

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМ. І. І.  
СІКОРСЬКОГО»**

**ННК «ІПСА»**

**Кафедра системного проектуванн**

**«ПРОЕКТУВАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ»**

**Лабораторна робота № 4**

**Розробити тестовий мобільний додаток за темою індивідуального  
завдання.**

**Виконала:**

**студентка 4-го курсу**

**групи ДА-72**

**Жужа Ангеліна**

**Київ – 2020**

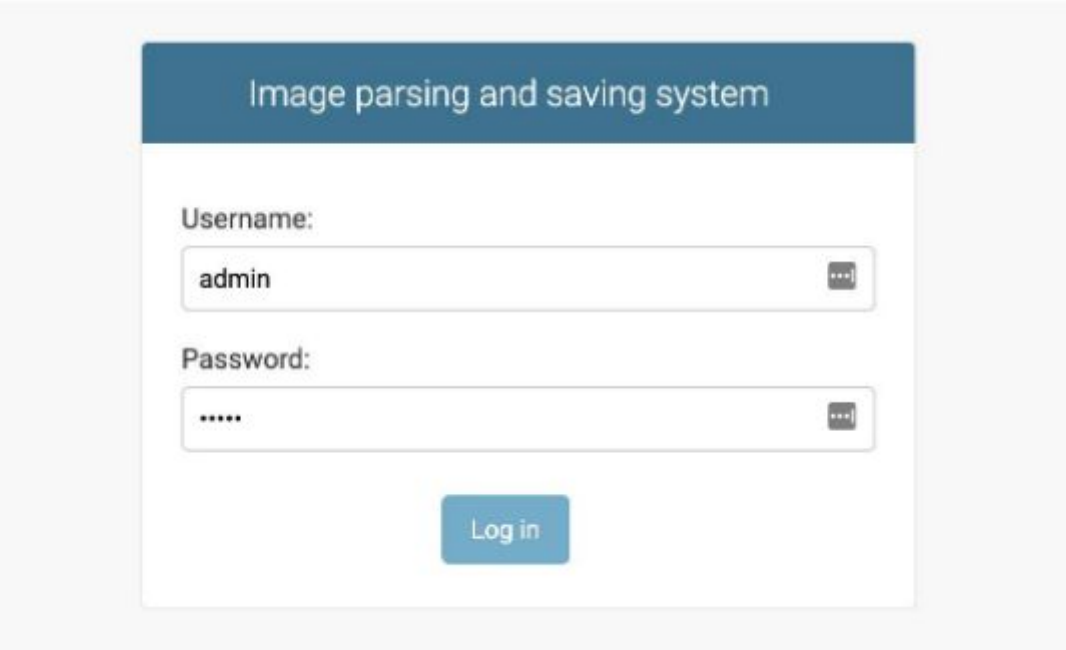
## Завдання

1. Створити інтерфейс користувача мобільного додатку інформаційної системи.
2. Розробити основний функціонал мобільного додатку CRUD.
3. Провести тестування мобільного додатку відповідно до SRS з л.р. 2.

## Хід роботи

1. Описати інтерфейс користувача мобільного додатку.

- Сторінка авторизації користувача



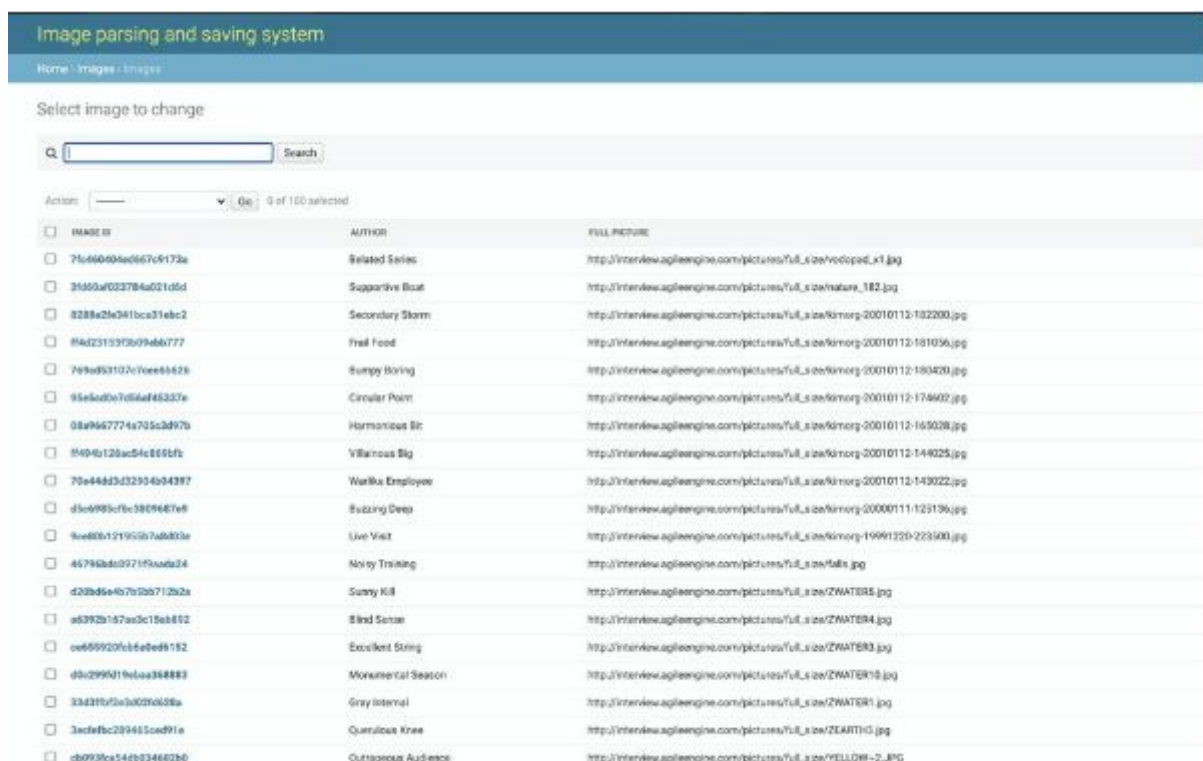
The screenshot shows a mobile application interface for the 'Image parsing and saving system'. It features a dark blue header with the system name. Below the header is a white login form with two input fields: 'Username:' containing the text 'admin' and 'Password:' containing six dots. Each field has a small icon on the right. At the bottom of the form is a blue 'Log in' button.

- сторінка авторизованого користувача

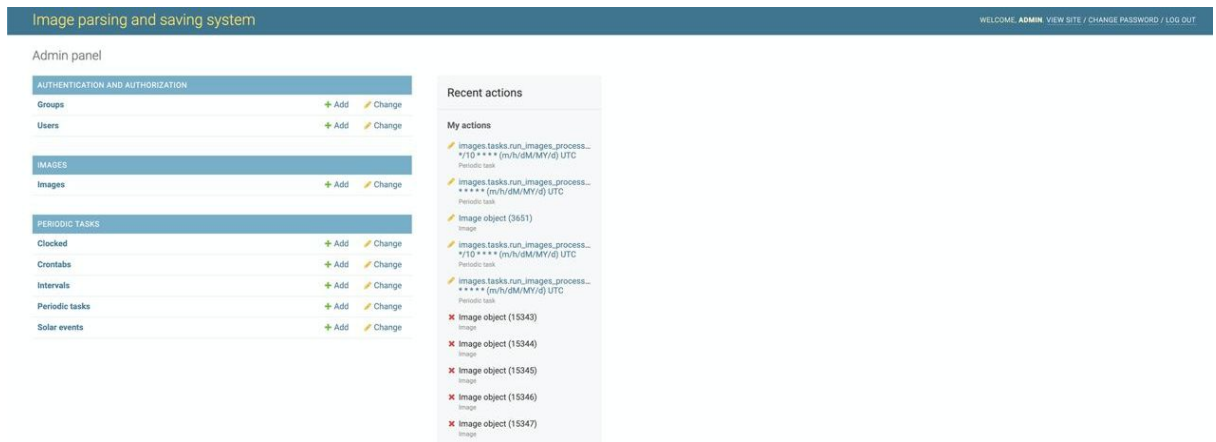


На головній сторінці у користувача знаходиться фільтр для пошуку, поле для вибору дії, та критерії для фільтра.

- сторінка парсінгу зображень

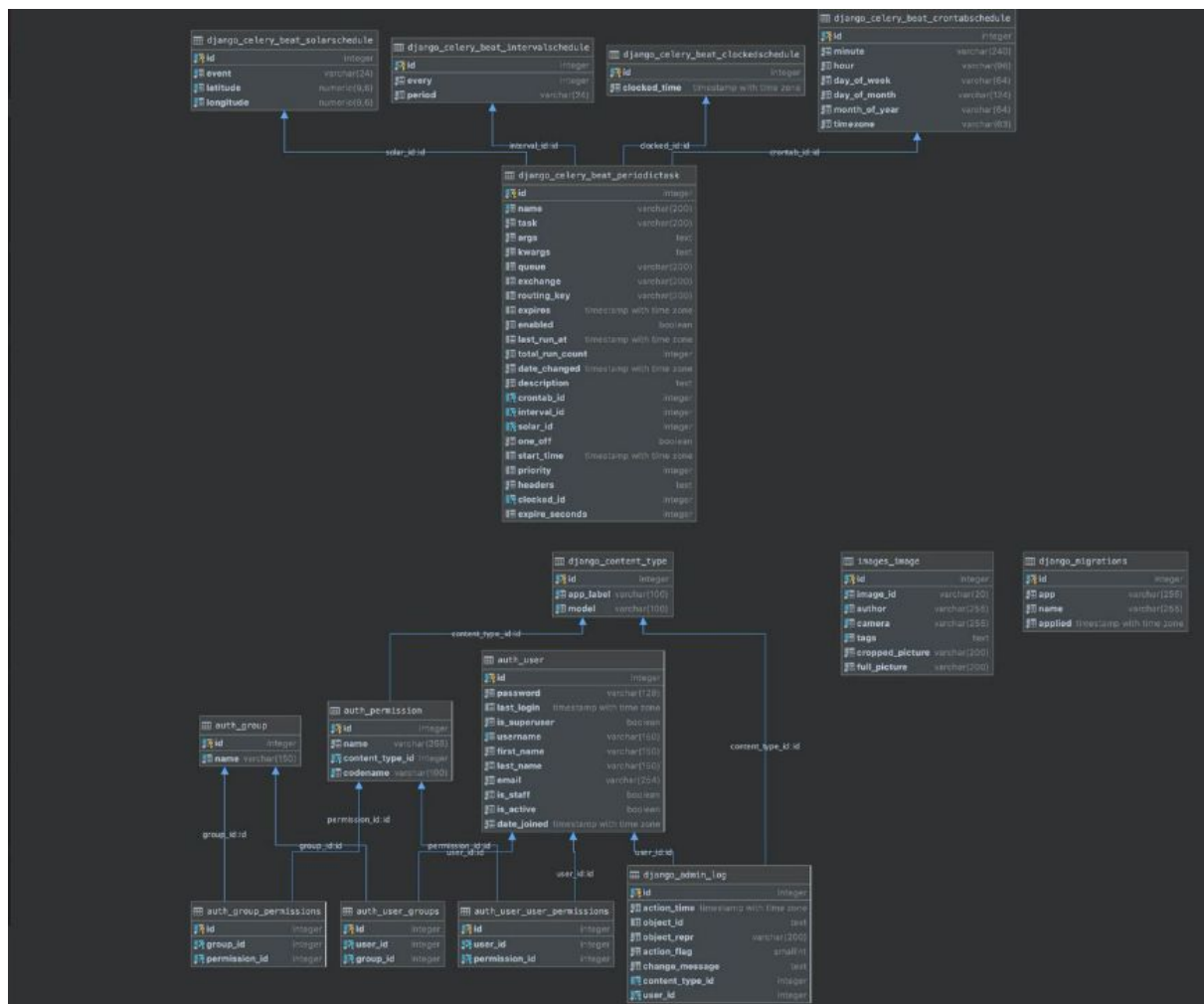


- сторінка для адміна



На сторінці знаходяться блок з нещодавніми діями та панель, де адмін може додавати та змінювати користувачів та фото.

## 2. Опис архітектури веб-додатку



## 3. Код для основного функціонала

- парсінг фото

```
def process_images():
    page_num = 1
    while True:
        page_data = retrieve_images_page(page_num)
        image_ids = [image_id['id'] for image_id in page_data['pictures']]

        with ThreadPoolExecutor() as executor:
            images_data = list(executor.map(retrieve_image_info, image_ids))

        with transaction.atomic():
            Image.objects.filter(image_id__in=image_ids).delete()
            Image.objects.bulk_create(
                map(lambda data: Image(**data), images_data),
                ignore_conflicts=True # in case of parallel tasks run
            )
            # or use kind of manager_utils.bulk_upsert to update existing data

        if not page_data['hasMore']:
            break

    page_num += 1
```

- пошук фото

```
from django.db import models
from django.db.models import QuerySet

from images.models import Image

_DEFAULT_LOOKUP = '__icontains'
SEARCH_FIELDS = [
    field.name
    for field in Image._meta.get_fields()
    if not field.primary_key and field.db_index
]

def search_images(qs: QuerySet, search_query: str):
    filter_q = models.Q()

    for field_name in SEARCH_FIELDS:
        filter_q |= models.Q(**{f'{field_name}_{DEFAULT_LOOKUP}': search_query})

    return qs.filter(filter_q)
```

- отримання даних з сервісу

```
import requests
from django.conf import settings

from images.services.token_cache import token_cached
from shared.requests import raise_for_status, change_response_data
from shared.urls import join_paths, add_param

IMAGES_API_BASE_URL = 'http://interview.agileengine.com'

@token_cached
@change_response_data(return_only_field='token')
@raise_for_status
def retrieve_api_token():
    url = join_paths(IMAGES_API_BASE_URL, 'auth')
    data = {'apiKey': settings.IMAGES_API_KEY}
    return requests.post(url, json=data)

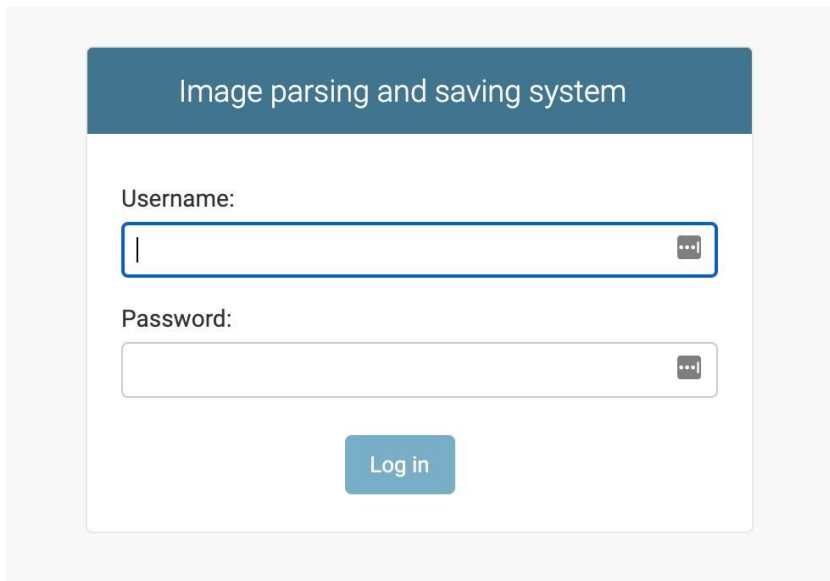
def _authorization_headers():
    return {'Authorization': f'Bearer {retrieve_api_token()}'}

@raise_for_status
def retrieve_images_page(page_num):
    url = join_paths(IMAGES_API_BASE_URL, 'images')
    return requests.get(
        add_param(url, page=page_num),
        headers=_authorization_headers()
    )

@change_response_data(image_id='id')
@raise_for_status
def retrieve_image_info(image_id):
    url = join_paths(IMAGES_API_BASE_URL, 'images', image_id)
    return requests.get(url, headers=_authorization_headers())
```

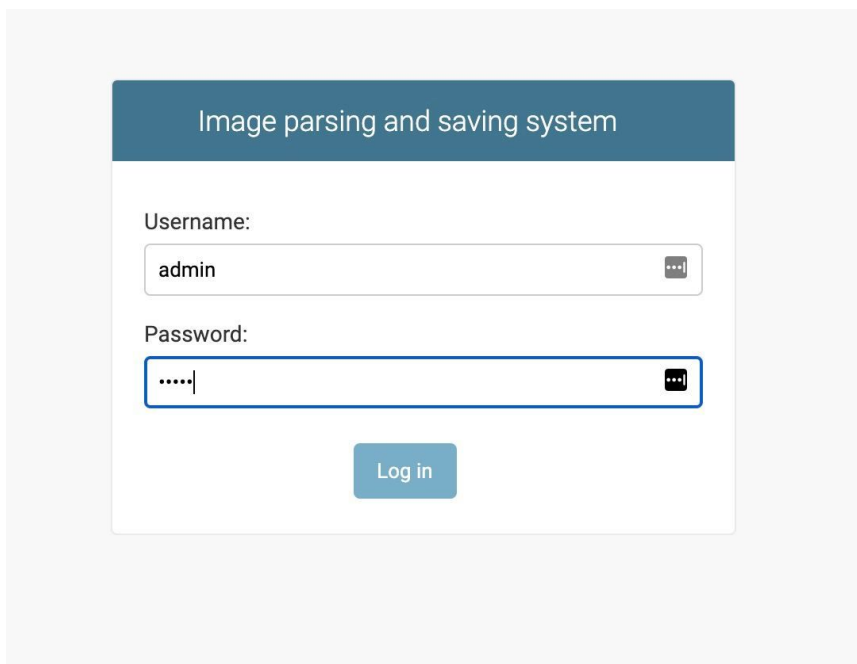
#### 4. Тестування веб-додатку

- тестування реєстрації користувача



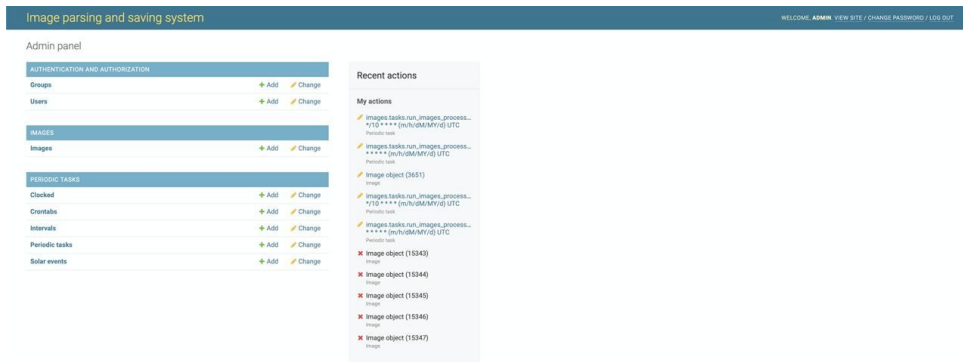
The screenshot shows a login form titled "Image parsing and saving system". It contains two input fields: "Username:" and "Password:". The "Username:" field has a cursor at the start. The "Password:" field is empty. Below the fields is a blue "Log in" button.

Вводимо логін та пароль



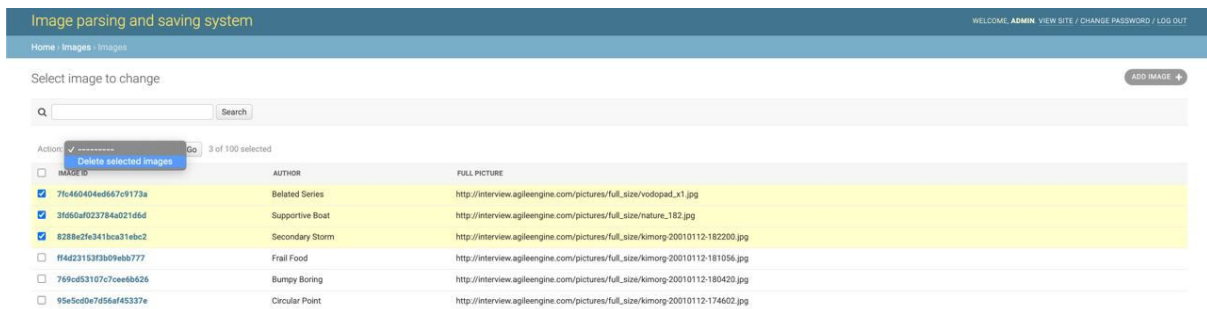
The screenshot shows the same login form, but now the "Username:" field contains the text "admin". The "Password:" field contains masked characters (dots). The "Log in" button remains visible.

Опиняємося на головній сторінці

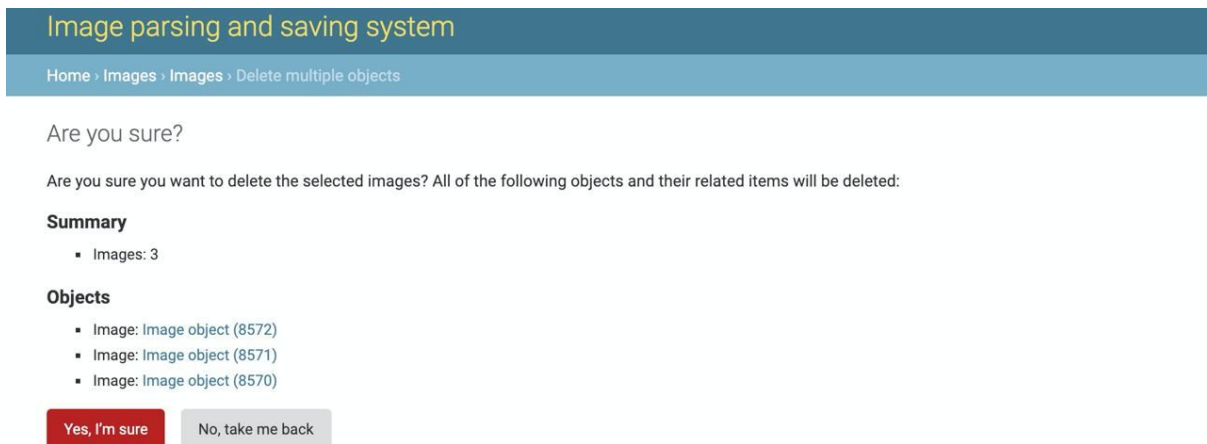


- видалення фото

Обираємо фото, що хочемо видалити та обираємо дію delete

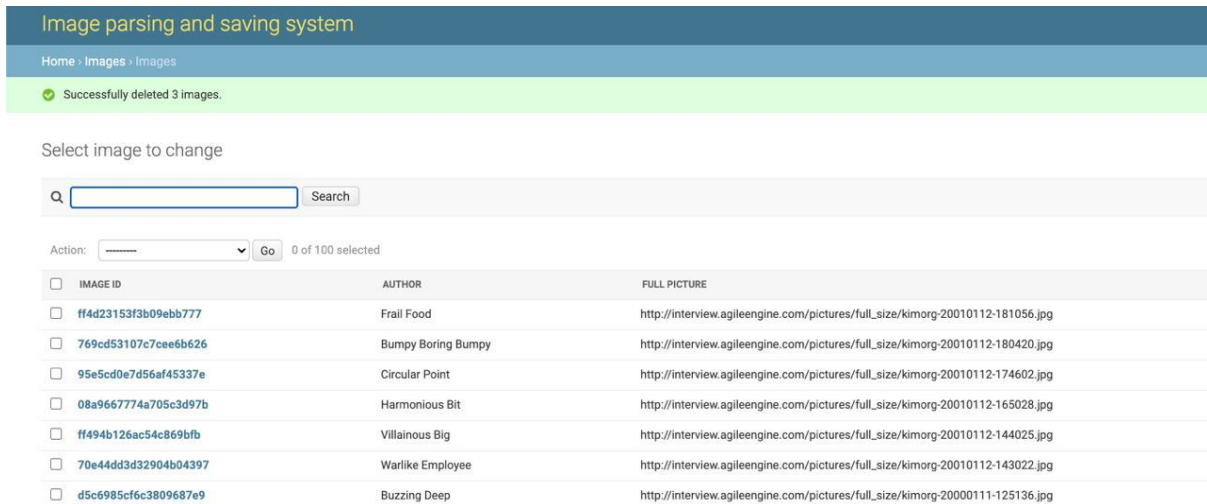


Підтверджуємо видалення зображень



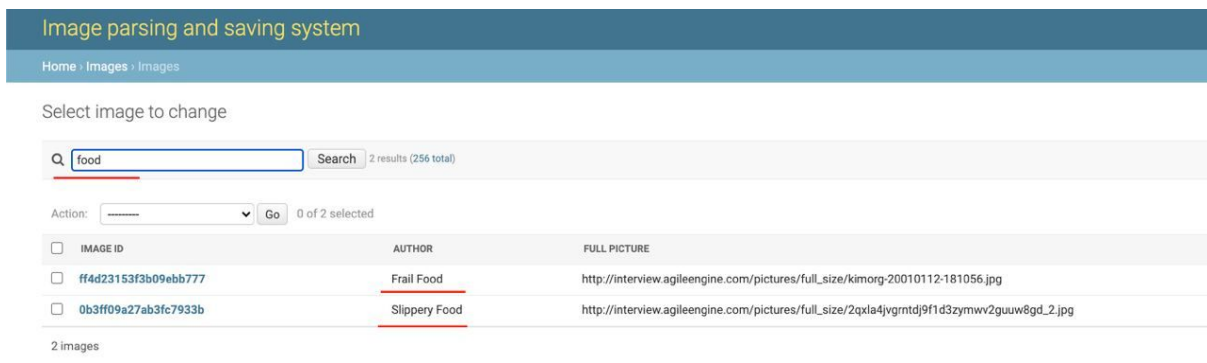
Зверху бачимо надпис, що 3 фото видаленні та у загальному списку їх не має





- пошук по фільтру

Вводимо слово за яким буде відбуватися пошук та отримаємо результат



**Висновок:** у ЛР №4 розробляла інформаційний сервіс для збереження фото з різних серверів та проводила функціональне тестування, а саме пошуку, видалення фото та авторизації користувача. Основна перевага функціонального тестування - імітація фактичного використання системи; Недоліки функціонального тестування: можливість упущення логічних помилок у програмному забезпеченні; ймовірність надмірного тестування. При розробці інформаційного сервісу розробила пошук, парсінг, авторизацію, видалення, редагування. В основному не виникало труднощів при виконанні роботи.