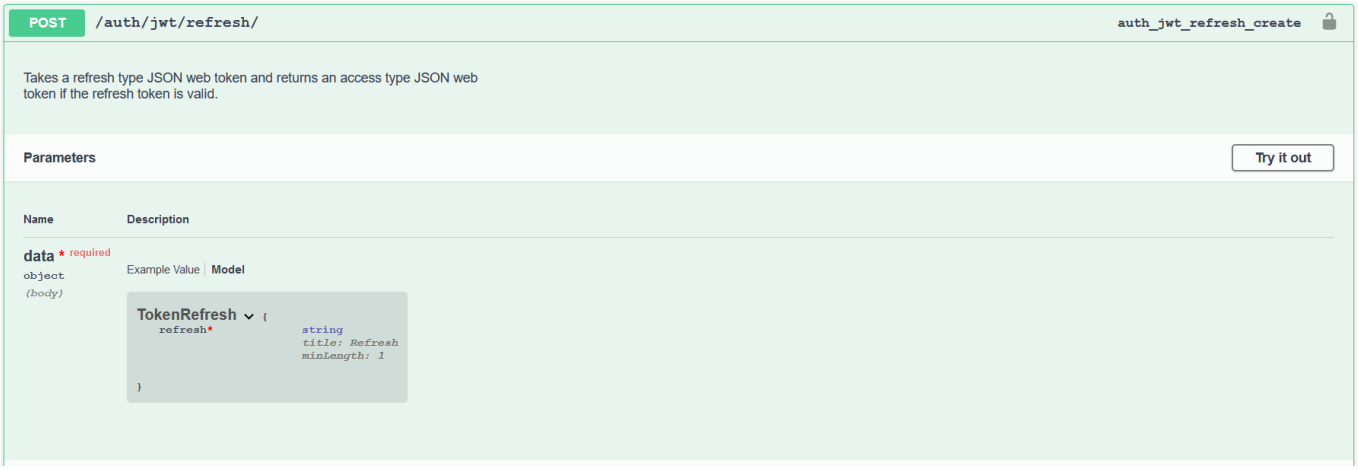

Рисунок 4.6 – Структура запиту для поновлення токену доступу

На рисунку 4.7 зображено структуру запиту для перевірки дійсності токену доступу. Запит приймає один єдиний параметр такий як токен доступу і після перевірки повертає статус 200 у випадку якщо токен дійсний, та статус 401 у випадку якщо термін життя токену доступу сплив або він не коректний.

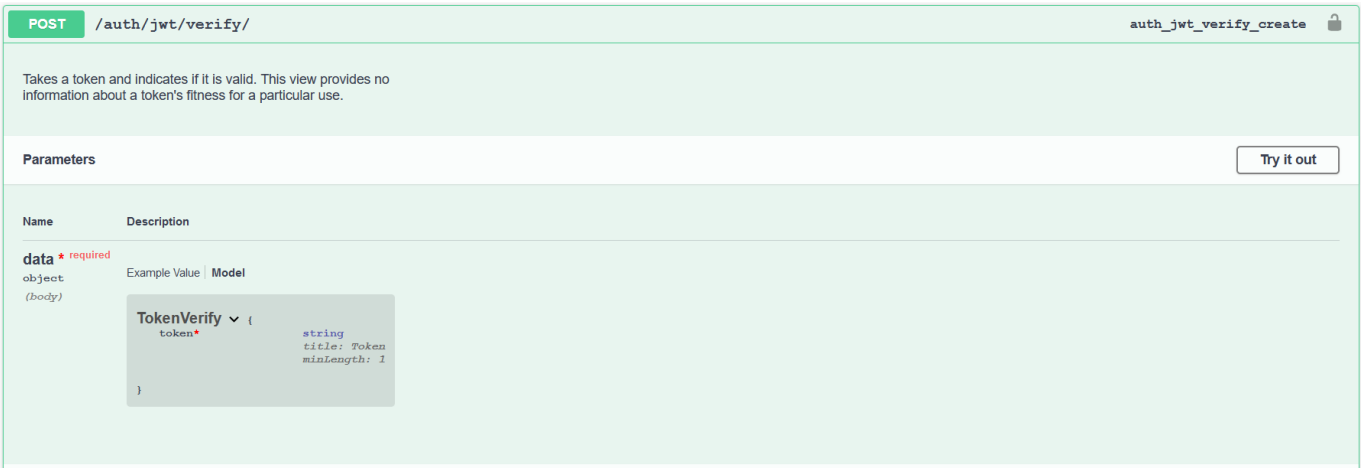

Рисунок 4.6 – Структура запиту для перевірки дійсності токену доступу

На рисунку 4.7 зображено схему запиту для реєстрації нового користувача. Обов’язковими параметрами є логін користувача, а також пароль. Імейл є не обов’язковим і за бажання його можна не вказувати.

Рисунок 4.7 – Структура запиту для реєстрації нового користувача

Поля в запиті перевіряються за наступними принципами:

— імейл повинен слідувати правилам викладеним в стандарті RFC 2822;

— логін повинен містити лише букви, цифри та знаки +, —, ., та _ також він не може бути коротшим ніж один символ та довшим ніж 254 символи;

—пароль не повинен бути коротшим ніж 8 символів, не може бути схожим на логін користувача, а також повинен містити як мінімум одну цифру та одну літеру в верхньому регістрі.

При вході в систему відбувається наступний ряд дій:

— У випадку якщо в локальному сховищі є токен доступу, тоді відправляється запит на перевірку його дійсності. Якщо токен дійсний користувача перенаправляють на головну сторінку, в противному випадку користувача направлять на сторінку авторизації.

— У випадку якщо термін життя токену доступу сплив, але токен для поновлення ще дійний тоді відправляється запит на поновлення і у випадку якщо він успішний в локальному сховищі обновляються токен доступу та токен для

поновлення і користувача перенаправляють на головну сторінку. В противному випадку користувача направляють на сторінку для авторизації.

— У випадку коли в локальному сховищі порожньо користувача відразу направляють на сторінку для авторизації.

В разі якщо в користувача немає облікового запису він може перейти зі сторінки авторизації на сторінку реєстрації. При реєстрації виконується наступний ряд дій:

— користувач заповнює всі необхідні поля;

— на стороні користувача проходить перевірка полів;

— у випадку якщо запит пройшов успішно відправляється ще один запит на отримання токену доступу після чого токен зберігається в локальному сховищі а користувача направляють на головну сторінку;

— у випадку коли запит не пройшов з’являється модальне вікно в якому міститься інформація про причину за якою обліковий запит не може бути створений з поточними параметрами. Після чого користувач може закрити вікно кліком на порожній простір та спробувати знову.

У разі якщо користувач хоче вийти з системи він може натиснути на кнопку Logout, що знаходиться у випадаючому меню розташованому зверху зправа на навігаційній панелі, після чого його буде направлено на сторінку для авторизації де він може увійти в інший обліковий запис або в той же самий, або створити новий.

На рисунку 4.8 зображено схему запиту для створення нового проекту. Запит приймає такі обов’язкові параметри як:

— Назва представляє собою рядок що складається з набору великих та малих латинських літер, а також чисел та знаків пробілу. Назва не повинна бути пустою та бути довшою ніж 150 символів. Також назва не є унікальним полем тому можна створити декілька проектів з однаковою назвою, але очевидно що так робити не слід, щоб не вносити плутаниці.

— Ідентифікатор користувача представляє собою цілочисельне поле яке зберігається в локальному сховищі і отримується при старті сесії шляхом запиту на ендпоінт /auth/users/me/.

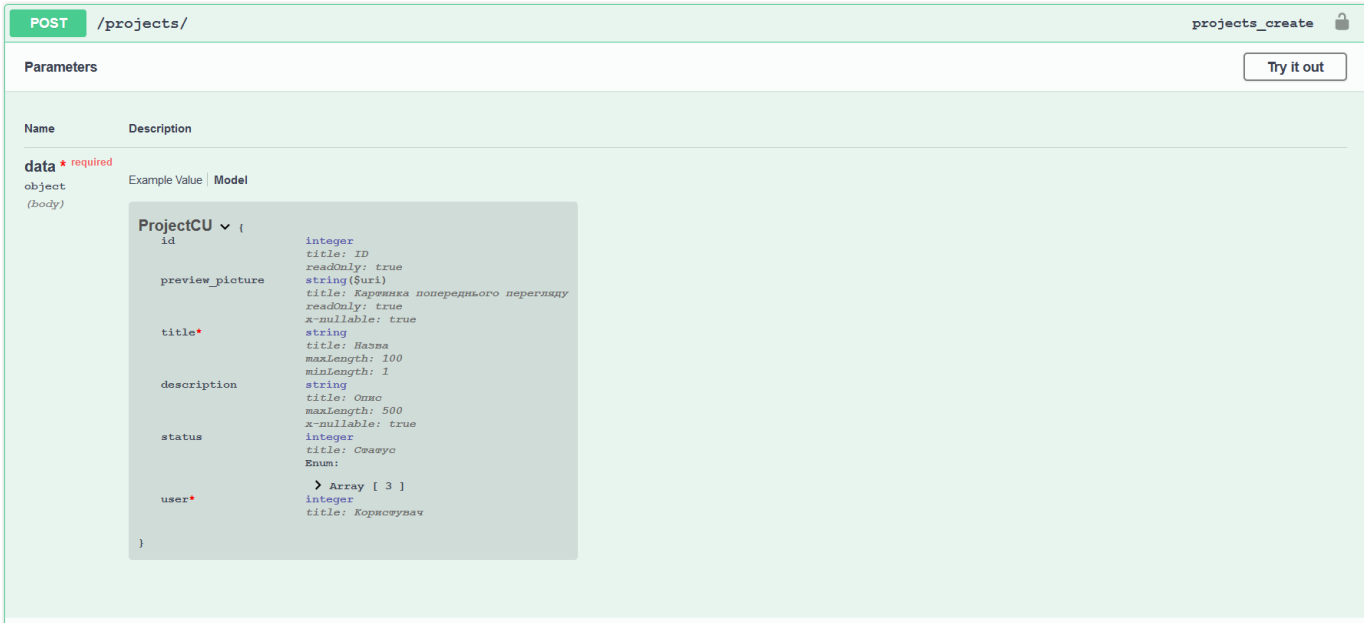

Рисунок 4.8 – Структура запиту створення нового проекту

Також при створенні проекту можна передати деякі необов'язкові такі як:

— Зображення для попереднього перегляду представляє собою файл з картинкою будь—якого формату який в подальшому відображається на картці проекту зліва. У випадку коли зображення не було передано замість нього буде відображатись альтернативний запис Preview picture тому бажано завжди передавати зображення щоб зберігати естетичний вигляд.

— Опис представляє собою рядок що містить стислу інформацію про проект що може бути корисним в деяких випадках.

Для того щоб отримати список всіх доступних для перегляду проектів необхідно відправити запит на ендпоінт GET /projects/ при цьому необхідно передати токен доступу. Після успішного запиту сервер поверне відповідь (рисунок 4.9) яка буде містити список з елементів які містять наступні поля:

— Числовий ідентифікатор проекту за допомогою якого адміністратор при необхідності може відправити запит на видалення проекту, також ідентифікатор потрібен для того, щоб перейти на сторінку для перегляду задач які відносяться до цього проекту.

— Кількість задач які відносяться до цього проекту.

— Кількість задач які відносяться до цього проекту та були успішно виконані користувачами.

— Посилання на зображення для попереднього перегляду.

— Назва проекту.

— Опис проекту.

— Статус проекту.

— Дата створення проекту.

— Дата останньої зміни в процесі виконання проекту.

— Ідентифікатор користувача за яким можна встановити хто саме з адміністраторів створив проект.

Рисунок 4.9 – Структура відповіді на отримання проектів

Для того щоб отримати список задач існує в системі передбачено два варіанти, а саме отримання списку всіх доступних задач та отримання задач які належать до одного конкретного проекту. Для того щоб отримати всі задачі необхідно відправити запит на ендпоінт GET /tasks/ при цьому в параметрах слід передати токен доступу. Після успішного запиту сервер поверне відповідь як на рисунку 4.10.

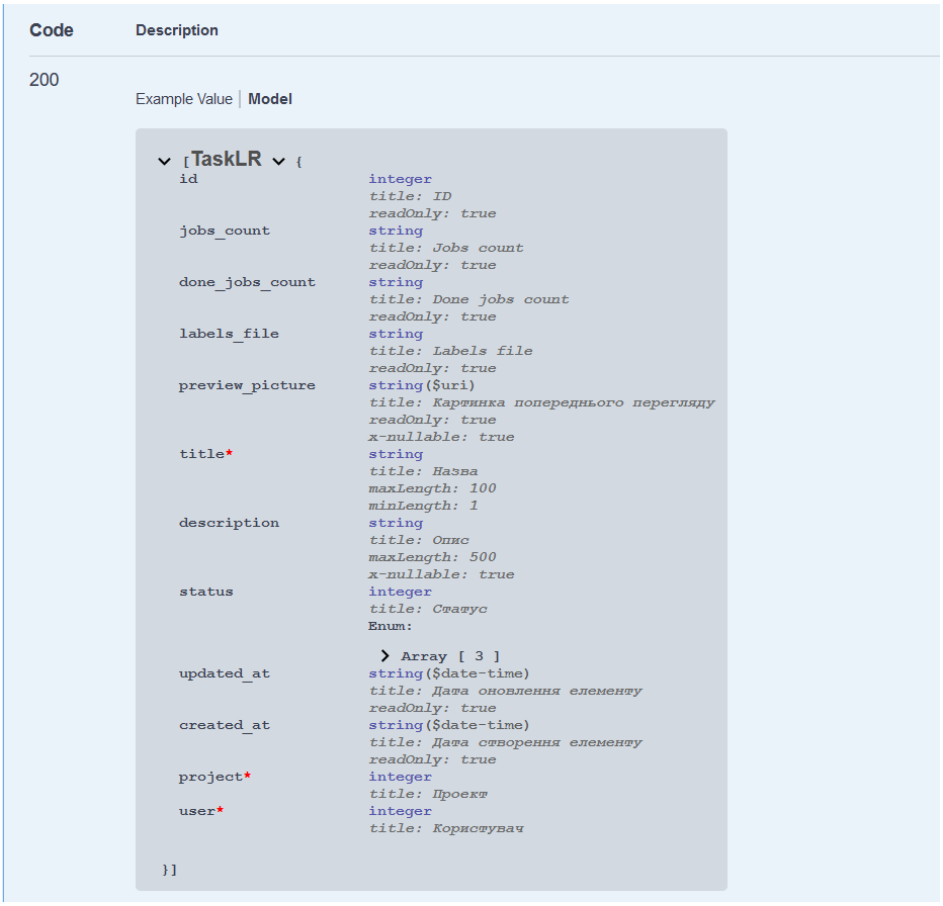

Рисунок 4.10 – Структура відповіді на отримання проектів

Відповідь містить в собі список з елементів кожен з яких представляє собою одну з сутностей задачі в базі даних застосунку а також деяку додаткову інформацію яка виникає в ході взаємодії користувачів з системою. Кожен елемент складається з наступних полів:

— Числовий ідентифікатор задачі за допомогою якого адміністратор при необхідності може відправити запит на видалення задачі, також ідентифікатор потрібен для того, щоб перейти на сторінку для перегляду робіт які відносяться до цієї задачі.

— Кількість робіт які відносяться до цього проекту.

— Кількість робіт які відносяться до цієї задачі та були успішно виконані користувачами.

— Посилання на зображення для попереднього перегляду.

— Посилання на файл в якому містяться теги до всіх зображень які належать до робіт які входять до складу даної задачі і були відмічені користувачами в процесі їх роботи.

— Числовий ідентифікатор проекту до якого належить дана задача.

— Назва задачі.

— Опис задачі.

— Статус задачі.

— Дата створення задачі.

— Дата останньої зміни в процесі виконання задачі.

— Ідентифікатор користувача за яким можна встановити хто саме з адміністраторів створив задачу.

Для того щоб отримати список всіх задач які належать до даного проекту необхідно відправити запит на ендпоінт GET /projects/{id}/ при цьому необхідно в параметрах відправити токен доступу, а в змінній шляху вказати числовий ідентифікатор проекту, задачі якого ви хочете отримати.

Для того щоб отримати список всіх робіт які належать до даної задачі необхідно відправити запит на ендпоінт GET /tasks/{id}/ в параметрах запиту необхідно передати токен доступу, а в змінній шляху потрібно вказати числовий ідентифікатор задачі для якої ви хочете отримати список робіт. У випадку якщо запит був успішний сервер поверне відповідь яка буде містити список задач як на рисунку 4.11. Кожен елемент у списку містить наступні поля:

— Числовий ідентифікатор роботи за допомогою якого адміністратор може відправити запит на видалення роботи, також ідентифікатор потрібен для того, щоб перейти на сторінку для виконання роботи.

— Кількість зображень з яких складається робота.

— Кількість зображень які відносяться до цієї роботи та були успішно розмічені користувачами.

— Посилання на зображення для попереднього перегляду.

— Назва роботи.

— Опис роботи.

— Статус роботи.

— Дата створення роботи.

— Дата останньої зміни в процесі виконання роботи.

— Ідентифікатор користувача за яким можна встановити хто саме з адміністраторів створив роботи.

Рисунок 4.11 – Структура відповіді на отримання робіт

Для того щоб отримати список зображень які входять до складу певної роботи необхідно відправити запит на ендпоінт GET /jobs/{id}/ при цьому в параметрах необхідно вказати токен доступу, а в змінній шляху записати числовий ідентифікатор роботи для якої ви хочете отримати список зображень які до неї входять. Якщо запит

був успішним з серверу повернеться відповідь (рисунок 4.12) у якій буде міститись список з сутностей які відповідають зображенням в базі застосунку.

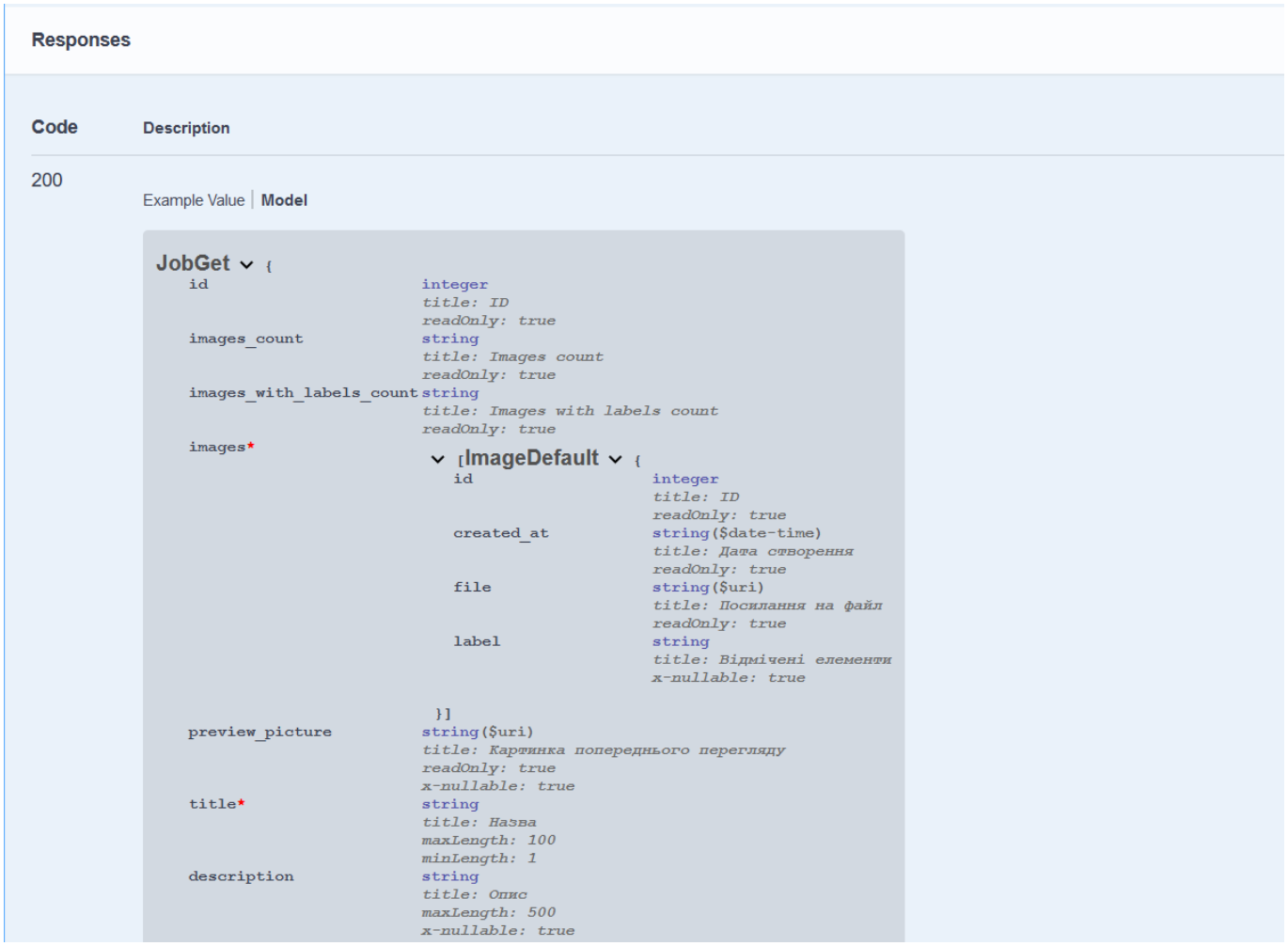

Рисунок 4.11 – Структура відповіді на зображень в роботі

Кожне зображення містить наступний набір полів:

— Унікальний числовий ідентифікатор зображення за допомогою якого можна відправляти запити для дій на зображенням.

— Дата створення відповідає даті створення роботи або даті коли зображення було додане до роботи шляхом її модифікації.

—Посилання на зображення за допомогою якого користувач може виконувати роботи, такий підхід дозволяє зменшити навантаження на сервер за рахунок того що не потрібно відразу передавати всі зображення, а також дозволяє користувачам не чекати довгий відрізок часу, а приступити до виконання роботи після завантаження

першого зображення. Ще одним з плюсів є те, що після того як зображення було завантажено воно залишається в оперативній пам’яті комп’ютера, тому між завантаженими зображеннями можна переходити миттєво.

— Теги представляють собою поле типу json в якому міститься вся інформація про об’єкти які відмітив користувач на зображенні.

Кожен об’єкт повинен містити наступний набір полів які характеризують його положення відносно зображення не залежно від того де знаходиться саме зображення з урахуванням зміни лінійних розмірів зображення викликаними в процесі роботи з ним. Також об’єкти містять інформацію про свій тип, а також що на них зображено при умові якщо користувач це зробив. Тому було вирішено створити такі поля:

— положення верхньої лівої точки за координатою X;

— положення верхньої лівої точки за координатою Y;

— положення нижньої лівої точки за координатою X у випадку якщо це область;

— положення нижньої лівої точки за координатою Y у випадку якщо це область;

— тип об’єкту;

— контекст якому відповідає об’єкт якщо він вказаний.

5. РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

Програма не потребує жодної інсталяції так як система розгорнута в хмарному сервісі GCP (Google Cloud Platform), тому все що потрібно користувачу це:

— персональний комп’ютер;

— один з популярних браузерів;

— доступ до мережі інтернет.

5.1 Опис інтерфейсу сторінки з проектами

Після успішної авторизації всі користувачі потрапляють на сторінку з проектами (рисунок 5.1)

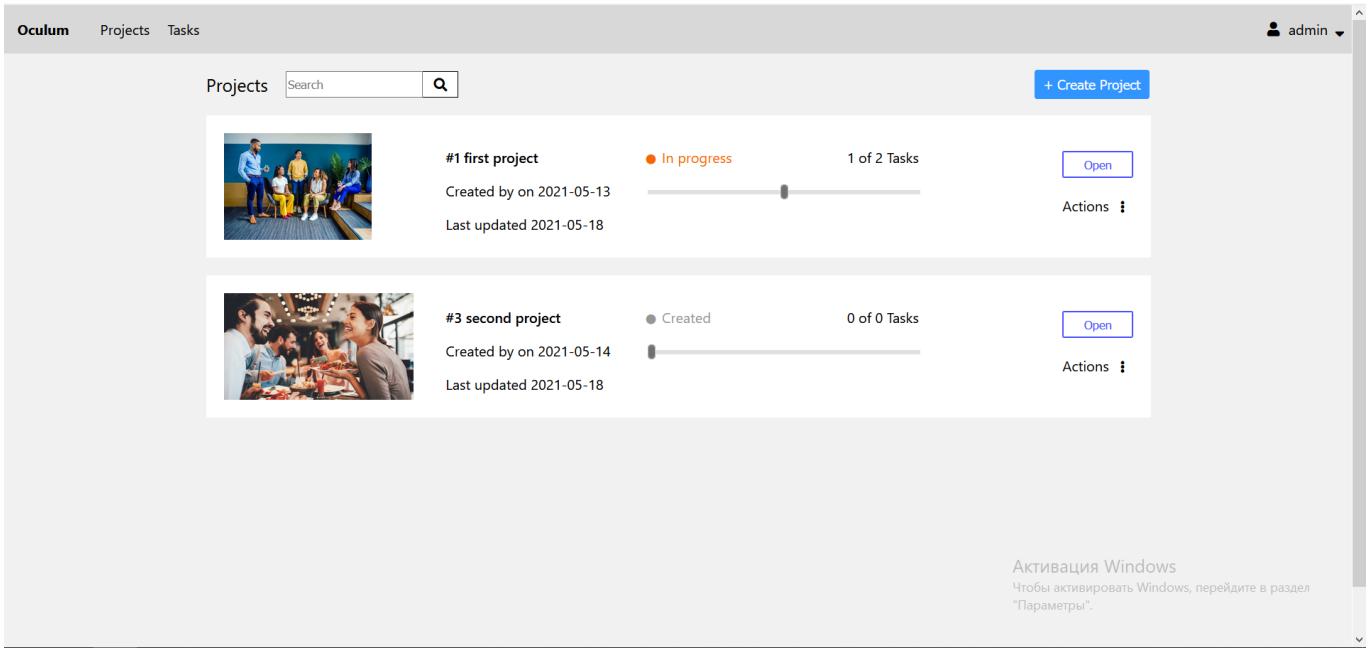

Рисунок 5.1 – Сторінка з проектами

Сторінка складається з наступних елементів:

— панель навігації;

— форма для пошуку проектів за іменем;

— кнопка створення нового проекту;

— картки з проектами.

Панель навігації містить в собі:

— посилання на сторінку з проектами, при цьому відображаються всі існуючі проекти;

— посилання на сторінку з проектами, при цьому відображаються всі проекти;

— ім’я користувача;

— кнопка яка відкриває випадаючий список з додатковими діями (рисунок 5.2).

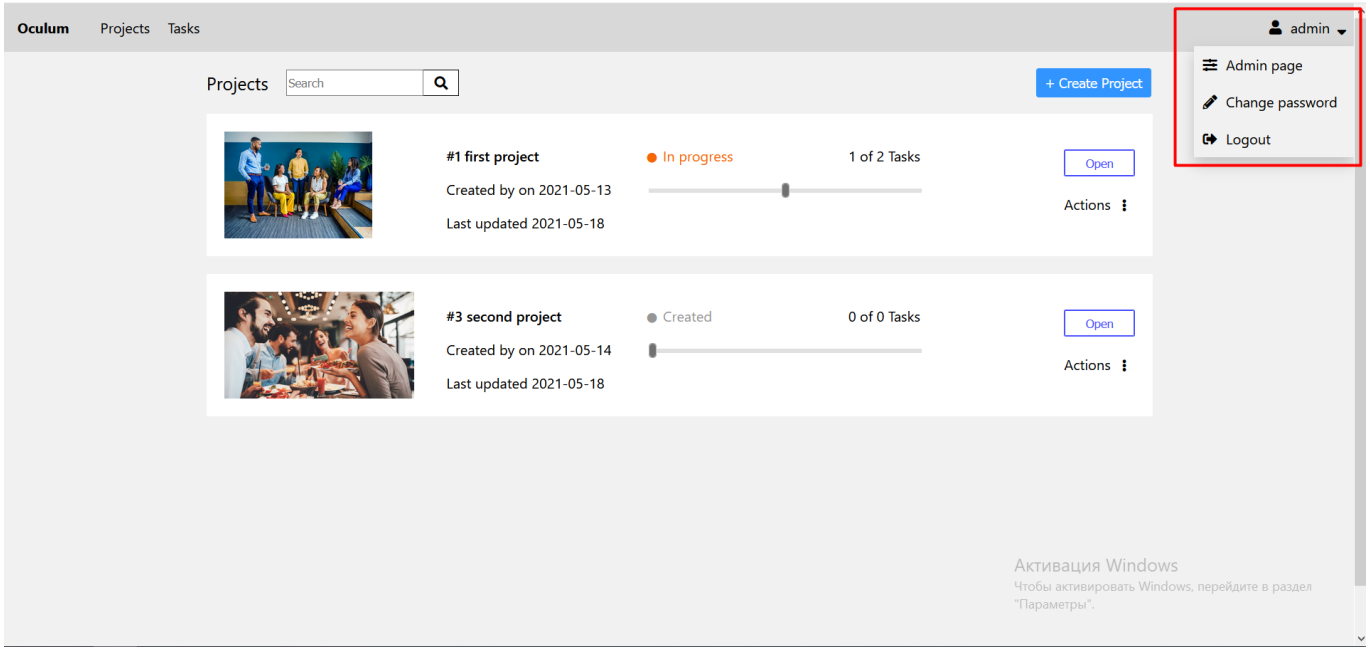

Рисунок 5.2 – Положення кнопки випадаючого списку

У випадаючому списку наявні 3 дії, а саме:

— посилання для переходу в адмінську панель Django;

— посилання на сторінку зміну паролю;

— кнопка виходу.

Форма для пошуку проектів за іменем шукає наявність заданого рядка в іменах усіх проектів і виводить усі відповідні варіанти

Картка з проектом містить в собі наступні елементи:

— зображення для попереднього перегляду;

— індекс проекту в базі даних;

— ім’я проекту;

— юзернейм користувача який створив цей проект;

— дата створення проекту;

— дата останньої зміни в стані проекту;

— стан проекту;

— кількість повністю виконаних задач в проекті;

— шкала прогресу яка відображає яка частина задач в проекті була завершена;

— кнопка для переходу на сторінку з задачами, при цьому будуть показань лише ті задачі які належать до даного проекту;

— кнопка яка відкриває випадаючий список з додатковими діями (рисунок 5.3)

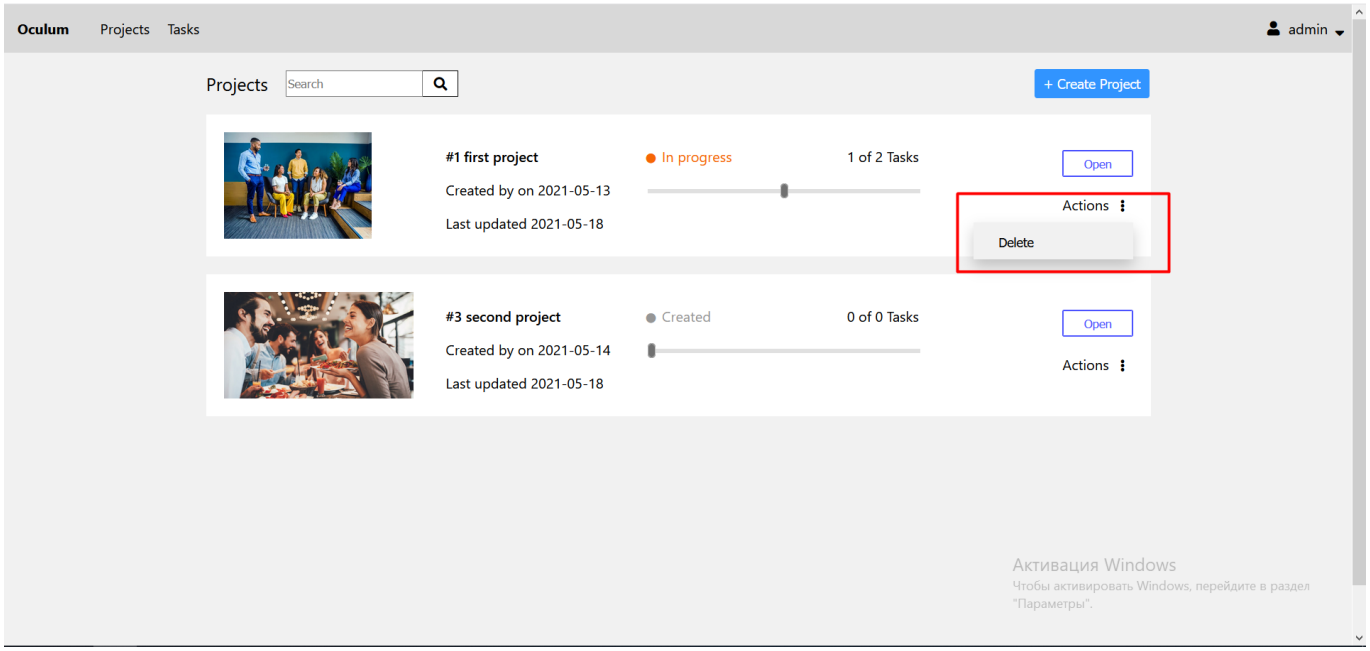

Рисунок 5.3 – Положення випадаючого списку з додатковими діями

При натисканні на кнопку створити проект відбувається перехід на сторінку для створення проекту. Слід зауважити що цю дію може виконати виключно користувач який має повноваження адміністратора

5.2 Опис інтерфейсу сторінки для створення проектів

На сторінці для створення проектів (рисунок 5.4) наявні наступні поля для вводу інформації:

— поле для вводу назви проекту;

— поле для вводу опису для проекту;

— кнопка для відкриття стандартного діалогу завантаження файлу з пристрою користувача для того щоб додати зображення для попереднього перегляду;

— кнопка для відправки запиту на створення проекту.

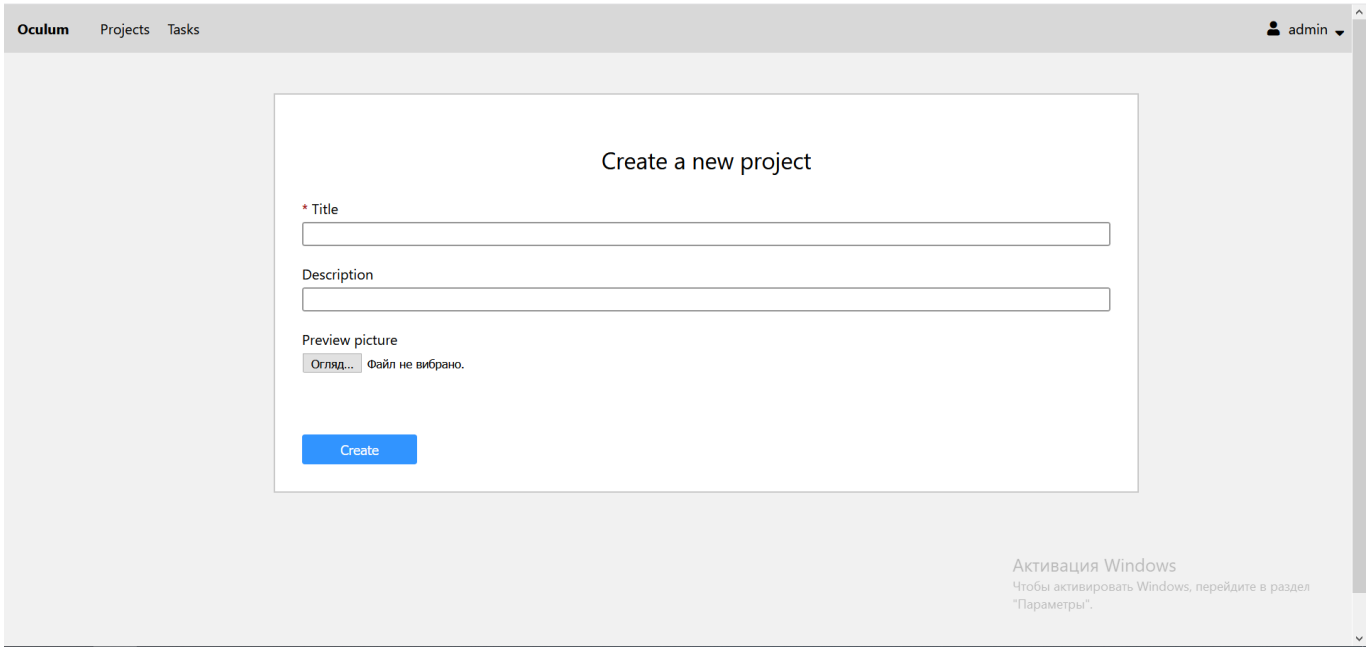

Рисунок 5.4 – сторінка для створення нового проекту

У разі того як запит на творення проекту був успішним користувача буде направлено на сторінку перегляду проектів де він відразу зможе побачити щойно створений проект. У разі якщо введені параметри порушують якісь обмеження, або бекенд знаходиться в не робочому стані з’явиться модальне вікно з поясненнями про причину невдалого запиту.

5.3 Опис інтерфейсу сторінки для виконання робіт

При натисканні на кнопку для відкриття роботи відбувається перехід на сторінку для виконання роботи (рисунок 5.5). На цій сторінці користувач повинен відмічати зони з обличчями людей, а також вказувати на них ключові точки при цьому вказуючи яку емоцію відчуває людина та якій зоні відповідає ключова точка.

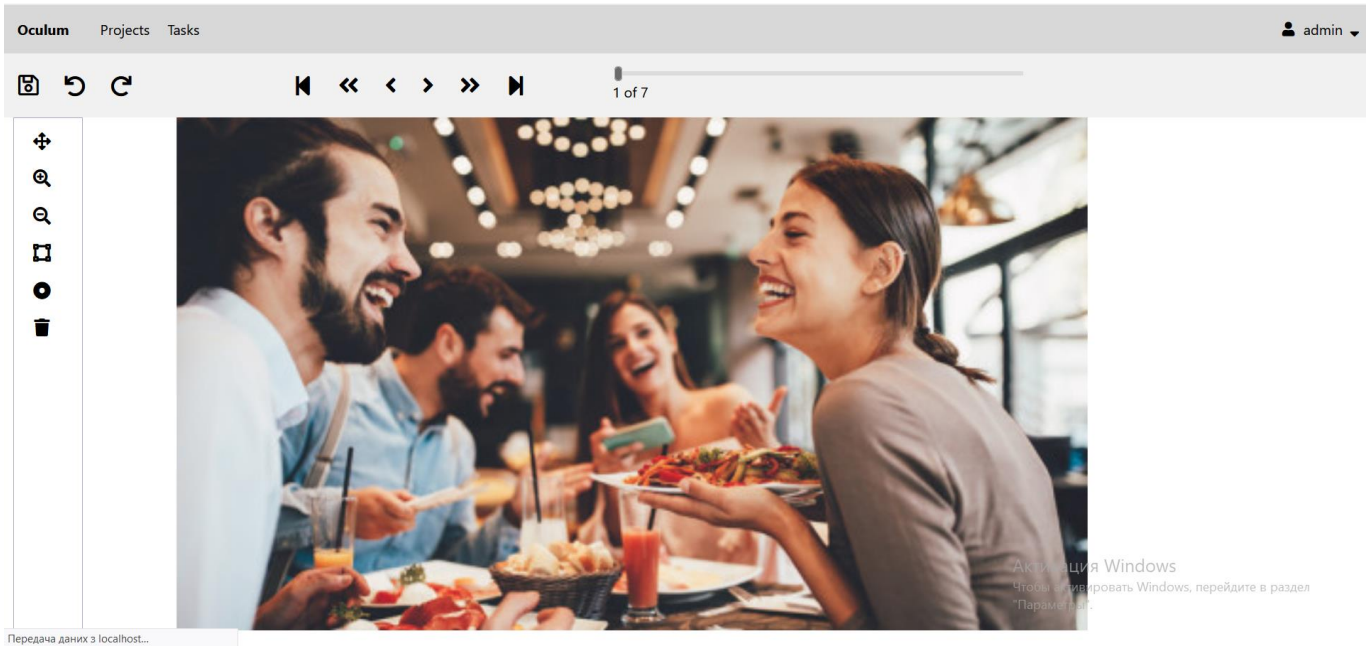

Рисунок 5.5 – сторінка для виконання роботи

На сторінці знаходяться декілька панелей з кнопками та зона для розмітки зображення. На сторінці наявні наступні елементи:

— кнопка для збереження відмічених об’єктів на зображенні;

— кнопка для відміни попередньої дії;

— кнопка для повернення відміненої дії;

— кнопка переходу до першого зображення;

— кнопка переходу до останнього зображення;

— кнопка повернення на 5 зображень;

— кнопка переходу на 5 зображень вперед;

— кнопка переходу до наступного зображення;

— кнопка переходу до попереднього зображення;

— кнопка для ввімкнення можливості рухати об’єкти;

— кнопка для збільшення зображення;

— кнопка для зменшення зображення;

— кнопка ввімкнення можливості розмічання області з обличчям;

— кнопка для ввімкнення можливості розмічання ключових точок;

— кнопка для видалення виділеного об’єкта;

— шкала прогресу.

5.4 Опис головної сторінки панелі адміністратора

Всі адміністратори системи можуть потрапити до адміністраторської панелі Django (рисунок 5.6) за допомогою пункту у випадаючому меню що знаходиться в навігаційній панелі.

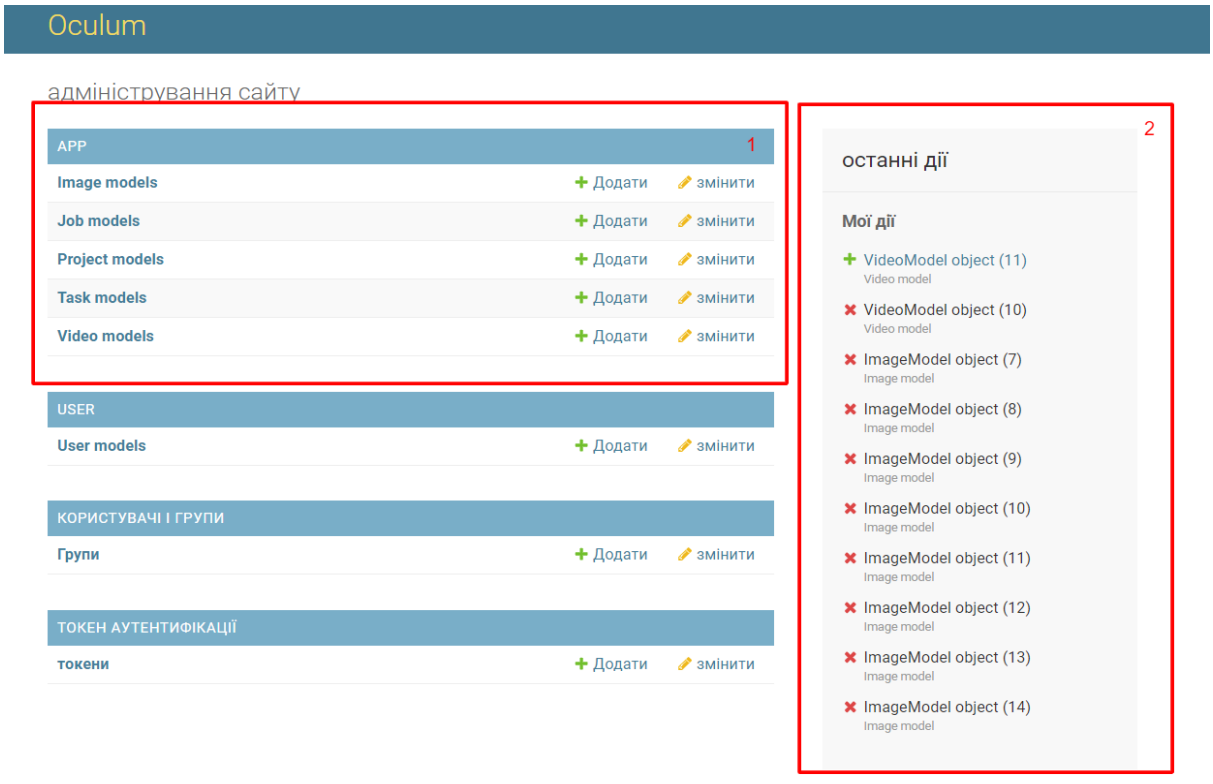

Рисунок 5.6 — Адміністративна панель Oculum

На сторінці можна переглянути останні дії в панелі, а також оглянути недавно змінені, та створені сутності. Якщо натиснути на кнопку з назвою моделі, або на посилання для зміни моделі буде відкрито сторінку з сутностями відповідної моделі. При взаємодії з кнопкою створити, буде відкрита сторінка для створення вибраної сутності. Також є функціонал для управління персональними даними користувачів або, а також групами користувачів.

5.5 Опис сторінки для управління проектами

На рисунку 5.7 зображено сторінку для управлінням головною сутністю в організації роботи користувачів такою як проект. На зображенні червоними областями виділено ключові області для взаємодії які будуть описані нижче в тексті.

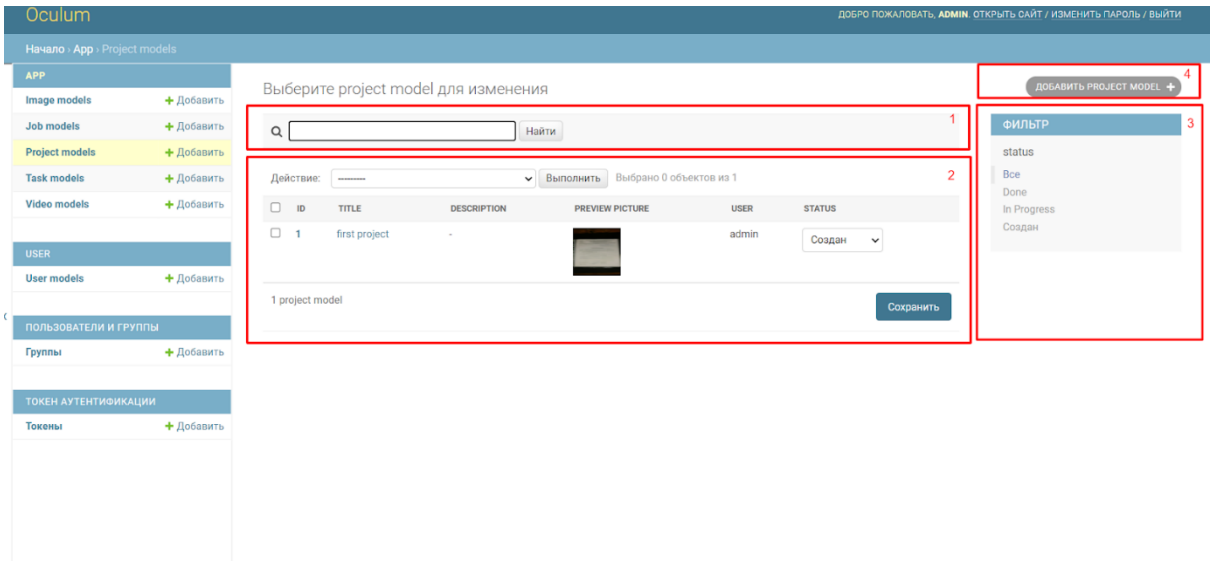

Рисунок 5.7 — Список проектів

В області під номером два зображено список всіх існуючих проектів з їхніми полями. Адміністратор може вибрати проект який він хоче змінити, або виділити один або декілька проектів які він хоче видалити та натиснути відповідну кнопку.

В області під номером три представлено набір фільтрів за якими можна здійснювати пошук серед проектів. У випадку якщо фільтрів було недостатньо можна скористатись пошуком за назвою, зображеним в області під номером один.

При натисканні на кнопку створити проект, буде виконано перехід на сторінку для його створення. Якщо натиснути на ідентифікатор проекту, або на його назву, то буде здійснено перехід на сторінку для його редагування (рисунок 5.8).

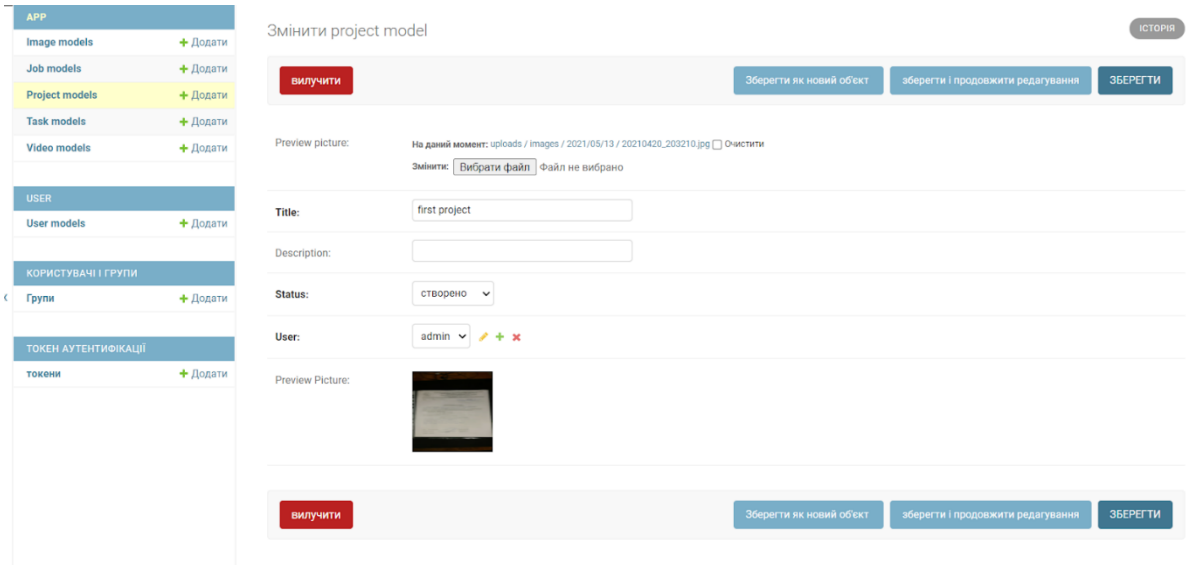

Рисунок 5.8 — Форма редагування проекту

На рисунку 5.9 показано список всіх полів для створення проекту, та зображення для попереднього перегляду.

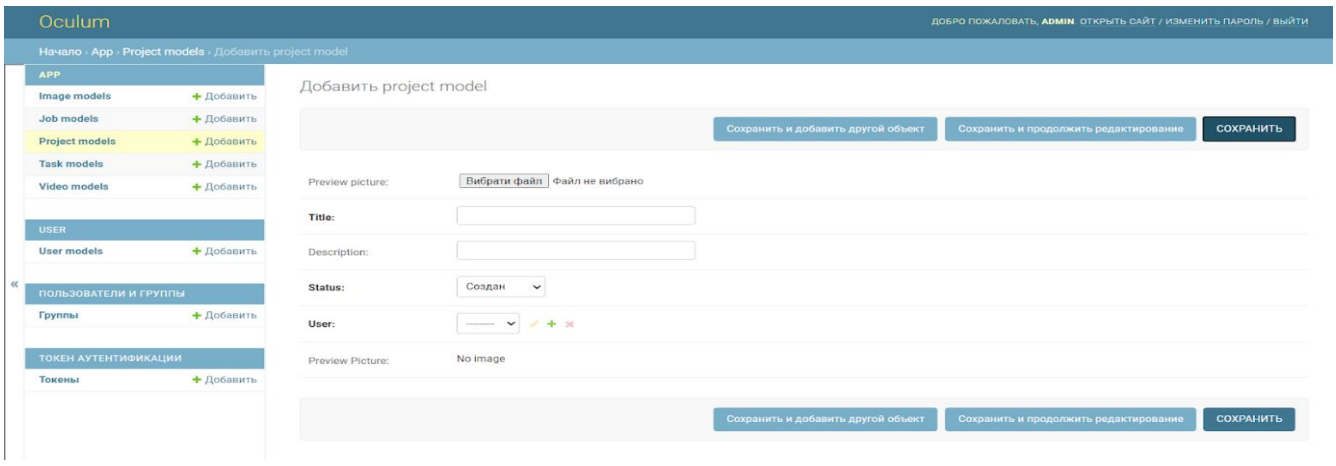

Рисунок 5.9 — Форма створення проекту

Всі поля можна вказувати та редагувати, після чого елемент можна зберегти, натиснувши кнопку “Зберегти”, після цього відбудеться перехід на сторінку зі списком проектів. Також можна створити та залишитись на сторінці для редагування або видалити сутність та створити з неї новий об’єкт. Останнє пункт дає можливість швидко створювати багато сутностей, але при цьому необхідно вказувати унікальну назву для проекту, в противному випадку буде згенеровано попередження, про це.

5.6 Опис сторінки для управління задачами

На рисунку 5.10 зображено інтерфейс сторінки для управління такими сутностями як задачі. Тут адміністратори можуть виконувати дії для редагування а також видалення задач у разі якщо це потрібно, а також можна відкрити сторінку для створення нової задачі.

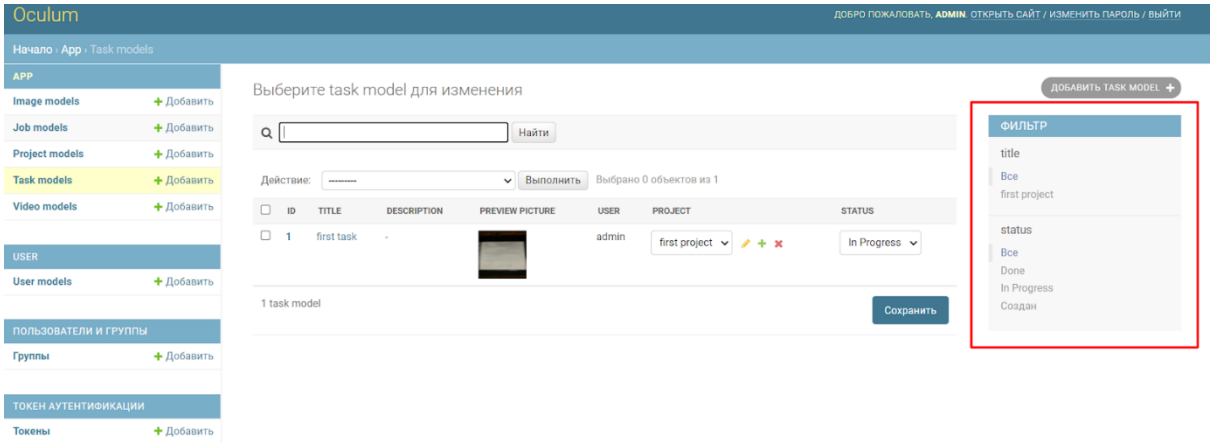

Рисунок 5.10 — Список завдань

На рисунку 5.11 зображено інтерфейс сторінки для редагування задач, тут можна подібно до сторінки редагування проектів вказувати такі поля як: назва, опис, статус та картинку для попереднього перегляду.

На відміну від створення проектів при створенні задачі необхідно вказувати до якого проекту вона належить

При редагуванні задач так само як і при редагуванні проектів можна відразу після збереження перейти на сторінку з задачами, або залишитись і редагувати ще

одну задачу. Весь цей функціонал забезпечується стандартними засобами фреймворку Django.

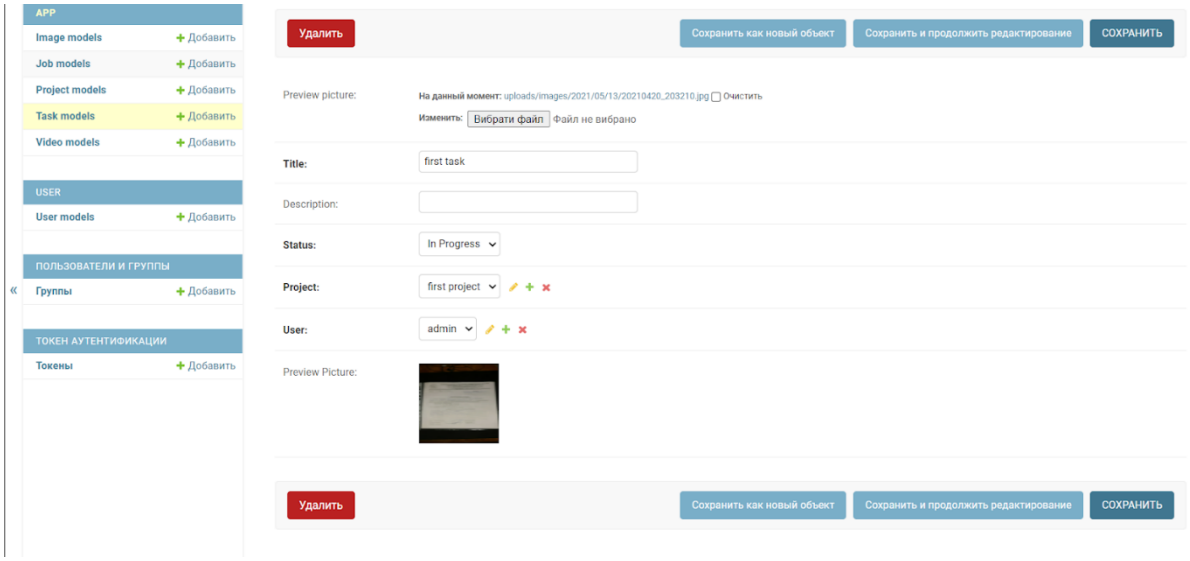

Рисунок 5.11 — Форма редагування завдання

На рисунку 5.12 зображено інтерфейс сторінки для створення нової задачі з усіма необхідними при цьому полями. За допомогою неї можна створити сутності середньої ланки такі як задачі.

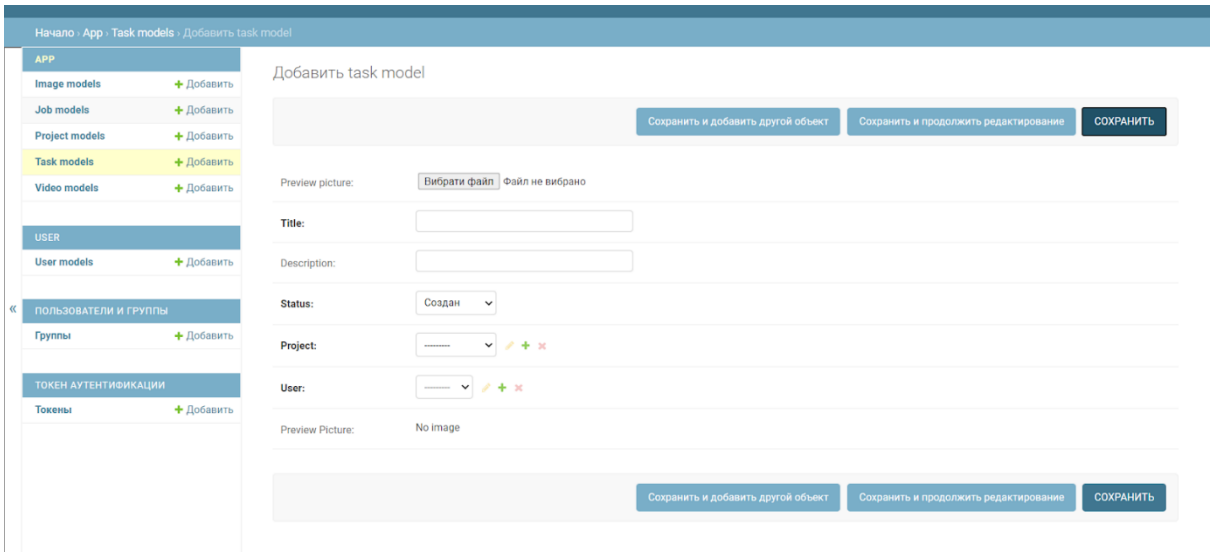

Рисунок 5.12 — Форма створення завдання

Функціонал сторінки для створення задач подібний до сторінки створення проектів. Сторінка для редагування (рисунок 5.11) подібна до сторінки для створення

задач (рисунок 5.12). Але на сторінці для редагування є можливість пошуку за назвою проекту.

5.7 Опис сторінки для управління роботами

На рисунку 5.13 зображено інтерфейс сторінки призначений для редагування робіт. На відміну від сторінок для перегляду проектів та задач тут можна побачити хто саме з користувачів повинен виконувати роботу, а також хто з адміністраторів повинен її перевіряти

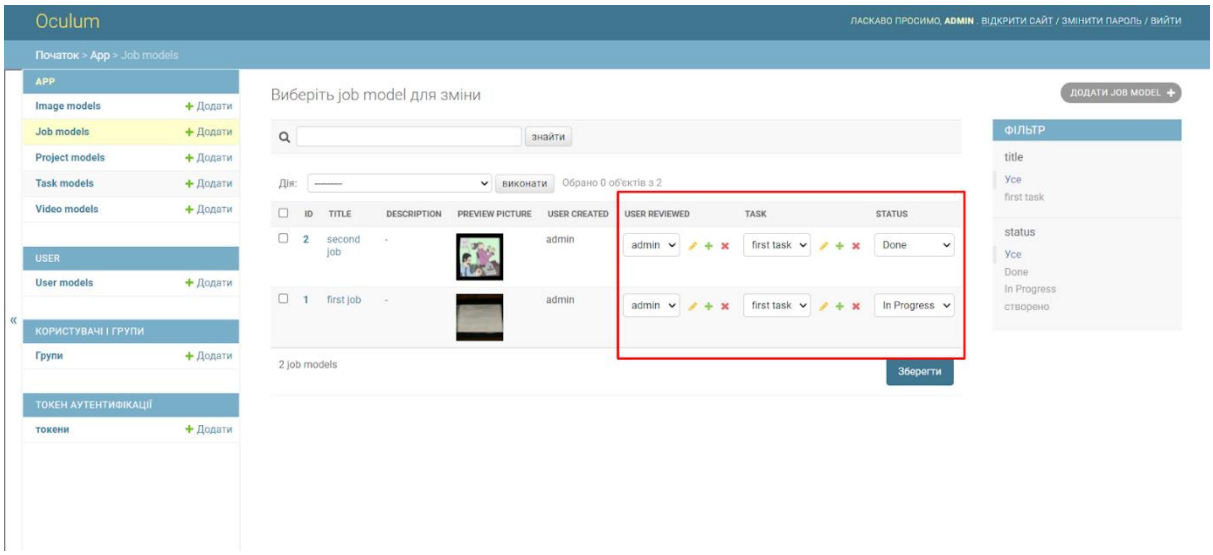

Рисунок 5.13 — Список робіт

На сторінці зі списком робіт (рисунок 5.13) на відміну від розглянутих вище сторінок призначених для редагування проектів та задач є можливість не відкриваючи сторінки для редагування сутності редагувати такі параметри як завдання до якого належить ця робота, а також користувач який повинен виконати дану роботу. Також зверху знаходиться поле для пошуку робіт за іменем.

На рисунку 5.14 зображено інтерфейс сторінки для редагування робіт, де можна редагувати всі поля на відміну від сторінки для перегляду серед яких: назва, статус, користувач який створив роботу, опис задачі, а також користувач який відповідальний за виконання задачі. Після того як адміністратор виконав всі

необхідні дії він може скористатися одним з варіантів збереження які були описані у пунктах про роботу з проектами та задачами.

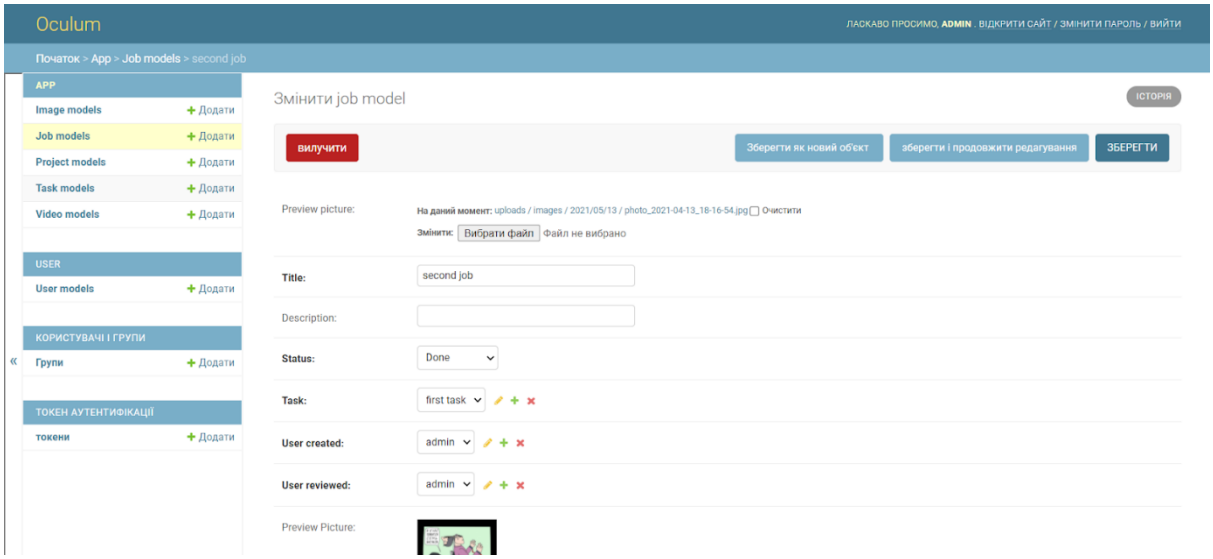

Рисунок 5.14 — Форма редагування роботи, частина 1

На рисунку 5.15 зображено другу частину інтерфейсу сторінки для редагування робіт. Окрім функціоналу описаного вище на цій сторінці є можливість: переглядати вже створені зображення, а також теги які відмітили користувачі в процесі роботи, є можливість змінювати будь—які зображення завантаживши новий файл, можна створити будь—яку кількість нових зображень, можна змінити вміст файлу з тегами, а також можна видалити за бажанням будь—яку кількість зображень з роботи.

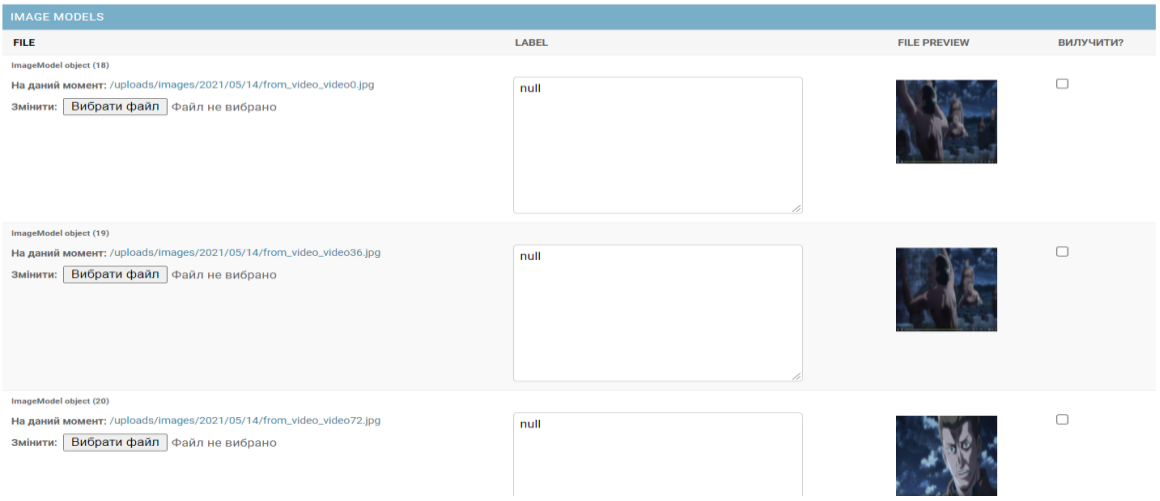

Рисунок 5.15 — Форма редагування роботи, частина 2

На рисунку 5.16 зображено інтерфейс сторінки для створення робіт. На сторінці є можливість завантажити набір картинок з яких повинна складатися робота Форми для зображень та поле з тегами не мають додаткових функцій.

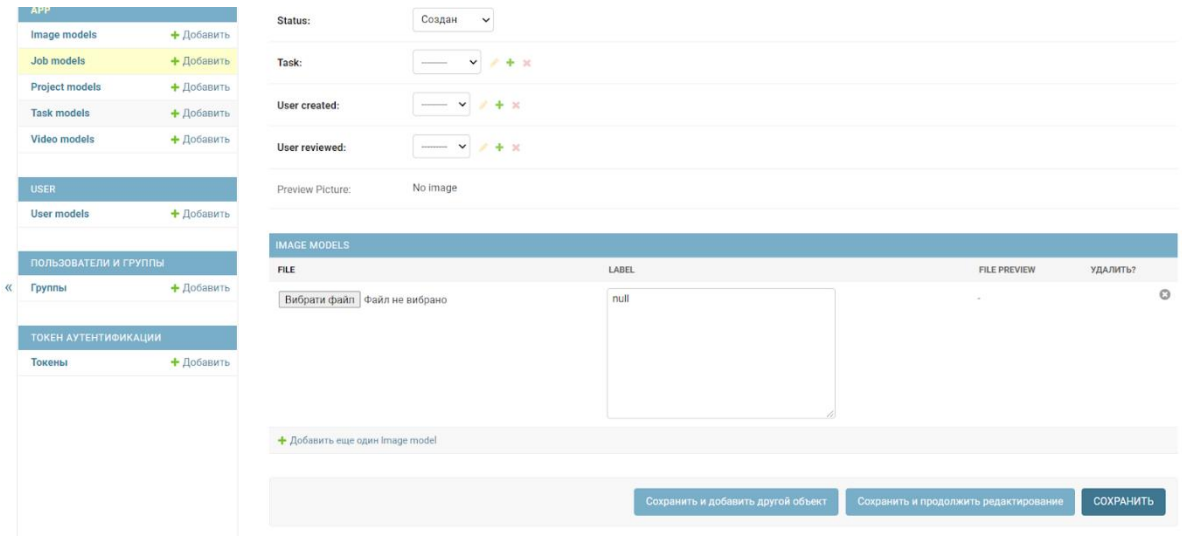

Рисунок 5.16 — Форма створення роботи.

На стороні серверу створено додаткові можливості, що спрощують роботу з застосунком:

— Якщо хоча б для одного з зображень в роботі було збережено теги. То зображення переходить в статус “Готово”, а робота отримує статус “В процесі”.

— У випадку якщо всі зображення які належать до роботи були розмічені, то вона переходить до статусу “Готово”, а завдання до якого належить ця робота до статусу — “В процесі”.

— Такі самі правила діють у відношенні до задач та проектів.

— Після завантаження будь-якого зображення до роботи, вона і всі батьківські елементи, отримають це зображення як картинку для попереднього перегляду, але у випадку, якщо це поле не було передано при створенні батьківських елементів.

6. ВИПРОБУВАННЯ РОЗРОБЛЕНИХ ПРОГРАМНИХ ЗАСОБІВ

В ході випробування коректності роботи системи було перевірено ряд таких функцій:

— створення проекту;

— створення задачі;

— створення роботи;

— видалення проекту;

— видалення задачі;

— видалення роботи;

— пошук за назвою серед проектів задач та робіт;

— підтримка різних форматів зображень при використанні їх в якості зображень для попереднього перегляду та в ролі змісту для виконання роботи;

— завантаження тегованого датасету для завдання;

— коректність зміни стану виконання проектів, задач та робіт після завершення частини роботи на будь якому рівні ієрархії;

— динамічне обновлення списків проектів задач та робіт після видалення з них елементів;

— направлення користувача на попередню сторінку після успішного створення проектів задач та робіт;

— відображення помилок у разі якщо виникли якісь неполадки при відправленні запитів;

— коректність роботи авторизації та реєстрації;

— коректність координат при створенні тегів;

— переходи між зображеннями в ході виконання робіт;

— можливість переміщувати в графічному редакторі такі об’єкти як області та ключові точки;

— можливість виділяти області в графічному редакторі;

— можливість виставляти ключові точки в графічному редакторі;

— можливість зміни розмірів виділеної області в графічному редакторі;

— коректність переміщення всіх об’єктів при умові що було виділено зображення

— видалення областей та ключових точок;

— вказання емоцій на виділених областях, а також вказання зони обличчя до якої належить ключова точка;

— коректність переміщення об’єктів при зміні розмірів зображення;

— коректність зміни виділених областей при зміні розмірів зображення;

— відображення вибраних емоцій та зон на обличчі при виборі виділених областей та ключових точок для яких попередньо були вказані ці параметри;

При перевірці всіх вище зазначених функцій не було виявлено помилок в роботі системи як на стороні фронтенду з яким працювали звичайні користувачі, так і на стороні бекенду та бази даних.

ВИСНОВКИ

У ході розробки системи Oculum було проведено наступний ряд дій:

— проаналізовано доступну літературу на базі практики яка відноситься до практичного застосування датасетів зображень в ході тренування моделей штучного інтелекту в такому розділі машинного навчання як computer vision;

— досліджено літературу пов’язану з обробкою цифрових зображень з метою визначення способів лінійних перетворень зображень, а також визначення алгоритмів переміщення точок на зображеннях в ході зміни їх лінійних розмірів та переміщення.

— досліджено та порівняно існуючі системи з аналогічним призначенням з метою визначення слабких та сильних сторін в роботі графічних редакторів для подальшого планування способів покращення існуючого функціоналу;

— розроблено модель роботи системи яка дозволить реалізувати всі ідеї отримані в ході аналізу доступних аналогів, а також дасть можливість організовувати паралельну роботу великої кількості користувачів;

— вибрано стек технологій який дозволить ефективно та швидко реалізувати поставлені цілі щодо створення графічного редактора який буде поєднувати в собі всі переваги доступних аналогів, а також можливості які виникли в ході аналізу застосування датасетів при визначенні емоцій моделями штучного інтелекту;

— створено графічний редактор який дозволяє швидко та зручно тегувати датасети зображень не залежно від того чи знайомий користувач з системою чи ні, а також від того чи має він профільну технічну освіту та навички програмування;

— розгорнуто систему в хмарному сервісі та протестовано коректність взаємодії між різними складовими елементами інфраструктури;

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Вступ до Django [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en—US/docs/Learn/Server—side/Django/Introduction.

2. Схема DRF [Електронний ресурс] – Режим доступу до ресурсу: https://www.django—rest—framework.org/api—guide/schemas/.

3. Що таке PostgreSQL, її плюси і мінуси [Електронний ресурс] – Режим доступу до ресурсу: https://oracle—patches.com/common/3214—%D1%87%D1%82%D0%BE—%D1%82%D0%B0%D0%BA%D0%BE%D0%B5—postgresql.

4. Офіційний сайт Docker [Електронний ресурс] – Режим доступу до ресурсу: https://www.docker.com/.

5. Gitlab документація [Електронний ресурс] – Режим доступу до ресурсу: https://docs.gitlab.com/ee/ci/.

6. Огляд Google Cloud [Електронний ресурс] – Режим доступу до ресурсу: https://cloud.google.com/docs/overview/cloud—platform—services.

7. Загальні запитання та відповіді про Python [Електронний ресурс] – Режим доступу до ресурсу: https://docs.python.org/3/faq/general.html#what—is—python.

8. Вступ до JavaScript [Електронний ресурс] – Режим доступу до ресурсу: https://javascript.info/intro.

9. Вступ до React [Електронний ресурс] – Режим доступу до ресурсу: https://reactjs.org/tutorial/tutorial.html.

10. Офіційна документація webpack [Електронний ресурс] – Режим доступу до ресурсу: https://webpack.js.org/concepts/.

11. Osmani A. Вступ до webpack [Електронний ресурс] / Addy Osmani – Режим доступу до ресурсу: https://developers.google.com/web/fundamentals/webpack/.

12. About npm [Електронний ресурс] – Режим доступу до ресурсу: https://docs.npmjs.com/about—npm.

13. @kangax. Знайомимося з Fabric.js [Електронний ресурс] / @kangax – Режим доступу до ресурсу: https://habr.com/ru/post/162367/.

14. Офіційна документація fabric.js [Електронний ресурс] – Режим доступу до ресурсу: http://fabricjs.com/docs/.

15. Керівництва для початківців HTML [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/ru/docs/Web/HTML.

16. Довідник по HTML [Електронний ресурс] – Режим доступу до ресурсу: http://htmlbook.ru/html.

17. HTML 5.2 [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3.org/TR/html52/.

18. Довідник по CSS [Електронний ресурс] – Режим доступу до ресурсу: http://htmlbook.ru/css.

19. Навчіться стилізувати HTML за допомогою CSS [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en—US/docs/Learn/CSS.

20. Що таке HTML та CSS [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3.org/standards/webdesign/htmlcss#whatcss.

ДОДАТОК А

Web - система тегування датасетів зображень

Специфікація

УКР.НТУУ«КПІ»_ТЕФ_АПЕПС_ТІ71103_21А 1

Аркушів 2

Київ – 2021

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТІ71103_20Б 2	Записка Кузьменчук Д. О. ТІ-71.docx	Текстова частина дипломної роботи
Компоненти		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТІ71103_20Б 2-1	editorState.js	Компонент для управління станом графічного редактора
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТІ71103_20Б 2-2	Editor.js	Компонент графічного редактора

ДОДАТОК Б

Web - система тегування датасетів зображень

Лістинг програми

УКР.НТУУ«КПІ»_ТЕФ_АПЕПС_ТІ71103_21Б 2

Аркушів 7

Київ – 2021

editorState

```
import {
   ADD_ACTION,
   COPY,
   DEL, DELETE_OBJECT, MOVE_OBJECT,
   PASTE,
   REDO, SET_ACTION,
   SET_CANVAS,
   SET_IMAGE, SET_IMAGE_INDEX, SET_IMAGES, SET_JOB_ID, SET_SELECTED_OBJECT_TYPE,
   UNDO,
   ZOOM_IN,
   ZOOM_OUT
} from "../actions/actionTypes";


const initialState = {
   canvas: {},
   image: {},
   undoQueue: [],
   redoQueue: [],
   images: [],
   imageIndex: 0,
   activeActionButton: null,
   objectType: '',
   jobId: -1
}

export default function editorState(state = initialState, action) {
   let undoQueueCopy = Array.from(state.undoQueue)
   let redoQueueCopy = Array.from(state.redoQueue)
   let centerX = 0
   let centerY = 0
   switch (action.type) {
      case SET_CANVAS:
         return {
            ...state,
            canvas: action.canvas
         }
      case SET_IMAGE:
         return {
            ...state,
            image: action.image
         }
      case SET_ACTION:
         if (action.action === MOVE_OBJECT || action.action === DELETE_OBJECT) {
            state.canvas.forEachObject(function (object) {
               object.selectable = true
            })
         } else {
            state.canvas.discardActiveObject()
            state.canvas.requestRenderAll()
            state.canvas.forEachObject(function (object) {
               object.selectable = false
            })
         }
         return {
            ...state,
            activeActionButton: action.action
         }
      case ZOOM_IN:
         if (state.image.scaleX < 5){
            centerX = state.image.left + (state.image.width * state.image.scaleX) / 2
            centerY = state.image.top + (state.image.height * state.image.scaleX) / 2
            state.canvas.forEachObject(function (object) {
               if (object.get('type') !== 'image') {
                  object.top = object.top + ((object.top - centerY) / (state.image.height * state.image.scaleX / 2)) * ((state.image.height / 2) *
0.05)
                  object.left = object.left + ((object.left - centerX) / (state.image.width * state.image.scaleX / 2)) * ((state.image.width / 2) *
0.05)
               }
               if (object.get('type') === 'rect') {
                  object.scaleToWidth(object.width * (object.scaleX + 0.05))
               }
            })
```

```
            state.image.scaleToWidth(state.image.width * (state.image.scaleX + 0.05))
            state.image.left = centerX - state.image.width * (state.image.scaleX) / 2
            state.image.top = centerY - state.image.height * (state.image.scaleX) / 2
            state.canvas.renderAll()
        }
        return {
            ...state
        }
    case ZOOM_OUT:
        if(state.image.scaleX > 0.1){
            centerX = state.image.left + (state.image.width * state.image.scaleX) / 2
            centerY = state.image.top + (state.image.height * state.image.scaleX) / 2
            state.canvas.forEachObject(function (object) {
                if (object.get('type') !== 'image') {
                    object.top = object.top - ((object.top - centerY) / (state.image.height * state.image.scaleX / 2)) * ((state.image.height / 2) *
0.05)
                    object.left = object.left - ((object.left - centerX) / (state.image.width * state.image.scaleX / 2)) * ((state.image.width / 2) * 0.05)
                }
                if (object.get('type') === 'rect') {
                    object.scaleToWidth(object.width * (object.scaleX - 0.05))
                }
            })
            state.image.scaleToWidth(state.image.width * (state.image.scaleX - 0.05))
            state.image.left = centerX - state.image.width * (state.image.scaleX) / 2
            state.image.top = centerY - state.image.height * (state.image.scaleX) / 2
            state.canvas.renderAll()
        }
        return {
            ...state
        }
    case UNDO:
        let undoAction = undoQueueCopy.pop()
        if (undoAction) {
            redoQueueCopy.push(undoAction)
            switch (undoAction.type) {
                case ZOOM_IN:
                    state.image.scaleToWidth(state.image.width * (state.image.scaleX - 0.05))
                    state.image.center()
                    break
                case ZOOM_OUT:
                    state.image.scaleToWidth(state.image.width * (state.image.scaleX + 0.05))
                    state.image.center()
                    break
                default:
                    break
            }
        }
        return {
            ...state,
            undoQueue: undoQueueCopy,
            redoQueue: redoQueueCopy
        }
    case REDO:
        let redoAction = redoQueueCopy.pop()
        if (redoAction) {
            undoQueueCopy.push(redoAction)
            switch (redoAction.type) {
                case ZOOM_IN:
                    state.image.scaleToWidth(state.image.width * (state.image.scaleX + 0.05))
                    state.image.center()
                    break
                case ZOOM_OUT:
                    state.image.scaleToWidth(state.image.width * (state.image.scaleX - 0.05))
                    state.image.center()
                    break
                default:
                    break
            }
        }
        return {
            ...state,
            undoQueue: undoQueueCopy,
            redoQueue: redoQueueCopy
        }
    case DEL:
        state.canvas.remove(state.canvas.getActiveObject())
        return {
            ...state,
            objectType: ''
```

```
            }
        case COPY:
            return {
                ...state
            }
        case PASTE:
            return {
                ...state
            }
        case ADD_ACTION:
            undoQueueCopy.push(action.action)
            undoQueueCopy = undoQueueCopy.slice(undoQueueCopy.length - 15 < 0 ? 0 : undoQueueCopy.length - 15,
undoQueueCopy.length)
            return {
                ...state,
                undoQueue: undoQueueCopy
            }
        case SET_IMAGE_INDEX:
            return {
                ...state,
                imageIndex: action.index
            }
        case SET_JOB_ID:
            return {
                ...state,
                jobId: action.id
            }
        case SET_IMAGES:
            return  {
                ...state,
                images: action.images
            }
        case SET_SELECTED_OBJECT_TYPE:
            return {
                ...state,
                objectType: action.objectType
            }
        default:
            return {...state}
    }
}
```

Editor.js

```
import React from 'react';
import {fabric} from 'fabric';
import {connect} from "react-redux";
import {
    addAction,
    del,
    redo,
    setCanvas,
    setImage,
    setSelectedObjectType,
    undo,
    zoomIn,
    zoomOut
} from "../../redux/actions/actions";
import {ADD_KEY_POINT, HIGHLIGHT_ZONE, ZOOM_IN, ZOOM_OUT} from "../../redux/actions/actionTypes";
import "./Editor.css"


class Editor extends React.Component {
    componentDidMount() {
        let canvas = new fabric.Canvas('canvas')
        let mouseDownX = 0
        let mouseDownY = 0
        let mouseUpX = 0
        let mouseUpY = 0

        this.props.setCanvas(canvas)

        let resizeCanvas = () => {
            canvas.setHeight(window.innerHeight * 0.8)
            canvas.setWidth(window.innerWidth * 0.8)
            if (this.props.currentImage) {
                if (this.props.currentImage.width / window.innerWidth > this.props.currentImage.height / window.innerHeight) {
                    this.props.currentImage.scaleToWidth(window.innerWidth * 0.8)
```

```
            this.props.currentImage.center()
        } else {
            this.props.currentImage.scaleToHeight(window.innerHeight * 0.8)
            this.props.currentImage.center()
        }
    }

    canvas.renderAll()
}

window.addEventListener('resize', resizeCanvas, false)

canvas.setHeight(window.innerHeight * 0.8)
canvas.setWidth(window.innerWidth * 0.8)

canvas.on('mouse:down', options => {
    mouseDownX = options.e.layerX
    mouseDownY = options.e.layerY

    if (this.props.action === ADD_KEY_POINT) {
        let circle = new fabric.Circle({
            top: options.e.layerY - 4,
            left: options.e.layerX - 4,
            radius: 4,
            fill: 'rgb(98,169,231)'
        })
        circle.setControlsVisibility({
            tl: false,
            mt: false,
            tr: false,
            ml: false,
            mr: false,
            bl: false,
            mb: false,
            br: false,
            mtr: false
        })

        circle.toObject = function() {
            return { name: 'trololo' }
        }
        canvas.add(circle);
        canvas.forEachObject(function (object) {
            object.selectable = false
        })
    }
})

canvas.on('mouse:up', options => {
    mouseUpX = options.e.layerX
    mouseUpY = options.e.layerY

    if (Math.abs(mouseDownX - mouseUpX) > 20 && Math.abs(mouseDownY - mouseUpY) > 20
        && this.props.action === HIGHLIGHT_ZONE) {
        let rect = new fabric.Rect({
            left: mouseDownX > mouseUpX ? mouseUpX : mouseDownX,
            top: mouseDownY > mouseUpY ? mouseUpY : mouseDownY,
            width: Math.abs(mouseDownX - mouseUpX) * (1 / this.props.currentImage.scaleX),
            height: Math.abs(mouseDownY - mouseUpY) * (1 / this.props.currentImage.scaleX),
            fill: 'rgba(0,0,0,0)',
            stroke: 'black',
            strokeWidth: 1,
            scaleX: this.props.currentImage.scaleX,
            scaleY: this.props.currentImage.scaleY
        });
        rect.setControlsVisibility(
            {
                mtr: false
            }
        )

        canvas.add(rect);
        canvas.forEachObject(function (object) {
            object.selectable = false
        })
    }
})

canvas.on('selection:created', e => {
```

```
    this.props.setSelectedObjectType('')
    this.props.setSelectedObjectType(canvas.getActiveObject().get('type'))

    if (canvas.getActiveObject().get('type') === 'image') {
      canvas.discardActiveObject();
      let sel = new fabric.ActiveSelection(canvas.getObjects(), {
        canvas: canvas,
      });
      sel.setControlsVisibility({
        tl: false,
        mt: false,
        tr: false,
        ml: false,
        mr: false,
        bl: false,
        mb: false,
        br: false,
        mtr: false
      })
      canvas.setActiveObject(sel);
      canvas.requestRenderAll();
    }
  })

  canvas.on('selection:updated', e => {
    this.props.setSelectedObjectType(canvas.getActiveObject().get('type'))
    this.props.setSelectedObjectType('')
    this.props.setSelectedObjectType(canvas.getActiveObject().get('type'))
    if (canvas.getActiveObject().get('type') === 'image') {
      canvas.discardActiveObject();
      let sel = new fabric.ActiveSelection(canvas.getObjects(), {
        canvas: canvas,
      });
      sel.setControlsVisibility({
        tl: false,
        mt: false,
        tr: false,
        ml: false,
        mr: false,
        bl: false,
        mb: false,
        br: false,
        mtr: false
      })
      canvas.setActiveObject(sel);
      canvas.requestRenderAll();
    }
  })

  // canvas.on('selection:cleared', e => {
  //     console.log('cleared')
  //     this.props.setSelectedObjectType('')
  // })

  canvas.on('mouse:wheel', options => {
    canvas.discardActiveObject();
    canvas.requestRenderAll();
    if (options.e.deltaY < 0) {
      this.props.zoomIn()
      this.props.addAction({
        type: ZOOM_IN,
        options: {}
      })
    } else {
      this.props.zoomOut()
      this.props.addAction({
        type: ZOOM_OUT,
        options: {}
      })
    }
  })

  document.onkeydown = e => {
    if (e.ctrlKey && e.keyCode === 90 && !e.shiftKey) {
      // console.log("Undo")
      this.props.undo()
    }
    if (e.shiftKey && e.ctrlKey && e.keyCode === 90) {
      // console.log("Redo")
```

```
            this.props.redo()
          }
          if (e.keyCode === 8) {
            // console.log("Delete")
            this.props.delete()
          }
          if (e.ctrlKey && e.keyCode === 67) {
            console.log("Copy")
          }
          if (e.ctrlKey && e.keyCode === 86) {
            console.log("Paste")
          }
        }
    }

    render() {
        return <div className='Editor'>
            <canvas id="canvas"/>
        </div>
    }
}

function mapStateToProps(state) {
    return {
        canvas: state.editorState.canvas,
        currentImage: state.editorState.image,
        action: state.editorState.activeActionButton
    }
}

function mapDispatchToProps(dispatch) {
    return {
        setCanvas: canvas => dispatch(setCanvas(canvas)),
        setImage: image => dispatch(setImage(image)),
        addAction: action => dispatch(addAction(action)),
        zoomIn: () => dispatch(zoomIn()),
        zoomOut: () => dispatch(zoomOut()),
        undo: () => dispatch(undo()),
        redo: () => dispatch(redo()),
        delete: () => dispatch(del()),
        setSelectedObjectType: type => dispatch(setSelectedObjectType(type))
    }
}

export default connect(mapStateToProps, mapDispatchToProps)(Editor)
```

ДОДАТОК В

Web - система тегування датасетів зображень

Опис програмного коду

УКР.НТУУ«КПІ»_ТЕФ_АПЕПС_ТІ71103_21В 3

Аркушів 11

Київ – 2021

АНОТАЦІЯ

Розроблена система є застосунком призначеним для тегування датасетів зображень з метою подальшого використання при тренуванні моделей штучного інтелекту в такій області як computer vision. Головними можливостями розробленої системи є:

— створення контексту завдань для організації роботи користувачів;

— перегляд контексту завдань;

— моніторинг стану виконання завдань;

— тегування датасетів за допомогою графічного редактора;

— завантаження тегованих датасетів.

Система розроблена за допомогою середовищ розробки WebStorm та PyCharm з використанням таких мов програмування як Python та JavaScript, з використанням таких фреймворків як React, Django та DRF.

ЗМІСТ

ЗАГАЛЬНІ ВІДОМОСТІ

Додаток А містить специфікацію розробленої системи.

Додаток Б містить програмний код основних модулів які містять код з реалізацією графічного редактора.

Додаток В містить опис основних класів розробленої системи призначеної для тегування датасетів зображень.

Програма не потребує жодної інсталяції так як система розгорнута в хмарному сервісі GCP (Google Cloud Platform), тому все що потрібно користувачу це:

— персональний комп’ютер;

— один з популярних браузерів;

— доступ до мережі інтернет.

Система розроблена за допомогою середовищ розробки WebStorm та PyCharm з використанням таких мов програмування як Python та JavaScript, з використанням таких фреймворків як React, Django та DRF.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Система розроблена для тегування датасетів зображень з метою подальшого використання при тренуванні моделей штучного інтелекту в такій області як computer vision має наступні можливості:

— створення проекту;

— створення задачі;

— створення роботи;

— видалення проекту;

— видалення задачі;

— видалення роботи;

— пошук за назвою серед проектів задач та робіт;

— підтримка різних форматів зображень при використанні їх в якості зображень для попереднього перегляду та в ролі змісту для виконання роботи;

— завантаження тегованого датасету для завдання;

— автоматична зміна стану виконання проектів, задач та робіт після завершення частини роботи на будь якому рівні ієрархії;

— динамічне обновлення списків проектів задач та робіт після видалення з них елементів;

— направлення користувача на попередню сторінку після успішного створення проектів задач та робіт;

— відображення помилок у разі якщо виникли якісь неполадки при відправленні запитів;

— авторизації та реєстрації;

— переходи між зображеннями в ході виконання робіт;

— можливість переміщувати в графічному редакторі такі об’єкти як області та ключові точки;

— можливість виділяти області в графічному редакторі;

— можливість виставляти ключові точки в графічному редакторі;

— можливість зміни розмірів виділеної області в графічному редакторі;

— переміщення всіх об’єктів при умові що було виділено зображення

— видалення областей та ключових точок;

— вказання емоцій на виділених областях, а також вказання зони обличчя до якої належить ключова точка;

— переміщення об’єктів при зміні розмірів зображення;

— зміна виділених областей при зміні розмірів зображення;

— відображення вибраних емоцій та зон на обличчі при виборі виділених областей та ключових точок для яких попередньо були вказані ці параметри;

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Згідно з архітектурою система складається з двох головних підсистем призначених для виконання наступних задач:

— тегування датасетів за допомогою графічного редактора;

— створення контексту завдань для організації роботи користувачів.

Модуль для тегування зображень складається з набору функцій призначених для створення тегів, а також для маніпуляції над ними при зверненні безпосередньо до них а також при зміні розмірів зображення.

Модуль для створення контексту завдань для організації роботи користувачів служить для створення, зміни та видалення елементів ієрархії, а також моніторингу стану їх виконання.

ВИКОРИСТАНІ ТЕХНІЧНІ ЗАСОБИ

Програма не потребує жодної інсталяції так як система розгорнута в хмарному сервісі GCP (Google Cloud Platform), тому все що потрібно користувачу це:

— персональний комп’ютер;

— один з популярних браузерів;

— доступ до мережі інтернет.

Для того щоб почати користуватись системою необхідно перейти в браузері за посиланням на сайт https://web.dataset.yourcofounder.com.ua/ та створити обліковий запис.

ВИКЛИК І ЗАВАНТАЖЕННЯ

Програма не потребує жодної інсталяції так як система розгорнута в хмарному сервісі GCP (Google Cloud Platform), тому все що потрібно користувачу це:

— персональний комп’ютер;

— один з популярних браузерів;

— доступ до мережі інтернет.

Для того щоб почати користуватись системою необхідно перейти в браузері за посиланням на сайт https://web.dataset.yourcofounder.com.ua/ та створити обліковий запис.

ВХІДНІ ДАНІ

Вхідними даними для роботи системи є зображення будь-якого формату які формують контекст завдань для організації роботи який складається з трьох рівнів таких як:

— проекти;

— задачі;

— роботи.

ВИХІДНІ ДАНІ

Вихідними даними роботи системи є файли з розширенням .json які складаються зі списку об’єктів кожен з яких містить наступний набір полів які характеризують його положення відносно зображення не залежно від того де знаходиться саме зображення з урахуванням зміни лінійних розмірів зображення викликаними в процесі роботи з ним. Також об’єкти містять інформацію про свій тип, а також що на них зображено при умові якщо користувач це зробив. Тому було вирішено створити такі поля:

— положення верхньої лівої точки за координатою X;

— положення верхньої лівої точки за координатою Y;

— положення нижньої лівої точки за координатою X у випадку якщо це область;

— положення нижньої лівої точки за координатою Y у випадку якщо це область;

— тип об’єкту;

— контекст якому відповідає об’єкт якщо він вказаний.