МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ, КОМП'ЮТЕРНОЇ
ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ
КАФЕДРА КОМП'ЮТЕРНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

_____ Аліна САВЧЕНКО
(підпис) (ПІБ)

"_____"_____2022 р.

ДИПЛОМНИЙ ПРОЕКТ
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСНИКА ОСВІТНЬОГО СТУПЕНЯ "БАКАЛАВР"

ЗА СПЕЦІАЛЬНІСТЮ 122 «КОМП'ЮТЕРНІ НАУКИ»

Тема: Веб-сервіс для колористів

Виконавець: Самчук Богдан Володимирович

Керівник: к.т.н., доцент Зудов Олег Миколайович

Нормоконтролер: _____ Володимир БОРОВИК
(підпис)

Київ 2022

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії

Кафедра комп'ютерних інформаційних технологій

Освітній ступінь Бакалавр

Спеціальність 122 «Комп'ютерні науки»

Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ:

Завідувач кафедри

_____ Аліна Савченко
(підпис) (ПІБ)

«____»_____2022 р.

ЗАВДАННЯ

на виконання дипломного проекту

Самчука Богдана Володимировича
(прізвище, ім'я, по батькові випускника)

1. Тема дипломного проекту: Веб-сервіс для колористів

затверджена наказом ректора від «29» 04 2022 р., № 454/ст

2. Термін виконання проекту: з «09» 05 2022 р. по «19» 06 2022 р.

3. Вихідні дані до роботи: 1. Зробити аналіз представлення кольору в ЕОМ. 2. Розглянути інструменти для розробки веб-сервісів. 3. Розглянути питання сортування зображень.

4. Зміст пояснювальної записки: 1. Фізичні властивості кольору та його інтерпретація у електронно-обчислювальних машинах. 2. Засоби реалізації веб-сервісу. 3. Реалізація веб-сервісу. Сортування пікселів по їх характеристикам

5. Перелік обов'язкового графічного (ілюстративного) матеріалу: 1. Відомі аналоги веб-сервісу. 2. Колірні моделі. 3. Використані технології. 4. Формула переведення RGB у HSV модель. 5. Діаграма послідовності сортування зображень. 6. Сторінка вивантаження файлу. 7. Головна сторінка з можливістю вибору кольору.

6. Календарний план-графік

№ з/п	Завдання	Термін виконання	Підпис керівника
1.	Розглянути поняття кольору та його представленні у ЕОМ	10.05.2022 - 12.05.2022	
2.	Розібрати види колірних моделей та зв’язків між ними	13.05.2022 – 20.05.2022	
3.	Здійснити пошук інструментів для розробки веб-сервісу	21.05.2022 – 27.05.2022	
4.	Розробити детальну реалізацію веб-сервісу з із можливістю сортування зображень	28.05.2022 – 03.06.2022	
5.	Оформити текстові і графічні матеріали дипломного проекту	04.06.2022 – 09.06.2022	
6.	Підготовити презентацію по дипломному проекту	10.06.2022 – 13.06.2022	

7. Дата видачі завдання: «_10_»___05___ 2022р.

Керівник дипломного проекту _____ Зудов О. М.
(підпис керівника) (ПІБ)

Завдання прийняв до виконання _____ Самчук Б. В.
(підпис випускника) (ПІБ)

РЕФЕРАТ

Пояснювальна записка до дипломного проекту «Веб-сервіс для колористів» представлена на 43 сторінках, містить 14 рисунків та 16 інформаційних джерел.

КОЛІР, BACKEND, FRONTEND, RGB, HSL, SPRING, JAVA, MVC, IMAGE, ВЕБ-СЕРВІС, СОРТУВАННЯ, BLOL-STORAGE, ХАРАКТЕРИСТИКИ ЗОБРАЖЕННЯ, ПІКСЕЛІ.

Об’єкт дослідження — веб-сервіс для дизайнерів-колористів.

Мета проекту — підвищення швидкості та якості роботи колористів за допомогою спеціального веб-додатку для сортування зображень.

Метод дослідження та програмні засоби — побудова веб-сервісу для колористів із дослідженням характеристик зображень та використанням новітніх програмних засобів на базі мови програмування Java.

Результат проекту — веб-сервіс із функціоналом зберігання зображень та можливістю їх сортування із заданим кольором.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

CI/CD (Continuous Integration/ Continuous Deployment) – технологія автоматизації тестування та доставки нових модулів проекту, що розробляється.

Framework (Software framework) – це програмне забезпечення, яке розробляється та використовується розробниками для створення додатків.

Frontend (Data Transfer object) це публічна частина web-додатків (вебсайтів), з якою користувач може взаємодіяти і контактувати напряму.

Backend (Data Access Object) – програмно-апаратна частина сервісу, що відповідає за функціонування його внутрішньої частини.

Бін (Bean) – це об'єкти класів, які є компонентами програми та управляються Spring фреймворком.

SAS (Shared Access Signature) – це рядок символів, який містить усю інформацію, необхідну для аутентифікації, а також для визначення служби та ресурсу сховища Azure, дозволів, необхідних для доступу, і періоду часу, протягом якого підпис дійсний.

ВСТУП

Дизайнер — одна з найпопулярніших професій у сучасному інформаційному світі. Ми бачимо результат їх праці щодня і повсюди. Онлайн-сервіси, ігри, сайти, інтернет-магазини — все це уже трапляється нам кожного дня, хоча вони знаходяться лише в мережі інтернет. Тим не менш, перше що бачать користувачі — колірні схеми та структуру проекту. Саме колір часто називають основним інструментом дизайнера-колориста, адже він може підштовхнути людей до прийняття рішень, дати оцінку товару на підсвідомому рівні буквально за час менший секунди.

Для вибору правильного кольору під кожний конкретний випадок, потрібно мати чималий обсяг знань та чутливу інтуїцію. Одна з найрозповсюдженіших проблем колористів - це вибір потрібного кольору під проект, товар тощо.

Нерідко перебирається величезна кількість варіантів колірних схем для знаходження необхідного поєднання. Для спрощення проходження цієї процедури щоразу, дизайнери мають у запасі шаблони фонів, градієнтів, візерунків. Проблема полягає у їх структурованому зберіганні, часто пошук потрібного зображення проводиться по пам’яті, що не дозволяє швидко добитись необхідного результату та приводить до додаткових трат часу.

Щоб полегшити роботу значній кількості колористів, їм потрібен інструмент для швидкого та зручного пошуку серед графічних об’єктів, з можливістю відсортування по колірній схемі, яскравості, контрастності тощо. Це дозволить зменшити кількість витраченого часу на пошук, та збільшення концентрації на вибір правильного рішення.

Пошук потрібних файлів, звичайно, не перша проблема для людей цієї професії. Часто вони створюють у графічних редакторах, креслять на папері, малюють на планшеті власні рішення. Дизайнерських ідей зараз настільки багато, що необов’язково створювати щось нове для кожного завдання чи проекту. Структура таких сервісів як інтернет-магазини досить поширена,

перевірена часом та людьми, для яких таке рішення оформлення здається простим та інтуїтивним. Оформлення саме таких невеликих проектів з часто вживаними архітектурними структурами становить левову долю завдань для дизайнерів.

Для вищезгаданої ніші людей буде корисно розробити веб-сервіс зберігання зображень, на якому буде доступна функція пошуку, сортування за кольором чи інших графічних характеристик стане корисним і має всі шанси стати щоденний обов'язковим інструментом. Для зберігання великого масиву даних, такий сервіс не має бути програмою, що розташована на одному конкретному ПК. Веб-сервіси розміщуються із використання хмарних технологій, тому доступ буде звідусіль де є доступ до інтернету.

Для такого результату потрібно проаналізувати теорію кольору та його інтерпретація у електронно-обчислювальних машинах. Необхідно порівняти вживані випадки перетворення та зберігання графічних об'єктів задля ефективних та правильних операцій із порівнянням зображень.

Також мають бути проаналізовані інструменти для розробки програмної частини веб-сервісу та його базового функціоналу. Використання новітніх технологій покращить оптимальний розподіл ресурсів та швидку реалізацію логічної інфраструктури на стороні бек-енду.

Складання покрокових рішень реалізації програми та приведення прикладів використання веб-сервісу із наведенням графічних ілюстрацій базового використання, стане основною ціллю роботи.

РОЗДІЛ 1
ФІЗИЧНІ ВЛАСТИВОСТІ КОЛЬОРУ ТА ЙОГО ІНТЕРПРЕТАЦІЯ У ЕЛЕКТРОННО-ОБЧИСЛЮВАЛЬНИХ МАШИНАХ

1.1 Теорія кольору

Колір – це наша суб’єктивна реакція на світло у видимому діапазоні частот. Довжина електромагнітної хвилі повинна знаходитись від 380(фіолетовий) до 750(червоний) нанометрів[1]. Все що ми бачимо навколо, предмети, явища і т. д., відбиває світло, знаходиться саме в цьому діапазоні.

Властивості кольору можна розділити на фізичні та суб'єктивні. Фізичні властивості є наслідком розуміння кольору як фізичного явища, пучка електромагнітних хвиль певної довжини або потоку частинок певної щільності. Суб'єктивні властивості – це особливості людського зору і сприйняття, вплив кольору на психіку людини, ілюзії колірного зору.

1.2 Фізичні властивості світла

Світло — це електромагнітне випромінювання, яке може виявити людське око. Воно виникає в надзвичайно широкому діапазоні довжин хвиль, від гамма променів до радіохвиль. В залежності від довжини хвилі видимого світла, наше око по різному реагує і ми бачимо широкий спектр кольору.

1.3 Білий як суміш різних кольорів

Світло має дуалістичну природу: це електромагнітні хвилі або ж фотони - кванти електромагнітного випромінювання. Близько 1700 року Ісак Ньютон дійшов висновку, що світло являє собою групу частинок (корпускулярна теорія). Приблизно в той же час були інші вчені, які думали, що світло може бути хвилею (хвильова теорія). Світло поширюється прямолінійно, і тому для Ньютона було цілком природно думати про нього як про надзвичайно малі частинки, що випускаються джерелом світла і відбиваються предметами.

Однак корпускулярна теорія не може пояснити хвилеподібні світлові явища, такі як дифракція та інтерференція. З іншого боку, хвильова теорія неспроможна пояснити, чому фотони вилітають із металу, піддається впливу світла (явище називається фотоефектом, відкритим наприкінці XIX століття).

Хоча Ісак Ньютон був винахідником корпускулярної теорії, саме він довів, що світло має частотні характеристики за допомогою всім відомого експерименту із призмою. Ньютон розпочав свої дослідження з того, що прорізав маленький отвір у віконній шторі, щоб впустити сонячне світло, яке позначилося на його стіні у вигляді круглої освітленої області. Заломлюючись призмою, воно перетворювалося на довгасту область з веселкою квітів.

Заінтригований зміною форми, Ньютон вирізав безліч отворів різних розмірів і форм, але незалежно від форми вихідного променя заломлене світло ставало більш довгим. Ньютон також помістив другу призму того ж типу на шляху світла і зміг знову перетворити кольори на біле світло. Це показало, що біле світло не було чистим, а складалося із суміші різних кольорів.

Вирішальним експериментом Ньютона було заломлення світла на шматок дерева, у якому був просвердлений невеликий отвір. Таким чином він отримав промінь світла чистого кольору. Він зміг показати, що синє світло, наприклад, заломлюючись через другу призму, знову дає лише синє світло.

Червоне світло дало тільки червоне світло. Більш того, кут, під яким світло відбивалося на його стіну, залежало від кольору. Різні кольори світла мали різний ступінь «заломлюваності» (використовуючи термін Ньютона), який був невід'ємною властивістю цього кольору.

Рис. 1.1. Експеримент Ньютона із призмою [2]

1.4 Пікселі як одиниця кольору

Ми бачимо зображення із різними кольорами через екран смартфону, телевізору чи великих екранних панелей. Сучасні технології опираються на те що білий колір це суміш всіх кольорів. Базовим елементом такого цифрового зображенння є Піксель. (pix = малюнок, el = елемент). Це найменша одиниця цифрового зображення або графіки, яка може бути відображена та представлена на цифровому пристрої відображення.

Піксель — основна логічна одиниця цифрової графіки. Пікселі об’єднуються, щоб утворити цілісне зображення, відео, текст або будь-яку видиму річ на дисплеї комп’ютера.

Піксель представлений точкою або квадратом на екрані монітора комп’ютера тому зображення створюються за допомогою сітки геометричних координат.

Залежно від відеокарти та монітора, кількість, розмір і колірної комбінації пікселів змінюються і вимірюються з точки зору роздільної здатності дисплея. Наприклад, комп’ютер з роздільною здатністю дисплея 1280 на 768 відтворить на екрані максимум 98 3040 пікселів.

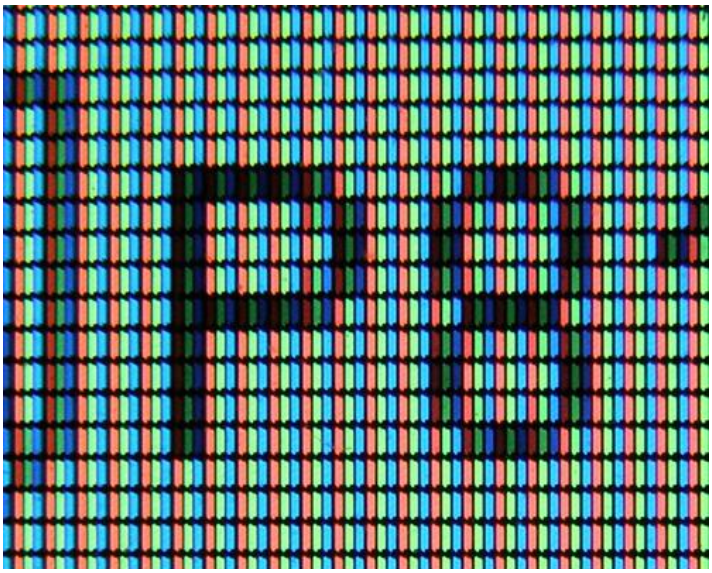

Рис. 1.2 Вигляд пікселя на моніторі [3]

Роздільна здатність пікселів також визначає якість відображення; більше пікселів на дюйм екрана монітора дає кращі результати зображення. Наприклад, зображення розміром 2,1 мегапікселя містить 2 073 600 пікселів, оскільки воно має роздільну здатність 1920 x 1080.

Фізичний розмір пікселя змінюється в залежності від роздільної здатності дисплея. Він дорівнюватиме розміру кроку точки, якщо на дисплеї встановлено максимальну роздільну здатність, і буде більшим, якщо роздільна здатність нижча, оскільки кожен піксель використовуватиме більше точок. Через це окремі пікселі можуть стати видимими, що призведе до блокового та масивного зображення, яке визначається як «піксельне».

Пікселі рівномірно розташовані у двовимірній сітці, хоча доступні деякі різні моделі вибірки. Наприклад, на РК-екранах три основні кольори відбираються в різних місцях шахової сітки, тоді як цифрові кольорові камери використовують більш регулярну сітку.

У комп'ютерних моніторах пікселі мають квадратну форму, що означає, що їхні вертикальні та горизонтальні кроки вибірки рівні. В інших системах, таких як анаморфний широкоекранний формат стандарту цифрового відео 601, форма пікселя є прямокутною.

Кожен піксель має унікальну логічну адресу, розмір вісім біт або більше, а в більшості високоякісних пристроїв відображення — здатність проектувати мільйони різних кольорів. Колір кожного пікселя визначається специфічним змішуванням трьох основних компонентів колірного спектру RGB.

Залежно від системи кольору, для визначення кожного колірного компонента пікселя може бути виділена різна кількість байтів. Наприклад, у 8-розрядних системах кольору лише один байт виділяється на піксель, що обмежує палітру лише 256 кольорами.

У загальних 24-розрядних системах кольорів, які використовуються майже для всіх моніторів ПК і дисплеїв смартфонів, виділено три байти, по одному для кожного кольору шкали RGB, що призводить до 16 777 216 кольорових варіацій. 30-бітна система глибокого кольору виділяє по 10 біт червоного, зеленого та синього кольорів, що становить 1,073 мільярда кольорових варіацій. Однак, оскільки людське око не може розрізняти більше десяти мільйонів кольорів, більша кількість кольорових варіацій не обов'язково додає більше деталей і навіть може призвести до проблем з кольоровими смугами.

1.5 Колірні моделі

Колірна модель – це система, яка допомагає нам визначати та описувати кольори за допомогою числових значень. Існує багато типів колірних моделей,

які використовують різні математичні системи для представлення кольорів, хоча більшість колірних моделей зазвичай використовують комбінацію трьох або чотирьох значень або колірних компонентів.[5]

Найпопулярніші колірні моделі, які використовуються в індустрії дизайну:

- RGB (червоний, зелений, синій);
- HSL (відтінок, насиченість, яскравість);
- CMYK (блакитний, пурпуровий, жовтий, чорний);

1.5.1 RGB

Модель RGB(Red, Green, Blue) використовується під час роботи з дизайнами на основі цифрових екранів, наприклад, на екрані комп'ютера чи телефону. У колірній моделі RGB кожному з основних кольорів, червоному, зеленому та синьому, призначається значення від 0 до 255, де 0 — темний, а 255 — яскравий. Перерахувавши три значення для червоного, зеленого та синього люмінофорів, ви можете вказати точний колір, який буде змішано(див. рис. 3)

Колірна модель RGB — це система адитивних кольорів, що означає, що кольори стають світлішими при змішуванні. Коли кожен компонент світла змішується, комбінація стає новим кольором. Червоний, зелений і синій є трьома основними адитивами. Ви можете створити будь-який колір в межах обмежень пристрою, використовуючи різні комбінації адитивних первинних елементів. Коли ви змішаєте всі три разом у збалансованих кількостях, ви отримаєте білий колір.

Телевізійні екрани та монітори комп'ютерів створюють колір, вмикаючи первинні червоні, зелені та сині кольори в кожному пікселі. Змінюючи кожен із основних червоних, зелених та синіх кольорів у пікселі на різну яскравість, монітор створює близько 16,7 мільйонів унікальних кольорів.

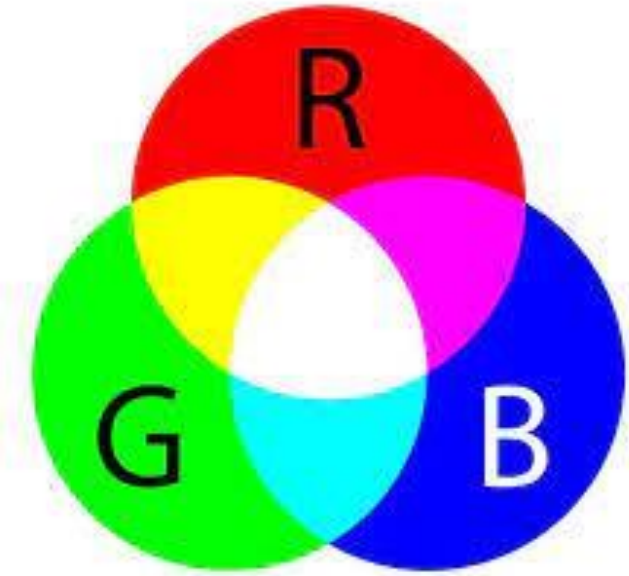

Рис. 1.3 Колірна модель RGB [6]

1.5.2 CMYK

Колірна модель CMYK описує кольори на основі їхнього відсотка блакитного, пурпурного, жовтого та чорного. Багато комп'ютерних принтерів і традиційних «чотирьох кольорових» друкарських машин використовують модель CMYK. Можна рахувати, що ця колірна модель обернена до RGB.

Теоретично ви можете змішати будь-який світловідбиваючий колір, змішавши лише блакитний, пурпурний і жовтий. Однак у реальному світі чорнила, які використовують принтери, не ідеальні. Це стає найбільш очевидним, коли ви змішаєте всі три однаково, щоб отримати чорний колір. Отриманий колір мутно-коричневий, оскільки основні кольори перекриваються і не повністю віднімають світло при змішуванні.

CMYK — це субтрактивна колірна модель, що означає, що кольори стають темнішими при змішуванні. Кожна із змішаних фарб або чорнила поглинає різні компоненти світла. Якщо змішати правильне поєднання фарб, всі компоненти світла поглинаються, і в результаті виходить майже чорний колір.

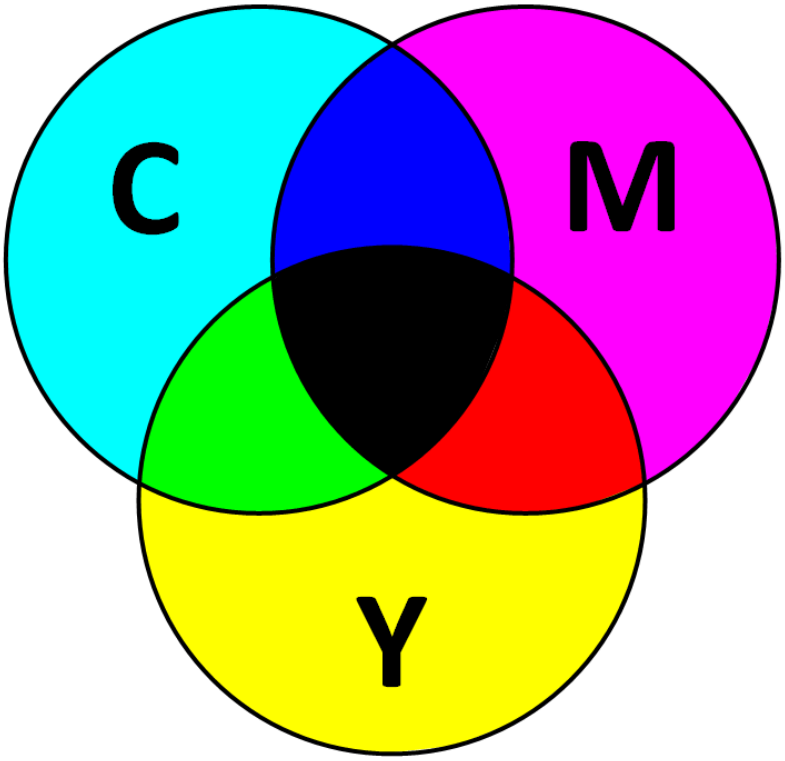

Рис. 1.4 Колірна модель CMYK [7]

1.5.3 HSL

Модель HSL (Hue, Saturation, Lightness) дуже схожа на колірну модель RGB. Насправді, коли вони виражені математично, вони ідентичні. Різниця полягає в тому, як кольори виражаються чисельно.

Відтінок визначає, який це основний колір. Червоний, зелений, синій, жовтий, помаранчевий тощо – різні відтінки. Насиченість і яскравість розповідають більше про варіації цих основних кольорів. Насиченість - це "чистота" кольору, тобто, скільки доповнюваного кольору змішано.

Нарешті, яскравість відноситься до "білості" кольору. Інші моделі, пов’язані з моделлю HSL, — це моделі HSB (відтінок, насиченість, яскравість) і HSI (відтінок, насиченість, інтенсивність). Усі ці терміни схожі, але не взаємозамінні.

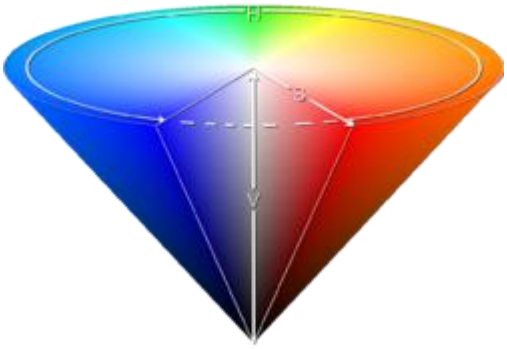

Рис. 1.5 Колірна модель HSL [7]

1.6 Суб’єктивні властивості кольору

Існують світлові хвилі різної довжини, які взаємодіють одна з одною, що поглинаються поверхнями предметів і відбиваються від них, й існує спостерігач, очі якого вловлюють ці хвилі та передають сигнал мозку, який перетворює отриманий сигнал у колірне відчуття.

Людина не вимірює кольори, а реагує на них емоційно, присвоює тонам і відтінкам якості, якими вони насправді не володіють (наприклад, тепло, легкість, дзвінкість тощо). Так з'являються властивості кольору, зумовлені сприйняттям, тобто суб'єктивні (рис. 1).

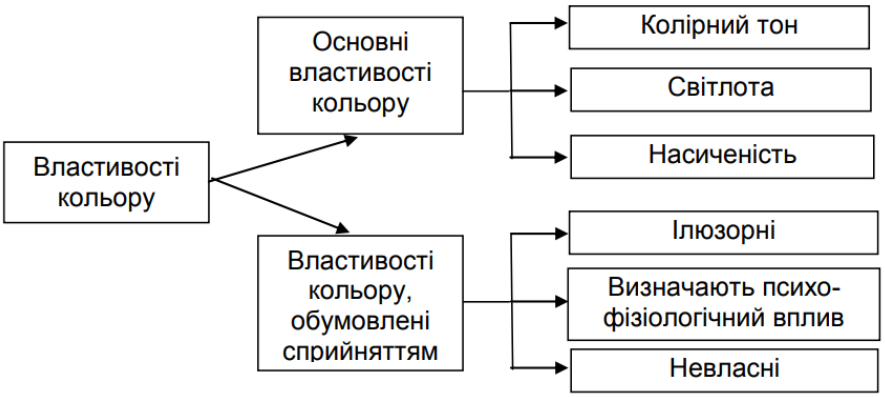

Рис. 1.6.Діаграма властивостей кольору

Основні властивості кольорів можуть бути виражені в числах за допомогою вищезгаданих колірних моделей. Вони піддаються вимірюванню за допомогою спеціальних приладів.

Можливість точного вимірювання та чисельного вираження кольору має велике значення під час підбору кольору в поліграфії, кольоруванні, дизайні, ілюструванні тощо. Через можливість вимірювання та кількісного вираження основні кольори можуть бути змодельовані на комп'ютері (колірні моделі HSL і HSB).

Кольори можна точно порівнювати між собою за основними властивостями. До основних властивостей кольору відносяться колірний тон, світлота (яскравість) і насиченість.

1.6.1 Колірний тон

Колірний тон(кольоровість) – це колірне відчуття, назва кольору та його відтінків: жовтий, синій, червоний, лимонний, коричневий, хакі і так далі.

Тон обумовлений довжиною хвилі світлового випромінювання. Колірний тон є первинним елементом колірної композиції. Він слугує вихідним матеріалом для подальших перетворень на основі яскравості, чистоти, фактури, насиченості. Він володіє композиційними можливостями. Наприклад, оперування зміною відтінків тону дозволяє вплинути на емоційну виразність колірної композиції. Природною шкалою колірних тонів слугує спектр сонячного світла, в якому розрізняють близько 130 тонів.

1.6.2 Яскравість

Світлота, або ж яскравість – це властивість, що виражає близькість ахроматичних і хроматичних кольорів до білого або чорного. З позиції фізики кольору, це величина, що характеризує щільність світлового потоку, відбитого

пофарбованим предметом у напрямку спостерігача. Чим більше світла відбилося, тим світліше колір поверхні об'єкта.

Яскравість – єдина характеристика ахроматичних кольорів. Максимальну світлоту з усіх можливих має ідеально біла поверхня, мінімальну – ідеально чорна.

За світлотою можна порівнювати будь-які кольори – ахроматичні з ахроматичними, хроматичні з хроматичними, ахроматичні з хроматичними. За світлотою розрізняються навіть спектральні тони. Так, наприклад, найсвітліші є жовті, найтемніші є сині та фіолетові.

Шкала світлоти – це рівноступний ахроматичний ряд від білого до чорного з різною кількістю сірих відтінків між ними. Людське око розрізняє близько 300 градацій світлоти. Шкала, що застосовується в комп'ютерній графіці, – 256 тонів.

1.6.3 Насиченість

Насиченість – це ступінь відмінності хроматичного кольору від рівного за світлотою ахроматичного. Зміна кольору в природі, пов'язана з впливом на нього зовнішнього середовища, відбувається переважно за всіма трьома ознаками, тому підбирати той чи інший колір треба і за світлотою, і за колірним тоном, і за насиченістю. Невірно знайдена одна з трьох ознак тягне за собою порушення колірної характеристики об'єкта. Насиченість є ступенем вираженості колірного тону, кількістю кольору в фарбі. Це показник сили та чистоти кольору.

Найбільш насичені кольори переважно використовуються для виділення акцентів. Насиченість тону зменшується від додавання до нього ахроматичного кольору (білого, чорного або сірого).

Так, наприклад:

- під час додавання білого та більш світлого сірого зменшується насиченість і збільшується яскравість.

- під час додавання чорного та більш темного сірого зменшується насиченість і яскравість.
- під час додавання рівного сірого зменшується насиченість.
- під час змішування двох хроматичних кольорів у більшості випадків насиченість отриманого кольору менше насиченості вихідних тонів.

Насиченість іноді плутають з інтенсивністю. Найбільш інтенсивнішими є чисті(спектральні) тони. Дуже світлі та темні тони виглядають не надто інтенсивними навіть за високої насиченості.

Поняттям близьким до насиченості є чистота кольору. Чистота кольору – це близькість кольору до спектрального. Чистота спектральних кольорів приймається за одиницю.

1.7 Представлення зображень у ЕОМ

Зображення може бути представлене в пам'яті ЕОМ двома принципово різними способами і отримано два різних типи зображення: растрове і векторне[5].

Растрова графіка складається з набору крихітних пікселів однакового розміру, які розташовані у двовимірній сітці, що складається з стовпців і рядків.

Векторна в свою чергу складається з об’єктів, що можна описати математичними функціями у дво- або три- розмірному просторі. При збільшені масштабу зображення, її якість не зменшується. Це корисно для емблем та логотипів, де потрібно плавне зображення із можливістю збільшення розміру.

В даному проекті я вирішив оперувати лише растровим зображенням, так як фотографічні знімки більш розповсюджені, а векторні зображення можна легко експортувати у растрове зображення з певною роздільною здатністю.

Кожен піксель містить один або кілька бітів інформації, залежно від ступеня деталізації зображення. Наприклад, чорно-біле зображення містить лише один біт на піксель (двійковий біт може бути в одному з двох станів; таким чином, один біт може представляти білий або чорний колір), а зображення із тінями і кольором зазвичай містить 24 біти інформації на піксель — з 2 у 24 степені, або більше 16 мільйонів можливих станів на піксель. Відомий як "truecolor", 24-бітовий колір може реалістично відображати кольорові зображення.

Кількість бітів, що зберігаються в кожному пікселі, називається глибиною кольору. Роздільна здатність, впливає на те, скільки деталей можна відобразити на зображенні. Роздільна здатність часто виражається як кількість пікселів у стовпці, помножена на кількість пікселів у рядку (наприклад, 800 на 600).

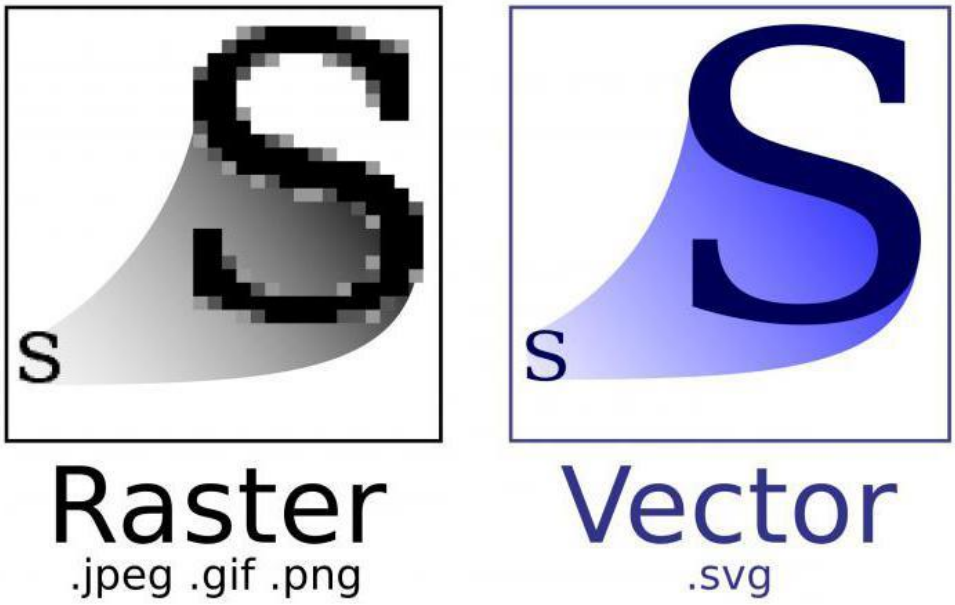

Рис. 1.7 Растрова і векторна графіки [8]

Детальні зображення часто призводять до великих розмірів файлів, хоча розміром файлу можна керувати за допомогою стиснення даних. Стиснення може бути як із втратами (це означає, що деякі дані відкидаються), так і без

втрат (дані не втрачаються). Популярні формати растрових файлів включають GIF (формат обміну графікою) і JPEG (назва компанії-розробника), які є форматами з невеликими втратами даних, а також BMP (бітовий масив) і TIFF (формат файлу з тегами), які не мають втрат.

Вибір правильного способу стиснення буде однією з ключових проблем дипломної роботи, адже для швидкої та злагодженої роботи потрібно оброблювати чималий масив даних, виконувати побітові операції для фільтрації правильних даних і т. д.

У мові програмування Java зображення можна подати як масив пікселей, де кожен піксель представляє собою колір у певній позиції, а розміри це протяжність пікселей по довжині та висоті. Найважливішим класом зображення для представлення таких зображень є BufferedImage клас з пакету java.awt.image.

Java 2D API [5] зберігає вміст таких зображень у пам'яті, щоб до них можна було отримати прямий доступ і як наслідок швидше оперувати. Але таке рішення може бути неоптимальним через надто велике використання пам'яті при обробці декількох зображень. Тому для оптимізації процесів порівняння зображень, потрібно скласти характеристику певного зображення і оперувати ним безпосередньо. Для таких цілей нам буде зручно використати HSB в якості колірної моделі. Ця модель дозволяє представити наступну характеристику пікселю:

- Відтінок — колір у спектральній палітрі задається цілим числом і задається цілим числом від 0 до 360° або ж у форматі від 0 до 100. Мінімальне і максимальне значення відповідає червоному кольору, а проміжні – іншим кольорам спектра.
- Насиченість — частка білого кольору, доданого до вибраного відтінку. Задається значення насиченості у відсотках від 0 до 100. При мінімальній насиченості будь-який відтінок кольору стає сірим.

- Яскравість — визначається домішкою чорного кольору до вибраного відтінку. Задається значення яскравості у відсотках від 0 до 100. Будь-який відтінок при мінімальній яскравості стає чорним.

Висновок

В даному розділі, проаналізовано тему світла та кольору, а також їх репрезентація у цифровому зображенні. Підсумовуючи, варто зазначити, що колір – це наше суб’єктивне бачення світла, що знаходиться у видимому діапазоні частот.

У електронно-обчислювальних машинах растрове зображення подається як масив дуже малих точок, які називаються пікселями. Кожна така точка може мати певний колір, яскравість, насичення та інші властивості, і як наслідок її можна перетворити у числові характеристики. Це досягається із використанням певних колірних моделей – схем представлення кольору як сукупність параметрів. Було порівняно найпопулярніші їх версії – RGB, CMYK та HSL. На мою думку, найзручніша у використанні остання – за її допомогою можна легко відділяти колір окремим параметром, аналогічна ситуація з яскравістю.

Також було досліджено представлення зображення у мові програмування Java, за допомогою якої зображення будь якого формату перетворюється у масив пікселів, котрі легко трансформуються із однієї колірної моделі в іншу.

РОЗДІЛ 2
ЗАСОБИ РЕАЛІЗАЦІЇ ВЕБ-СЕРВІСУ

2.1 Поняття веб-сервісу

Веб-сервіс – це вид електронних послуг які надаються в мережі Інтернет. Їх існує неймовірно велика кількість і кожна з яких має своє призначення. В даному проекті я буду реалізовувати Веб-сервіс на основі мови програмування Java. Так як ця мова дуже популярна для рішення саме таких задач, у спільноті Java існує велика кількість інструментів, які допоможуть нам у написанні програми.

2.2 Мікросервісна архітектура

Мікросервіси — це популярна архітектура програмного забезпечення, яка розбиває монолітні системи. Програми будуються як набір слабо пов’язаних служб. Кожен мікросервіс відповідає за одну функцію. Вони взаємодіють один з одним за допомогою протоколів зв'язку, таких як HTTP і TCP або ж RPC.

В останні роки мікросервіси набули величезної популярності. В основному тому, що вони мають переваги, які надзвичайно корисні в епоху контейнеризації та хмарних обчислень. Можна розробляти та розгортати кожен мікросервіс на іншій платформі, використовуючи різні мови програмування та інструменти розробника. Мікросервіси використовують API та протоколи зв’язку для взаємодії один з одним, але в іншому випадку

вони не залежать один від одного.

Найбільшим плюсом архітектури мікросервісів є те, що команди можуть розробляти, підтримувати та розгортати кожну мікросервісу незалежно.

Такий вид єдиної відповідальності також дає інші переваги. Програми, що складаються з мікросервісів, краще масштабуються, оскільки ви можете масштабувати їх окремо, коли це необхідно.

Якщо додаток представляє собою набір окремих підпрограм, кожна з цих програм виконує лише свою роботу, і зміни в одному місці програми не повинні стосуватися іншої частини.

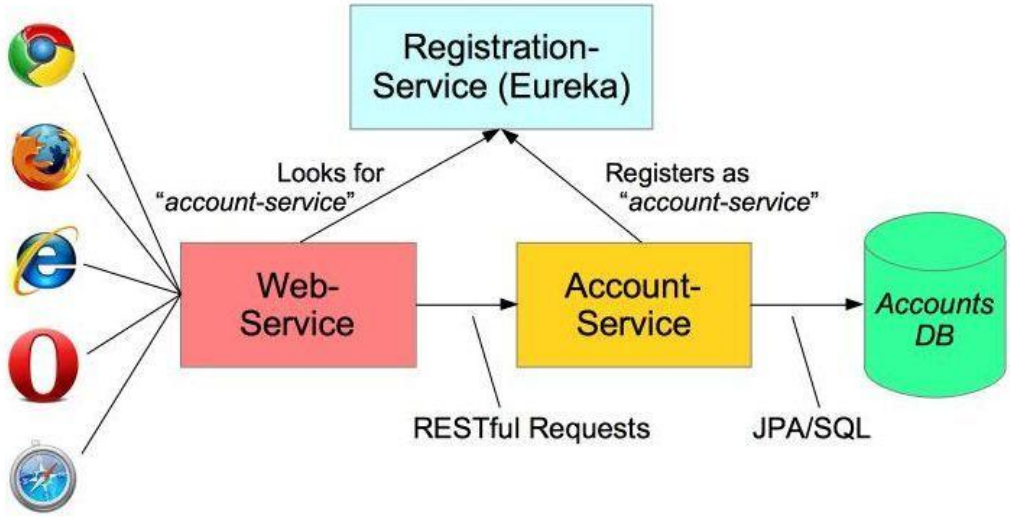

Рис. 2.1 Приклад додатку із мікросервісною архітектурою[9]

Мікросервіси також скорочують час виходу на ринок і прискорюють конвеєр CI/CD. Це також означає більше спритності до написання та тестування коду. Крім того, ізольовані служби мають кращу стійкість до відмов. Зрештою, легше підтримувати та налагоджувати легкий мікросервіс, ніж складну програму.

2.3 Інструменти для розробки. Фреймворки

Для написання веб-сервісу, буде доцільно використати різні сервіси для рішення різних задач. Інструменти, які я використовую, дуже популярні та мають добре підготовлену документацію, що спрощує та пришвидшує написання кодової бази і, як результат, перші результати.

2.3.1 Vue.js

Vue — це JavaScript фреймворк із відкритим вихідним кодом для створення інтерфейсів користувача та невеликих клієнтських програм. Він написаний на архітектурному стилі, який зосереджений на простій візуалізації та композиції різних невеликих компонентів.

Основна бібліотека зосереджена лише на шарі перегляду(view), хоча можна по різному його використовувати. Цей фреймворк дозволяє швидко та якісно побудувати front-end частину веб-застосунку завдяки наступним властивостям:

- Невеликий розмір самої бібліотеки
- Продуктивність
- Архітектура на основі компонентів
- Можливість багаторазового використання
- Гарно написана документація

2.3.2 Spring Framework

Spring — один з найпопулярніших фреймворків з відкритим кодом для розробки корпоративних додатків. Він забезпечує комплексну підтримку інфраструктури для розробки додатків корпоративного рівня Java.

Він має численні переваги, які дозволяють йому мати велику долю використання у розробці корпоративних додатків.

Основні його переваги це:

- Легкість - Spring Framework має легку вагу щодо розміру та прозорості у використанні.
- Інверсія керування (IOC) - у Spring Framework слабка зв’язність досягається за допомогою інверсії керування. Так програма ділиться на сукупність слабо пов’язаних сервісів, що дозволяє легко підтримувати та розширювати програму.
- Аспектно-орієнтоване програмування (AOP) - відокремлюючи бізнес-логіку додатків від системних служб, Spring Framework підтримує аспектно-орієнтоване програмування.
- Контейнеризація - Spring Framework створює і керує життєвим циклом і конфігурацією об'єктів програми. На плечі розробника лягає робота лише композиції залежностей об’єктів.
- MVC Framework — це фреймворк веб-додатків MVC. Цю структуру можна налаштувати за допомогою інтерфейсів і вміщує кілька технологій перегляду.
- Управління транзакціями - для управління транзакціями фреймворк Spring надає загальний рівень абстракції. Він не прив’язаний до середовища J2EE і може використовуватися в середовищах без контейнерів.
- Обробка винятків JDBC - рівень абстракції JDBC у Spring Framework пропонує ієрархію винятків, яка спрощує стратегію обробки помилок.

Фреймворк Spring використовує найкращі практики, які були доведені роками в кількох програмах і формалізовані як шаблони проектування.

2.3.3 PostgreSQL

PostgreSQL, використовується як реляційна база даних для зберігання товарів, їх опис, відношення користувачів до товарів, в той час MongoDB зберігає картинки, які відносяться до певного типу товару. PostgreSQL — це потужна об'єктно-реляційна система баз даних з відкритим вихідним кодом з більш ніж 30-річною активною розробкою, завдяки якій вона заслужила міцну репутацію за надійність, стійкість функцій і продуктивність.

В офіційній документації можна знайти велику кількість інформації, яка описує, як встановити та використовувати PostgreSQL. Спільнота PostgreSQL пропонує багато корисних місць, щоб ознайомитися з технологією, дізнатися, як вона працює, і знайти можливості для кар'єрного зростання. Звертайтеся до спільноти тут.

Переваги:

- Висока відмовостійкість. Вона відповідає властивостям атомарності, узгодженості, ізольованості та довговічності (ACID) для транзакцій баз даних. Крім того, PostgreSQL підтримує кілька мов для різних тригерів, атрибутів зовнішніх ключів, об'єднань та процедур, що зберігаються.
- Вихідний код PostgreSQL доступний під ліцензією з відкритим вихідним кодом та чудово написаною документацією. Можна легко використовувати, змінювати та впроваджувати нові функції якщо це потрібно розробнику.
- PostgreSQL підтримує географічні об'єкти, тому його можна використовувати як сховище даних для геослужб, що базуються на місцезнаходження, і геоінформаційних систем.
- База даних має у собі в комплекті графічну платформу для адміністрування та розробки баз даних.

2.3.4 Liquibase

Liquibase — це незалежна від бази даних бібліотека з відкритим вихідним кодом для відстеження, керування та застосування змін схеми бази даних. Він був започаткований у 2006 році, щоб полегшити відстеження змін у базі даних, особливо в умовах гнучкої розробки програмного забезпечення .

Liquibase дозволяє розробникам писати сценарії автоматичної міграції для баз даних. Liquibase гарантує, що скрипти виконуються лише один раз і не змінюються знову. Скрипти оновлення також включені у версію і, таким чином, також доступні для системи CI/CD. Це дозволить системі CI тестувати нові сценарії з попереднім випуском програмного забезпечення для забезпечення сумісності.

2.3.5 Azure Storage

Azure Storage — це служба хмарного сховища, керована Microsoft, яка забезпечує швидкодоступне, міцне, масштабоване та надлишкове сховище. Вона пропонує швидкодоступне, масштабоване, міцне та безпечне сховище для різноманітних об’єктів даних у хмарі.

Об’єкти даних Azure Storage доступні з будь-якої точки світу через HTTP або HTTPS через REST API. Azure Storage також пропонує клієнтські бібліотеки для розробників, які створюють додатки або служби за допомогою .NET, Java, Python, JavaScript, C++ і Go.

Особливості Azure Storage:

- Довговічність і висока доступність: збережені дані реплікуються та зберігаються в різних географічних місцях.
- Масштабованість: сховище має масову масштабування залежно від вимог. Дані автоматично збільшуються, щоб задовольнити будь-які пікові потреби.
- Безпека: доступ до будь-якої інформації зловмисника з вашого

сховища не є легким завданням, а отже, ваші дані захищені. Сховище Azure використовує модель спільного ключа для автентифікації користувача. Використання підпису спільного доступу (SAS) може обмежити доступ до даних.

- Доступність: ви можете отримати доступ до своїх даних з будь-якого місця через HTTP або HTTPS. Ви можете написати свій код на Azure PowerShell або в Azure CLI. Провідник сховища Azure і портал Azure надають вам простий спосіб роботи з даними.

Сховище Azure Blob — це рішення Microsoft для зберігання двійкових об’єктів у хмарі. Сховище BLOB оптимізовано для зберігання величезної кількості неструктурованих даних, таких як текстові або графічні дані.

Цей сервіс ідеально підходить для зберігання великої кількості зображень користувача.

Висновок

У другому розділі були проаналізовані технології для написання майбутнього веб-сервісу. Вони мають бути нові, щоб відповідати вимогам сучасних конкуруючих веб-сервісів.

Для того щоб була приваблива інтернет-сторінка було вибрано Vue.js через його можливості та простоту у використанні.

При пошуку інструментів для зберігання двійкових файлів, як наприклад, фото, вибір впав на хмарне сховище Microsoft Azure Storage. Воно легко масштабується та має великий можливий обсяг збережених даних.

Для побудови серверної частини я вибрав Spring Framework через його ефективність, велику кількість додаткових бібліотек, з якими можна побудувати швидку та зручну у розробці програму.

У якості бази даних було вибрано PostgreSQL, через його високу відмовостійкість та можливість адміністрування у графічному редакторі.

Для написання міграцій я буду використовувати Liquibase, тому що він добре зінтегрований із бібліотекою Spring та базою данних PostgreSQL.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ВЕБ-СЕРВІСУ. СОРТУВАННЯ ПІКСЕЛЕЙ ПО ЇХ ХАРАКТЕРИСТИКАМ

3.1 Загальний опис системи

Веб-сервіс представляє собою сайт, на якому користувач може шукати потрібно йому зображення і також викладати свої фото для подальшого збереження у будь який момент часу.

У доступному функціоналі користувача повинні бути кнопки пошуку потрібного кольору, за яким буде відбуватись пошук, кнопки вибору схеми сортування зображень за яскравістю.

Користувач може зареєструватись як новий, в такому разі у кожного користувача буде своя умовна папка із файлами, видима лише йому. Користувач має бути в змозі вільно керувати викладеними файлами, змінювати назви, видаляти та завантажувати нові фото.

3.2 Схема мікросервісної архітектури додатку

Front-end – це візуальна частина веб-сайту. Користувач може здійснювати певний набір операцій у своєму браузері. Наприклад, вивантаження нових фото, вибір способу сортування зображень, вхід у власний аккаунт.

Далі дані передаються на серверну частину(Backend), де відбувається обробка файлу, складання його характеристики, пошук потрібних зображень у базі даних PostgreSQL або посилання на збереження у хмарне сховище Azure.

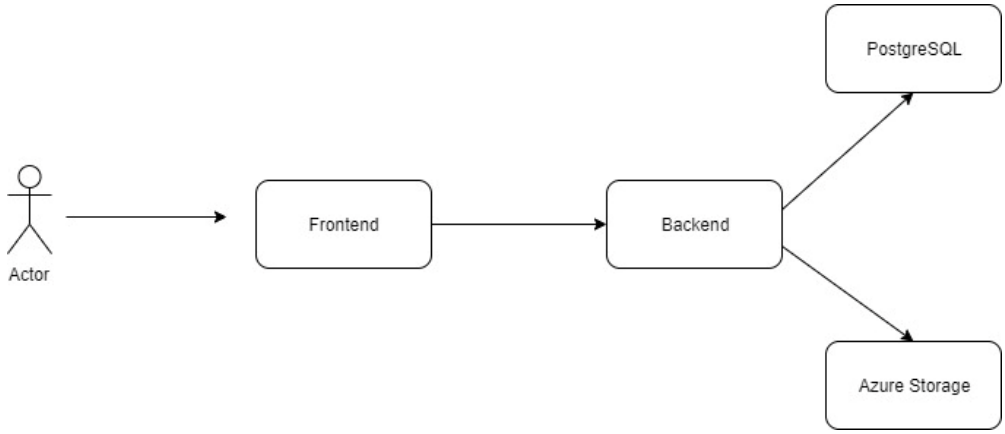

Рис. 3.1 Блок-схема компонентів веб-сервісу

3.3 Створення нового проекту

Для створення нового проекту, я скористався інструментом, що розроблений спеціально під програми на Spring Framework — Spring Initializr [10]. Він дає змогу створити новий проект уже з готовим архетипом та підключеним набором залежностей. Список залежностей у веб-сервісі я визначив наступний:

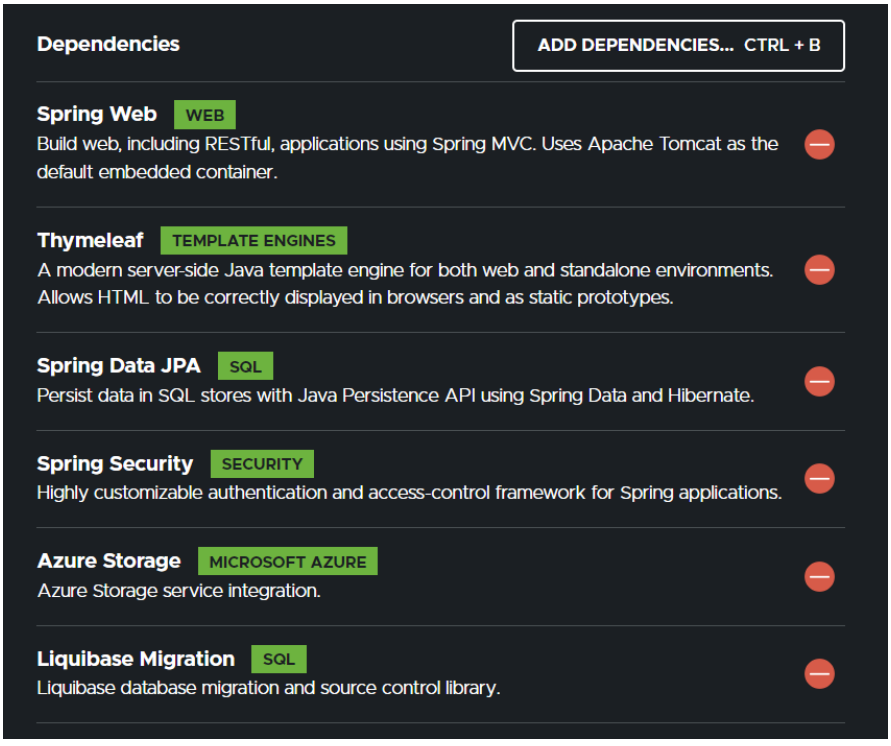

Рис. 3.2 Інтерфейс Spring Initilizr із списком залежностей

Список залежностей наступний:

- Spring Web – для побудови Web-додатку за архітектурою REST, що дозволить сервісу бути зручним у використанні та в майбутньому перевикористання його на інших платформах.
- Thymeleaf – шаблонізатор для побудови файлів типу .html.
- Spring Data JPA – дозволяє зберігати дані в SQL-сховищах та легко оперувати записами у таблицях як Java-об’єктами.
- Spring Security – гнучка система аутентифікації та контролю доступу для додатків Spring. Допоможе з авторизацією користувача.
- Azure Storage – набір конфігурацій із Azure Storage у напів готовому вигляді для керування запитами на сховище.
- Liquibase Migrations – бібліотека із системою міграцій бази даних.

У своєму проекті я буду використовувати мову програмування Java версії 17, а також збирач проектів Maven для швидкого та зручного методу організації залежностей.

3.4 Початкова побудова бази даних

Для побудови схеми бази даних було використано Liquibase міграції. Це невеликі файли із змінами у базі даних, які при кожному запуску програми виконуються та записуються у спеціальний журнал DATABASECHANGELOG. Там зберігається інформація щодо кожної виконаної міграція, така як час виконання, чек-сума, яка служить ідентифікатором незмінності міграції.

Liquibase розгортає лише нові, ще не виконані зміни. Це допомагає поступово змінювати структуру бази даних маленькими кроками, без ризику побудови неправильної схеми.

Перша створена сутність у базі даних це Користувач(user). Для простоти реалізації в таблиці буде зберігатись лише:

- username – назва акаунту користувача, а також первинний ключ.
- enc_pass – закодований пароль користувача за допомогою Bcrypt алгоритму.
- profile_photo_url – посилання на фото профілю користувача.

Так як всі фото будуть зберігатись в хмарному сховищі, у базі даних зберігати такі великі масиви даних як зображення, недоцільно. Натомість, в таблиці image можна зберігати характеристику зображення. Вона буде у стиснутому двійковому вигляді. В реляційних базах даних для цього є спеціальний тип данних — bytea.

Окрім піксельної характеристики зображення, у image буде зберігатись також:

- original_name – назва файлу.
- created_date – дата та час створення.
- User – зовнішній ключ на таблицю юзера.

Як результат, маємо наступну схему таблиць бази даних:

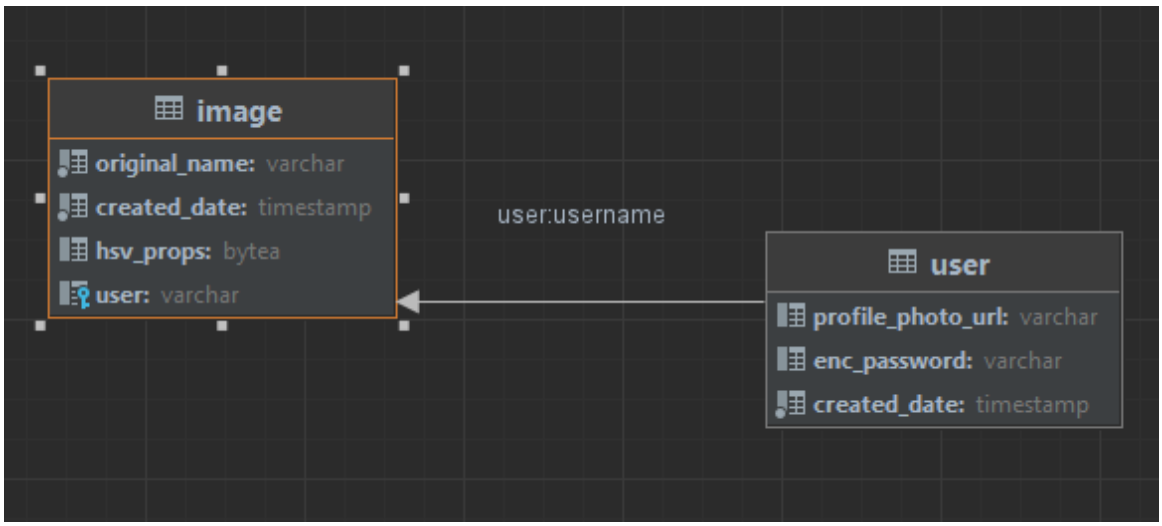

Рис. 3.3 Схема бази даних веб-сервісу

Щоб під’єднатись до БД, потрібно вказати адресу серверу та назву самої бази даних.

3.5 Реалізація сутностей

Наступним кроком буде визначення сутностей реляційної бази даних для їх репрезентації у Java-програмі. Я своєму проєкті я використовував залежніть під назвою Spring Data JPA. Ця бібліотека дає використовувати базовий набір імплементацій концепцій ORM — модель об’єктно-реляційного відображення - для зручної репрезентації записів таблиці у вигляді Java об’эктів.[JPA]

Spring Data JPA бере на себе реалізацію операцій з БД, використовуючи уже існуючі механізми.

3.5.1 Image

Щоб репрезентувати сутність “Image”, потрібно вказати такі анотації як «@Table» та «@Column».

@Table використовується для задання назви таблиці та схеми у БД.

@Column – анотація для указання назву стовпеця, що відповідає полю об’єкта.

Щоб об’явити первинний ключ потрібно вказати анотацію @Id, де у властивостях зазначити його стратегію генерації. Найчастіше використовують IDENTITY, тобто база даних буде сама видавати число, яке кожен раз збільшується на 1.

Окрім первинного ключа, дати створення та назви файлу, окремим полем буде HsvImageProperties — вкладений об’єкт, що буде серіалізуватись(перетворюватись у набір байтів), та зберігатись у такому ж вигляді. Проте у коді він буде звичайний об’єктом із набором властивостей пікселя по HSB-моделі(hue, saturation, brightness). Всі ці властивості виражаються у типі float та мають значення від 0 до 1.

Для можливості перетворити об’єкт в набір байтів, потрібно щоб він імплементував інтерфейс-маркер Serializable. Віртуальна машина Java розрізняє такі об’єкти, і без проблем його серіалізують.

Для базових операцій об’єктів Image з базою даних варто лише зробити новий інтерфейс, основні операції якого будуть уже нам доступні. Інтерфейс ImageRepository успадковується від JpaRepository<Integer, Image>. Так Spring Framework шукає спадкоємців цього інтерфейсу і на старті програми зробить анонімний об’єкт із реалізацією даного інтерфейсу.

3.5.2 User

Сутність користувача створюється аналогічним до Image способом.

Поле login виступає псевдонімом користувача і водночас первинним ключем. В базі даних можливий лише один користувач із таким псевдонімом. Це важлива концепція для реалізації назв файлів користувача.Використовуючи всі ті ж самі анотації, але з іншими назвами стовпців та таблиці, створюємо сутність User та репозиторій до нього.

3.6 Створення Контролерів

Важливо сказати про сам архітектурний паттерн MVC. Модель MVC є шаблоном програмування, що дозволяє ділити логіку програмної програми на 3 частини. Розшифровка абревіатури досить проста:

M – Model (Модель) . Model отримує дані від контролера, здійснює необхідні операції, а потім передає дані в View;

V – View (Вид, Подання) . View отримує дані від Model із наступним виведенням їх для користувача;

C – Controller (Контролер) . Контролер в MVC - це блок, що обробляє дії користувача і перевіряє отриману інформацію, яка потім передається в Model.

Spring WEB – бібліотека, яка надає швидкий доступ до написання частини, що стосується прийняття та віддача запитів та відповідей із використанням HTTP або HTTPS протоколів, маючи під собою MVC патерн.[WEB]

У цій бібліотеці є два види контролерів: звичайний та REST-контролер. Звичайний повертає View, а саме назву сторінки у форматі html або jsp, в той час як REST лише відповідь у форматі json.

Нам знадобиться обидва варіанта контролерів. У REST ми будемо приймати фото, а у звичайному віддавати сторінку, наповнену різними потрібними данними.

Спочатку створимо клас із назвою ImageController, та позначимо його як @RestController. Об'єкт цього класу буде створений при старті програми, завдяки внутрішній анотації @Component.

В анотації @PostMapping над методом класу ми кажемо, що коли відбуватиметься POST-запит за адресою «/addImage», а ендпоінт може приймати лише MULTI_PART_FORM_DATA об’єкти, то буде виконуватись це тіло методу.

Далі створимо новий клас ColoristController, але тепер позначимо його іншою анотацією – @Controller.

Додамо новий метод із назвою mainPage, котрий буде повертати String – назву сторінки. Spring сам шукає по цій назві сторінку у папці templates і повертає користувачу. Параметри методу будуть об’єкти класу Model та ImageSearchParams.

За замовчуванням веб-сервіс запускається на локальному хості та порті 8080. Тому у випадку GET-запиту, ми добавляємо у об’єкт Моделі список об’єктів Image, які відсортовано за параметрами ImageSearchParams.

3.7 Підключення сховища Azure

Спочатку потрібно створити обліковий запис на порталі Azure[2] та оформити спеціальну підписку для користуванням сховищем.

Azure має широкий набір інструментів доступу до власного контейнеру.

Є авторизація за допомогою AzureId, SAS-токена та пари публічного та приватного ключа. У своїй реалізації, я вибрав SAS-токен через простоту використання та швидкості підключення до власного сервісу. Для початку потрібно вказати у властивостях виданих на порталі Azure рядок для підключення. Він складається із 455 символів, де містяться:

- адреса BLOB-сховища
- адреса файлового сховища
- токен та термін його дії
- публічний підпис токена

Наступним кроком буде реалізація клієнта для доступу до основних операцій збереження та отримання самих зображень. За допомогою доданої раніше бібліотеки Spring Azure Starter Blob Storage, я використовую спеціальний об’єкт класу BlobServiceClient, передаючи у білдер вищезгаданий рядок для підключення.

3.8 Зберігання зображень

Для того щоб відсортувати зображення по кольору, потрібно скласти попіксельну характеристику зображення. Щоб зменшити обсяг вибірки, я вирішив стискати кожне зображення до розмірів. Це допоможе зменшити розмір даних, що будуть зберігатись у базі даних.

Так як зображення у Java я подаю в вигляді об’єкта класа BufferedImage, тому немає різниці у якому форматі воно було, адже у відповідності до специфікації даного класу, він зберігає лише сам масив пікселів, колірну модель та деякі метадані.

Для таких операцій як зміна розміру піксельного шару або зміна якості зображення, я використовував Tumbnails[12], яка легко використовує потоки вводу та виводу даних для таких операцій.

Зберігати бажано завжди в одній і тій самій роздільній затності, щоб дані були рівноправні.

Якщо одне зображення буде у форматі 3840 на 2160 пікселей, а інше 500 на 600, то кількість пікселів одного кольору на першому зображені буде більша, хоча візуально цього кольору менше, ніж на другому.

Втрати, при стисканні є, але вони помірні, і так якась кількість пікселів об’єднуються в один, можна сказати що ми беремо середнє арифметичне із суми цих пікселей.

На основі уже стиснутого об’єкту BufferedImage, я буду кожен піксель переводити із колірної моделі RGB у HSB-властивості. Зробити це можна за допомогою спеціальної формули переведення.[10]

Зберігання зображення має наступні етапи:

1. Користувач, використовуючи веб-сервіс, вибирає функцію збереження файлу, і на серверну частину йде запит уже безпосередньо із самим зображенням.
2. Зображення копіюється та стискається. На його основі складається кольорова характеристика.
3. Оригінал зображенння зберігається у сховище Azure Blob.
4. Сутність разом із характеристикою зберігається у базу даних

Якщо користувач захоче отримати зображення у себе на сторінці, він буде робити запит безпосередньо у сховище Azure. Це зроблено для уникнення зайвого навантаження серверу, адже зображення виликої роздільної здатності погано транспортувати. Azure має надсучасні сервери, тому частину роботи ми передаємо сховищу та браузеру клієнта.

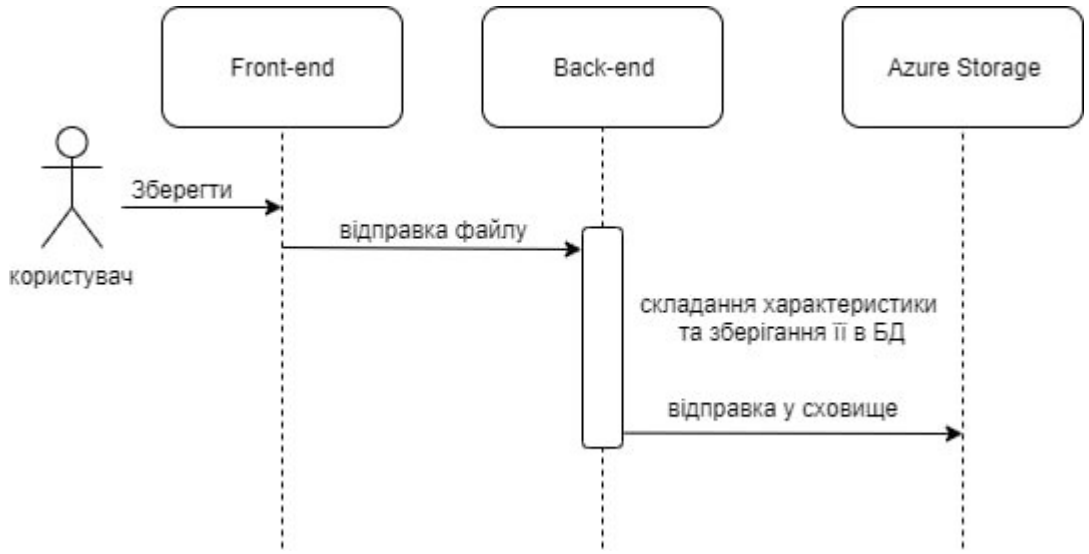

Рис. 3.4 Діаграма послідовностей при збереженні файлу

3.9 Розробка Візуальної частини

Front-end частина розроблюється за допомогою бібліотеки Vue.js та шаблонів стилів Bootstrap. Усе це використовується після опрацювання html сторінки бібліотекою Thymeleaf. Вона дозволяє генерувати html код з підтримкою звичайних конструкцій мов програмування, такі як цикли та умови if-else.

Сторінка зі списком зображення розроблюється з використанням готових CSS стилей з сховища Bootstrap.

Окремої уваги вартий елемент, що дозволить користувачу вибирати колір, яскравість та інші параметри, на основі яких буде відбуватись подальше опрацювання запитів. Такі елементи називаються кнопками вибору кольору.

Для свого проекту я використав vanilla-picker бібліотеку. Цей варіант може настроюватись у відповідності до наших потреб, в тому числі можна задати дії, при виборі самого кольору.

Вибір відтінку зводиться до пересування повзунка до кольору, що нас цікавить.

Для того щоб вибрати потрібну яскравість, потрібно курсором натиснути на потрібний елемент з піксельної дошки властивостей кольору.

Рис. 3.5 Графічний дизайн для кнопки вибору кольору

При натисканні на кнопку «ОК», буде відбуватись новий запит із параметрами у адресі веб-сервісу. Це дозволить візуально порівнювати числові значення не покидаючи головну сторінку.

Також необхідно звичайно додати можливість додавати нові фото. Це легко зробити за допомогою знову ж таких уже готових стилей із Bootstrap. Відбуватись це повинна також окремим запитом без зміни головної сторінки для покращення вражень від користування веб-сервісом.

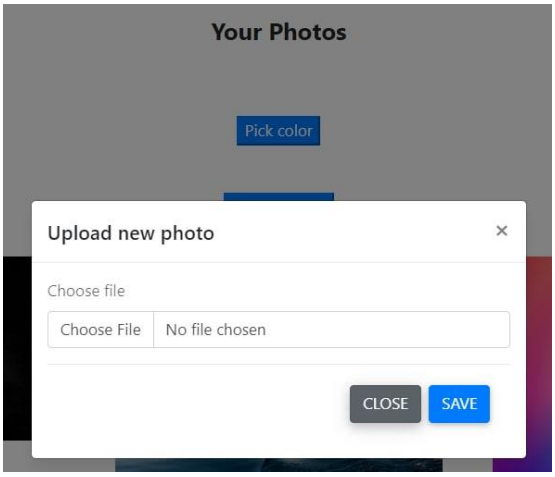

Рис. 3.6 Дизайн кнопки завантаження фото

3.10 Сортування зображень

Сортування зображень може відбуватись наступним чином.

1. Користувач робить запит із певними параметрами(колір, яскравість, насичення, режим).
2. Із бази даних дістаємо останні N зображень користувача.
3. На основі їх колірних характеристик, сортуємо у відповідності до режиму сортування, а саме:
4. У випадку сортування за кольором, ми шукаємо всі пікселі, які входять у діапазон значень пошукового параметру кольору + деякий діапазон, щоб розширити вибірку.
5. В разі сортування за яскравістю, ми сумуємо всі значення яскравості кожного пікселя певного зображення та вираховуємо середнє значення. Те що ближче до шуканого значення.
6. Масив із назв зображень передаємо у об’єкт моделі, де буде формуватись html сторінка, звідки браузер користувача буде запитувати у сховища Azure оригінальне зображення.

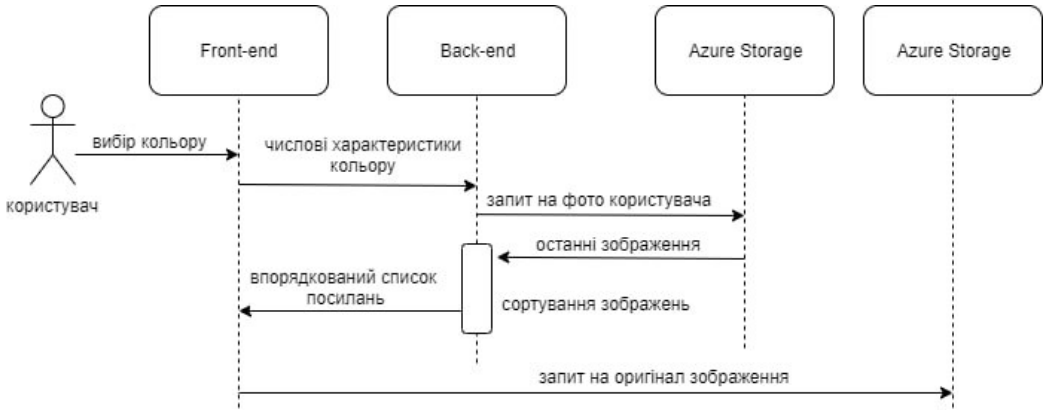

Рис. 3.7 Діаграма послідовностей при сортуванні

Сортування за кольором реалізовано з визначенням двох можливих інтервалів пошуку. Це зроблено для того щоб обробляти випадок коли червоний колір може бути як початковим значенням, так і кінцевим.

У компараторі ми рахуємо ті пікселі, що попали у наш діапазон одного та другого зображення. У кого більша кількість, те зображення буде першим.

3.11 Демонстрація роботи веб-сервісу

Для початку нам потрібен список фото по якому буде відбуватись сортування. Для більшої демонстративності, я вибрав контрастні та різні за колірною гаммою зображення. Для прикладу візьмемо 5 наступних фото.

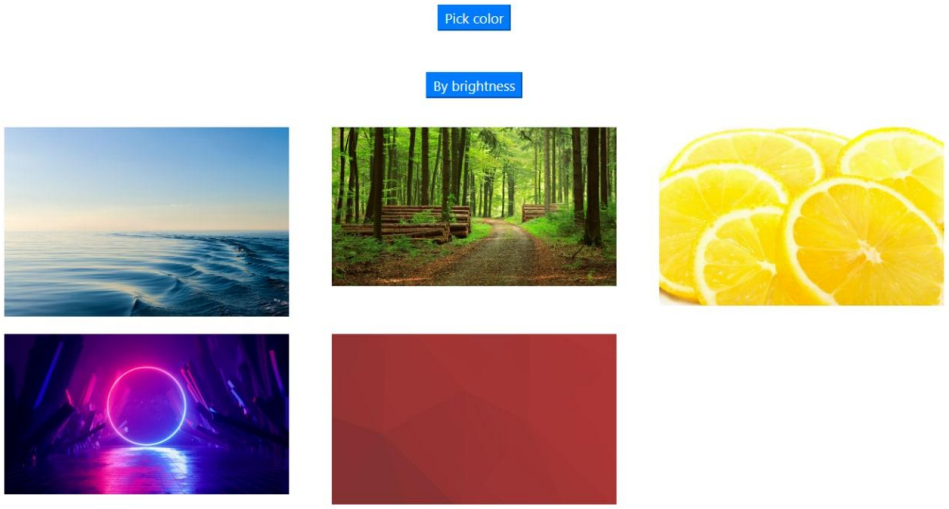

Рис. 3.8 Початкові зображення

При натисканні на кнопку «Pick Color» викликається нове вікно з шкалою відтінків, поле з яскравістю та насиченістю та, окремо, з прозорістю.

До прикладу, нехай буде вибрано жовтий колір. Насиченість середня, а отже колір буде найчистіший. Яскравість також середня, тому результат буде майже чистим жовтим кольором.

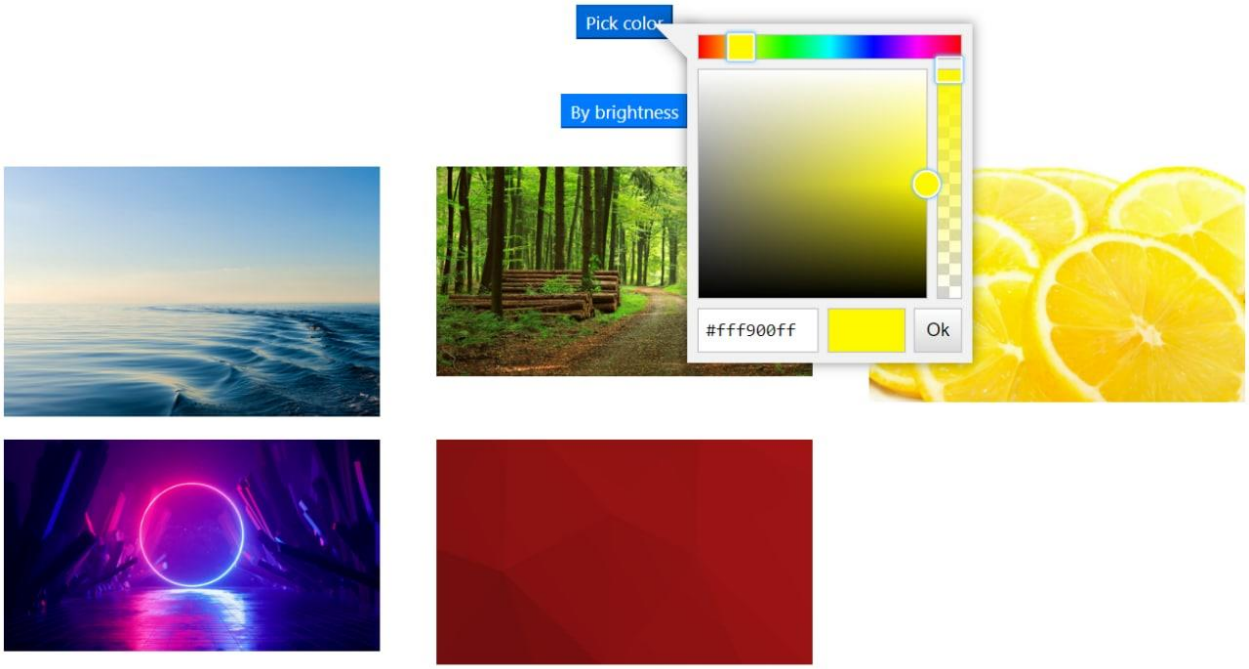

Рис. 3.9 Вибір кольору на веб-сайті

Натискаємо «ОК» і отримуємо нову послідовність картинок.

Так як ми вибрали жовтий колір, то програма розпізнала зображення, у якої жовтого кольору було найбільше — фотографія лимонів:

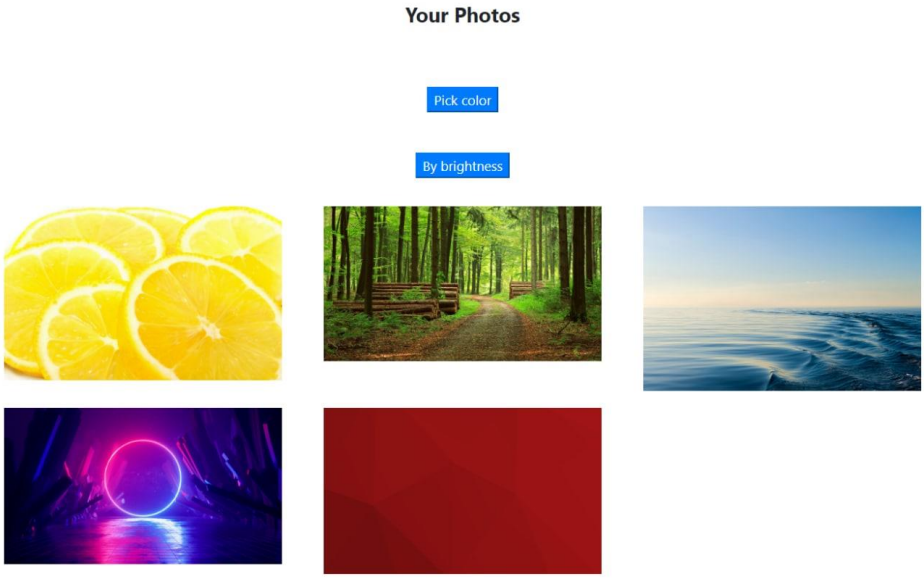

Рис. 3.10 Відсортована послідовність зображень

Так само із яскравістю, вибір кольору уже не грає роль, тому коли ми вибраємо сортування за яскравістю, нам досить визначитись із значенням яскравості. До прикладу, нехай яскравість буде на ближче до третини, тобто вибірка фото буде від темніших по зростанню яскравості.

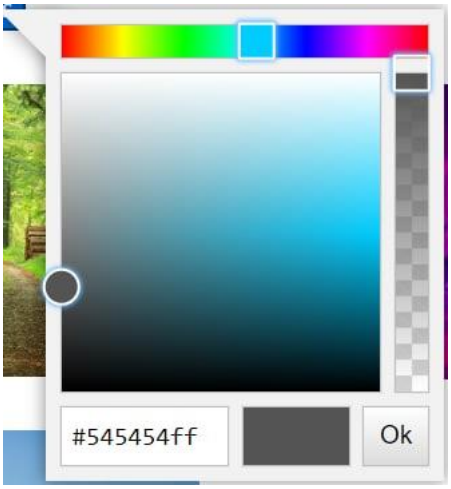

Рис. 3.11 Вибір яскравості

Як наслідок, порядок фото став іншим:

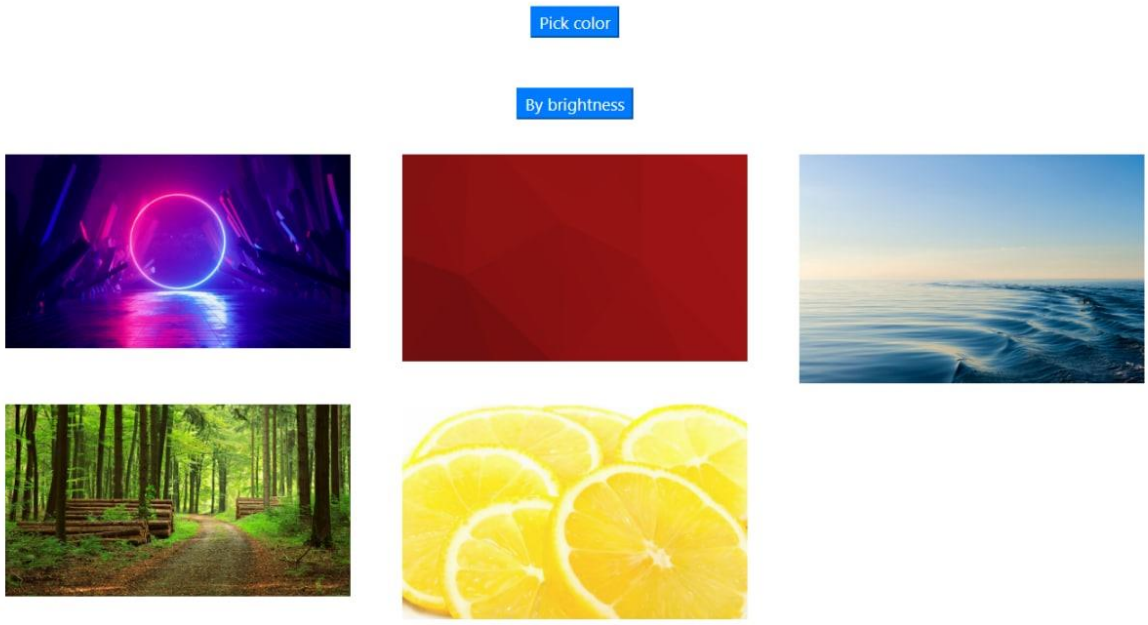

Рис. 3.12 Порядок сортування за заданою яскравістю

Висновок

В цьому розділі було розроблено веб-сервіс із використанням технологій хмарного сховища та сучасних фреймоворків.

Було проаналізовано операції з пікселями, та написані робочі прототипи переведення характеристики зображення із RGB моделі у HSV. У такий спосіб зображення можна легко відрізняти по відтінку, а сама модель більше нагадує саме наше сприйняття кольору.

Також досліджено можливі способи сортування зображень з використанням раніше обчисленої попіксельної характеристики, за яскравістю та кольором.

Розроблено графічну частину додатку, де користувач зможе інтуїтивно користуватись інтерфейсом веб-сервісу. Було продемонстровано роботу веб-сервісу із сукупності всіх розроблених модулів програми.

ВИСНОВКИ

В даній роботі, я дослідив тему світла та кольору, а також їх представлення в електронно-обчислювальній машині у вигляді пікселей. Розглянуто структуру цифрових та фізичних пікселів, їх характеристики та можливі репрезентації в різних мовах програмування. Було широко розглянуті колірні моделі, види та їх відмінності. Після складання опису кожної з моделей, стало вирішене питання з вибору колірної моделі для використання її у веб-сервісі.

Не менш важливим був вибір інструментів, для написання веб-сервісу. Так як додаток має бути швидким та гнучким у розробці, використовувались нові та добре документовані бібліотеки на базі мови програмування Java.

У якості основного фреймворка для розробки, я використав Spring Framework через його можливості ін'єкції залежностей. В результаті вийшла лаконічна та продуктивна система. В ролі хмарного сховища зображень було вибрано Azure Storage, через його універсальність та можливості по різному налаштовувати авторизований доступ до файлів. Дослідив та використав архітектурну модель веб-сервісів MVC, тому використовувались різні відокремлені рівні додатку для кращої зв’язності компонентів.

Було вирішено ряд проблем з сумісністю різних зображень та їх сортування, порівняння один з одним. В результаті розробки алгоритму складання характеристики зображень, стало можливим такі операції як сортування за будь-яким кольором чи яскравістю.

Задля покращення швидкодії, були прийнятті концептуальні рішення, що дають максимальний приріст та продуктивність. В результаті роботи був розроблений робочий прототип веб-сервісу з прикладами функціональності.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. М. Борн, Е. Вольф. Основи оптики. — 2-е вид. — Харків — М.Наука, 1973. – 121с.
2. Britannica – Isaac Newton/Career [Електронний ресурс] – Режим доступу: https://www.britannica.com/biography/Isaac-Newton/Career (дата звернення: 03.06.2022). – Назва з екрану.
3. Рудольф Ф. Граф. Сучасний словник електроніки. - Оксфорд: Ньюнс, 1999 – с. 569.
4. Глушаков, С.В. Комп’ютерна графіка / С.В. Глушаков, Г.А. Кнабе. - М.: Харків: Фоліо, 2002. - 420 с
5. Марк Д. Фершильд. Моделі колірного сприйняття. Переклад А. Шадрина. – СПб, 2006, - 420с.
6. RGB color model [Електронний ресурс] - https://www.hisour.com/rgb-color-model-24867/. (дата звернення: 03.06.2022). – Назва з екрана.
7. RGB and CMYK - differance [Електронний ресурс] - https://br24.com/en/rgb-cmyk-differences/. (дата звернення: 04.06.2022). – Назва з екрана.
8. Raster vs Vector. [Електронний ресурс] - https://www.adobe.com/creativecloud/file-types/image/comparison/raster-vs-vector.html/. (дата звернення: 05.06.2022). – Назва з екрана.
9. Microservices with Spring. [Електронний ресурс] - https://spring.io/blog/2015/07/14/microservices-with-spring. (дата звернення: 05.06.2022). – Назва з екрана.
10. RGB to HSV converter. [Електронний ресурс] - https://www.rapidtables.com/convert/color/rgb-to-hsv.html. (дата звернення: 06.06.2022). – Назва з екрана.

11. Walls C. Spring in action/ Craig Walls. – 2-ге вид. – Greenwich, CT: Manning Publications Co., 2008. – 730с.

12. Філософія Java. Брюс Еккель. – 4-е вид. – Київ. Пітер Прес. – 2015 – 1168с.

13. Spring Framework Documentation [Електронний ресурс] – режим доступу:https://docs.spring.io/spring-framework/docs/current/reference/html/ (дата звернення: 03.06.2022). - Назва з екрану.

14. Liquibase Database Migrations. [Електронний ресурс] – Режим доступу: https://docs.liquibase.com/ (дата звернення: 08.05.2022). - Назва з екрану.

15. Spring Framework - Overview [Електронний ресурс] – Режим доступу:https://www.tutorialspoint.com/spring/spring_overview.htm(дата звернення: 08.05.2022). – Назва з екрану.

16. Vue.js Documentation [Електронний ресурс]– Режим доступу: https://vuejs.org/guide (дата звернення: 08.06.2022). – Назва з екрана

17. Документація по службі сховища Azure. [Електронний ресурс]– Режим доступу: https://docs.microsoft.com/ua-ua/azure/storage/ (дата звернення: 07.06.2022). – Назва з екрану.