Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБКА ВЕБЗАСТОСУНКУ ДЛЯ НАДАННЯ ПЕРСОНАЛЬНИХ РЕКОМЕНДАЦІЙ ЩОДО ОДЯГУ І МУЗИКИ У ЗАЛЕЖНОСТІ ВІД ПОГОДИ
(тема)

Виконав:
студент 4 курсу, групи ІТІНФ-20-1
Писарєв В.С.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Вечірська І.Д.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові _____Писарєву Владиславу Сергійовичу_____
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка вебзастосунку для надання персональних рекомендацій щодо одягу і музики у залежності від погоди

затверджена наказом університету від 20 травня 2024 року № 464 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 03 червня 2024 р.

3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, мова програмування JavaScript, фреймворк Spring Boot, Deezer API, інтегроване середовищє розробки IntelliJ IDEA, вільна система керування реляційними базами даних MySQL.

4. Перелік питань, що потрібно опрацювати в роботі
1. Аналіз предметної області та постановка задачі.
2. Вибір та обґрунтування методів та технологій.
3. Реалізація поставленої задачі та тестування.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність проблеми розробки вебзастосунку для надання персональних рекомендацій щодо одягу і музики у залежності від погоди, постановка задачі, тестові зображення.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	08.04.2024	
2	Аналіз завдання, підбір літератури	08.04.24-16.04.24	
3	Аналіз літератури з досліджуваної проблеми	17.04.24-20.04.24	
4	Аналіз технічних засобів	21.04.24-29.04.24	
5	Розробка методів та алгоритмів	26.04.24-01.05.24	
6	Програмна реалізація	02.05.24-28.05.24	
7	Оформлення пояснювальної записки	29.05.24-03.06.24	
8	Перевірка на плагіат	04.06.24	
9	Рецензування	05.06.24	
10	Підготовка презентації та доповіді	06.06.24-11.06.24	
11	Занесення роботи в електронний архів	12.06.24	
12	Попередній захист кваліфікаційної роботи	12.06.24	

Дата видачі завдання 8 квітня 2024 р.

Студент _____
(підпис)

Керівник роботи _____ _____ доц. Вечірська І.Д._____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 70 с., 12 рис., 30 джерел.

ПІДБІР МУЗИКИ, ПІДБІР ОДЯГУ, АНАЛІЗ ДАНИХ, АНАЛІЗ ГЕОЛОКАЦІЇ, ГЕНЕРАЦІЯ РЕКОМЕНДАЦІЙ.

Об’єктом роботи є алгоритми для надання персональних рекомендацій щодо одягу та музики відносно погодних умов.

Метою роботи є створення системи, яка надаватиме користувачу рекомендації щодо одягу та музики відповідно до отриманої погоди в його геолокації.

Під час роботи використано методи для аналізу погодних умов, методи для аналізу музики та одягу на основі, яких було розроблено алгоритми для роботи сервера. Проведено дослідження, під час якого проаналізовано точність роботи алгоритмів. Розроблено API методи, завдяки яким можна звертатись до вебзастосунку незалежно від системи, для отримання персональних рекомендацій.

У результаті роботи розроблено повноцінний вебзастосунок, який надає рекомендації користувачу відповідно до його координат.

SELECTION OF MUSIC, SELECTION OF CLOTHES, DATA ANALYSIS, GEOLOCATION ANALYSIS, GENERATION OF RECOMMENDATIONS.

The object of the work is a web application for providing personal recommendations for clothing and music based on weather conditions.

The aim of the work is to create a system that will receive weather data based on the user’s geolocation and provide recommendations for clothing and music accordingly.

During the work, methods for analyzing weather conditions, methods for analyzing music and clothing were used, based on which algorithms for server operation were developed. Research was conducted, during which the accuracy of the algorithms’ operation was analyzed. API methods were developed, which allow accessing the web application independently of the system to obtain personal recommendations.

As a result of the work, a fully functional web application was developed, which provides recommendations to the user based on their coordinates

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API – Application Programming Interface (інтерфейс програмного застосунку)

AI – Artificial Intelligence (штучний інтелект)

SQL – Structured Query Language (структурована мова запитів)

NoSQL – Not only SQL (не тільки структурована мова запитів)

MVC – Model-View-Controller (модель вид контролер)

HTTP – Hypertext Transfer Protocol (протокол передачі гіпертексту)

JMX – Java Management Extensions (розширення керування Java)

MVP – Minimum Viable Product (мінімально життєздатний продукт)

ООП – Об’єктно-орієнтоване програмування

HTML – HyperText Markup Language (мова розмітки гіпертексту)

CSS – Cascading Style Sheets (каскадні таблиці стилів)

SEO – Search Engine Optimization (пошукова оптимізація)

IDE – Integrated Development Environment (інтегроване середовище розробки)

ВСТУП

Зв’язок між погодними умовами і нашим настроєм, емоціями і навіть фізичним здоров’ям завжди був очевидний. Сонячні дні можуть підняти настрій, але похмура погода може викликати сонливість і млявість. І, як не дивно, наші вподобання в одязі та музиці часто відображають наше ставлення до погоди. Тому ідея створення вебзастосунку, який автоматично аналізує погодні умови та надає персоналізовані рекомендації щодо вибору одягу та музики, видається найбільш прийнятною для реалізації.

Сучасний темп життя вимагає від нас гнучкості та адаптації до різних ситуацій, у тому числі до мінливих погодних умов. Немає кращого часу, ніж сьогодні, щоб створити інноваційні інструменти, щоб це сталося.

Метою цієї роботи є не просто розробка вебзастосунку, а створення системи, яка отримуватиме погоду відповідно до геолокації користувача та надаватиме йому рекомендації щодо одягу та музики. В майбутньому на основі цієї роботи планується розробити більш детальний вебзастосунок, який зберігав би інформацію від користувачів таку як: погодні умови, оцінка рекомендацій користувачем і так далі. Ці дані допоможуть розвивати застосунок ще більше і він зможе надавати більш точну інформацію далі.

В цій роботі досліджується:

– погодні умови та як вони впливають на наше здоров’я та настрій;

– зв’язок між погодою, нашими смаками, нашим вибором одягу та музикою;

– розробка алгоритмів, які враховують погодні умови та особисті переваги для надання персоналізованих рекомендацій;

– впровадження вебзастосунку з інтуїтивно зрозумілим інтерфейсом, який дозволяє користувачам легко отримувати рекомендації.

Першим етапом реалізації є збір та аналіз поточних даних про погоду. Це передбачає виклик загальнодоступних API, таких як OpenWeatherMap і AccuWeather, для отримання даних про температуру, вологість, тиск та інші

показники для певного географічного розташування. Використання історичних даних про погоду, щоб отримати більш точні та надійні рекомендації. Це дає змогу враховувати сезонні зміни, тенденції та інші фактори, які впливають на вибір одягу та музики. Дані можуть надходити з різних джерел, включаючи NOAA та інші погодні агентства. Розроблений алгоритм буде реалізований у вебзастосунку з інтуїтивно зрозумілим інтерфейсом. Це включає розробку інтерфейсу для відображення рекомендацій і бекенда для обробки та аналізу даних. Також, необхідно забезпечити безпеку даних і зручний доступ для користувачів.

Використовуючи статистичні методи та машинне навчання, можна знайти кореляції між погодними умовами та вибором одягу та музики. Наприклад, аналіз даних і алгоритми класифікації можуть бути використані для визначення того, які типи одягу найкраще підходять для певних діапазонів температур або погодних умов. Збираючи дані про особисті вподобання користувачів, як-от стиль одягу чи музичні уподобання, можна розробити алгоритми, які надають персоналізовані рекомендації. Цього можна досягти за допомогою спільних методів фільтрації та споживчої аналітики.

Отже, ціль – створювати інструменти, які не тільки спрощують повсякденне життя, але й покращують самопочуття та якість життя, адаптуються до різних погодних умов і зберігають індивідуальність та комфорт.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Об’єкт та предмет дослідження

Загальний опис галузі «Розробка вебзастосунку, який надає персоналізовані рекомендації щодо одягу та музики на основі погоди» відноситься до досліджень і розробки спеціалізованих вебсервісів, які використовують API сторонніх сервісів для надання точних даних погодних умов, бібліотеки музики, одягу тощо.

Метою дослідження є розробка вебзастосунку, який має надавати персоналізовані рекомендації щодо одягу та музики в залежності від погодних умов. Ця вебпрограма діє як інтерактивний інструмент, який надає користувачам актуальні та персоналізовані поради щодо вибору одягу та музики з урахуванням місцевих погодних умов.

Предметом дослідження є аналіз і розробка алгоритмів, які дозволяють вебзастосункам ефективно виконувати бажані функції. Це включає в себе аналіз даних про погоду [1, 2], розуміння її впливу на вибір одягу та музики, розробку алгоритмів для персоналізованих рекомендацій [3, 4] і створення інтерфейсів для зручного доступу до цих рекомендацій. Це дослідження також охоплює вивчення тенденцій у сфері рекомендаційних систем і сучасного стану аналізу погоди з метою впровадження їх у практичні застосунки [5].

Ця тема актуальна з кількох причин. По-перше, сучасні технології дозволяють збирати та аналізувати великі обсяги даних, наприклад дані про погоду та інформацію про вподобання користувачів. По-друге, все більша залежність людей від мобільних технологій значно збільшила попит на цей тип персоналізованих послуг. По-третє, через зміну клімату та непередбачуваність погодних умов можливість отримувати актуальні поради щодо вибору одягу та музики підвищує комфорт та зручність користувача. По-четверте, із зростанням свідомості про екологічні проблеми та збільшенням інтересу до сталого способу життя, персоналізовані рекомендації щодо одягу

та музики, враховуючи погодні умови, можуть сприяти більш ефективному використанню ресурсів та зменшенню екологічного впливу.

Таким чином, мета нашого дослідження полягає в тому, щоб розробити інноваційні інструменти, які сприяють підвищенню комфорту та задоволення користувачів, надаючи персоналізовані рекомендації щодо одягу та музики на основі поточних погодних умов.

1.2 Сучасні тенденції в галузі рекомендаційних систем та їх вплив на розробку персоналізованих інструментів

Рекомендаційні системи знаходяться в авангарді технологічних досягнень, які є результатом поєднання штучного інтелекту, аналізу даних і розмовних інтерфейсів. Останні тенденції в цій галузі визначають нові підходи до персоналізації та вдосконалення систем рекомендацій [6-8].

Однією з важливих тенденцій є зростання ролі AI у системах рекомендацій. AI дозволяє системам аналізувати великі обсяги даних і робити точні прогнози на основі цього аналізу. Використовуючи методи машинного та глибокого навчання, система може вчитися на попередніх виборах користувача та надавати персоналізовані рекомендації.

Ще одна тенденція – зростаюче значення контекстного контенту. Сучасні системи рекомендацій все більше звертають увагу на контекст взаємодії користувача та системи. Наприклад, система може рекомендувати відповідний одяг і музику з урахуванням погодних умов і місця розташування користувача [9-12].

Також варто відзначити зростаючу важливість взаємодії з користувачем. Системи рекомендацій стають все більш інтерактивними, враховуючи ваші відгуки, історію вебперегляду та вибір для надання більш точних і корисних рекомендацій.

Ці тенденції в контексті розробки персоналізованих інструментів для надання рекомендацій щодо одягу та музики, пов’язаних із погодою, використовують передові методи аналізу даних і створюють інтерактивні функції, які забезпечують комфорт і ефективність для користувачів. Це показує важливість розвитку [13, 14].

Сучасні тенденції в галузі рекомендаційних систем визначають нові підходи до персоналізації та вдосконалення інтерактивності. Зростаюча роль штучного інтелекту, урахування контексту та взаємодія з користувачем є ключовими аспектами розвитку цих систем. У контексті розробки персоналізованих інструментів для надання рекомендацій щодо одягу та музики відносно погоди, використання передових методів аналізу даних та інтерактивних функцій стає критичним для забезпечення зручності та ефективності для користувачів [15-18].

1.3 Технологічні інновації в аналізі погодних умов та їх використання для персоналізованих рекомендацій

Завдяки технологічним інноваціям, таким як використання сучасних метеорологічних супутників, датчиків, дронів і штучного інтелекту, аналіз погоди стає точнішим і доступнішим. Ці інновації відкривають нові можливості для розробки персоналізованих рекомендацій для користувачів у багатьох сферах, особливо в секторах одягу та музики.

Одним із ключових нововведень є використання сучасних метеорологічних супутників, які надають велику кількість даних про погоду в реальному часі. Ці дані дозволяють нам точно визначати температуру, вологість, швидкість вітру та інші параметри, які важливо враховувати при складанні рекомендацій щодо одягу та музики.

Крім того, датчики та дрони можна використовувати для збору додаткових даних про погоду на регіональному рівні. Наприклад, датчики

можуть вимірювати опади, а дрони можуть стежити за хмарами на низькій висоті та погодними умовами. Ці дані доповнюють дані супутників і дають більш детальну та повну картину погоди в певному регіоні [19, 20].

Тайрос-7 (рис. 1.1) — метеорологічний супутник сімейства супутників Тайрос (TheiaROS). Він призначений для збору даних про клімат і погоду Землі. Основна мета його місії — моніторинг стану атмосфери та надання точних прогнозів погоди для різних регіонів землі.

Тайрос-7 оснащений різноманітними приладами, включаючи спектральний радіометр, радар зондування атмосфери та прилади для вимірювання температури поверхні та теплового випромінювання. Всі ці пристрої дозволяють збирати різні дані про атмосферні умови, хмарність, температуру та інші показники, які впливають на погоду.

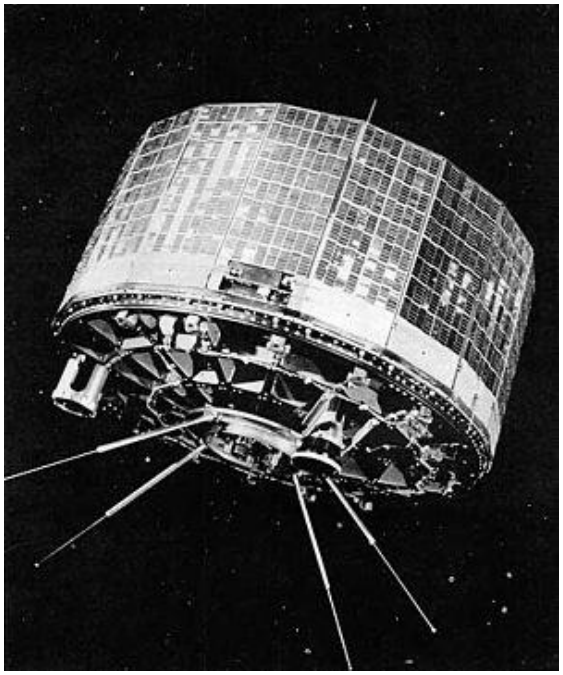

Рисунок 1.1 – Метеорологічний супутник «Тайрос-7»

Delta OHM HD2015 (рис. 1.2) – це датчик опадів, призначений для вимірювання опадів у м’якому кліматі. Цей датчик широко використовується на метеостанціях, сільськогосподарських дослідженнях, гідрологічних дослідженнях та інших галузях, де важливе точне вимірювання опадів.

HD2015 працює за принципом вимірювання опадів. Він використовує відстійник, у який потрапляють опади, і базується на вимірюванні змін рівня рідини в цьому відстійнику. За допомогою спеціального механізму ця зміна рівня перетворюється на вимірювання кількості опадів у міліметрах або дюймах.

Рисунок 1.2 – Delta OHM HD2015 датчик опадів для м’якого клімату

Моніторинговий дрон Reactive Drone RDM2 (рис. 1.3) – це безпілотний літальний апарат, розроблений спеціально для вимірювання хмарності та спостереження за погодою. Ці дрони відіграють важливу роль у зборі даних про атмосферні умови та в кліматичних і метеорологічних дослідженнях.

Рисунок 1.3 – Моніторинговий дрон Reactive Drone RDM2

Використання штучного інтелекту також відіграватиме ключову роль у покращенні аналізу погоди. Алгоритми машинного навчання дозволяють системі аналізувати великі обсяги даних і робити прогнози з високою точністю. Це відкриває можливість індивідуальних рекомендацій для конкретних погодних умов та індивідуальних уподобань користувача.

Технологічні інновації в аналізі погоди дозволяють розробляти точніші та персоналізовані системи рекомендацій. Це відкриває нові перспективи для створення інноваційних вебзастосунків, які надають користувачам зручні та ефективні рекомендації щодо одягу та музики на основі поточних погодних умов [21-23].

Ще одним ключовим нововведенням є використання сучасних алгоритмів машинного навчання, які дозволяють системі аналізувати та прогнозувати погодні умови на основі зібраних даних. Це відкриває широкі можливості для створення інтелектуальних систем, які автоматично адаптуються до змін погоди та уподобань користувачів.

Інновації в аналізі погодних умов мають великий потенціал у багатьох сферах життя, особливо в розробці персоналізованих систем рекомендацій. Це дозволяє нам краще зрозуміти наше середовище та надає інформацію, необхідну для прийняття правильних рішень.

1.4 Підходи до аналізу погодних даних та їх вплив на емоційний стан та вибір користувачів

Аналіз даних про погоду та їх вплив на емоційний стан і вибір користувачів є ключовим елементом у розробці персоналізованих систем рекомендацій. Різні підходи до аналізу погодних даних дозволяють не тільки прогнозувати погодні умови, але й зрозуміти, як ці умови впливають на психологічний стан користувачів і їхній вибір [24, 25].

Одним із підходів до аналізу є співвіднесення погодних умов з емоційним станом людей. Дослідження показують, що певні погодні умови, наприклад сонячна або дощова погода, можуть впливати на настрій і емоційний стан людини. Наприклад, сонячна погода часто асоціюється з піднесеним настроєм і енергією, а дощ може викликати почуття смутку і поганий настрій.

Інший підхід полягає у врахуванні погодних умов під час надання рекомендацій користувачам. Наприклад, у холодну та похмуру погоду рекомендований затишний та теплий одяг, а в сонячну та теплу погоду – більш легкий та легкий одяг. Також можна налаштувати музичні рекомендації залежно від погодних умов, забезпечуючи веселі й енергійні пісні для сонячних днів і заспокійливі пісні для дощових днів.

Усі ці підходи базуються на аналізі погодних даних та їх впливу на психологічний стан і поведінку користувачів. Застосування цих підходів дозволяє створювати більш персоналізовані та ефективні системи рекомендацій, які враховують не лише суб'єктивні потреби користувачів, а й їхній емоційний стан та настрій [26].

1.5 Аналіз індивідуальних вподобань користувачів та їх відображення у рекомендаційних системах

Аналіз індивідуальних уподобань користувачів є ключовим аспектом створення ефективних і персоналізованих систем рекомендацій. Для досягнення цієї мети можна використовувати різноманітні методи та підходи для збору, аналізу та використання даних про вподобання окремих користувачів.

Одним із підходів є використання історичних даних взаємодії користувача з системою. Ці дані включають інформацію про раніше переглянуті продукти, покупки, оцінки відгуків та інші взаємодії. Аналізуючи ці дані, можна визначити основні вподобання та інтереси кожного користувача

та надати рекомендації відповідно до їхніх потреб. Тому вебзастосунок, розроблений в ході цієї роботи, можна з легкістю імпортувати в будь-який погодний сервіс, сервіс новин, тощо [27].

Інший підхід полягає в розгляді контекстної інформації. На основі цього вебзастосунку можна реалізувати рекомендації для користувача на різноманітних маркетплейсах, інтернет-магазинах одягу і так далі. Це означає врахування додаткових факторів, таких як час доби, місце розташування, погодні умови та інші параметри, які можуть вплинути на вибір користувача. Наприклад, взимку можна рекомендувати товари для зимового відпочинку та спортивний інвентар, а влітку – товари для активного відпочинку.

Методи аналізу текстової інформації, такі як обробка природної мови, також використовуються для розуміння особистих уподобань користувачів на основі коментарів, відгуків і соціальних мереж. Це дозволяє системі краще розуміти індивідуальні потреби та вподобання кожного користувача та надавати кращі рекомендації.

Поєднуючи всі ці підходи, можна ефективно аналізувати індивідуальні переваги користувачів і відображати їх у системах рекомендацій. Це дозволяє створювати персоналізовані та корисні рекомендації, які відповідають унікальним потребам та інтересам кожного користувача [28, 29].

1.6 Визначення основних завдань та цілей розробки вебзастосунку для надання персональних рекомендацій щодо одягу та музики відносно погоди

Основне завдання та мета розробки вебзастосунку, який надає персоналізовані рекомендації щодо одягу та музики, пов’язані з погодою, полягає в тому, щоб надати відповідні та корисні рекомендації з урахуванням погодних умов та їх впливу на особисті уподобання [30].

Основними завданнями розробки вебзастосунків є:

– збір і аналіз даних про погоду;

– аналіз особистих уподобань;

– розробка алгоритмів рекомендацій;

– розробка зручного інтерфейсу;

– забезпечення ефективного способу отримання рекомендацій.

1.7 Постановка задачі

Таким чином, розробка вебзастосунку для надання персональних рекомендацій щодо одягу і музики у залежності від погоди є актуальним завданням. Тому ставиться завдання розробки програми, яка б надавала таку можливість.

Об’єктом роботи є алгоритми для надання персональних рекомендацій щодо одягу та музики відносно погодних умов.

Метою роботи є створення системи, яка надаватиме користувачу рекомендації щодо одягу та музики відповідно до отриманої погоди в його геолокації.

Для досягнення мети необхідно вирішити такі завдання:

– провести аналіз існуючих рішень щодо збору інформації про погоду;

– розробити метод отримання музикальних композицій за допомогою стороннього API;

– розробити алгоритм надання персоналізованих рекомендацій щодо одягу на основі зібраних даних про погоду;

– розробити алгоритм надання персоналізованих рекомендацій щодо музики на основі зібраних даних про погоду та індивідуальних уподобань користувачів;

– розробити зручний та привабливий користувацький інтерфейс для вебзастосунку;

– протестувати розроблений вебзастосунок, щоб виявити помилки та недоліки.

2 ОБҐРУНТУВАННЯ ВИБРАНИХ МЕТОДІВ ТА ТЕХНОЛОГІЙ

2.1 Моделювання залежності між погодними умовами та вибором одягу

Вибір одягу – складний і багатогранний процес, часто залежний від погодних умов. Люди звикли підбирати одяг залежно від конкретних погодних умов, таких як температура, вологість, опади та інші фактори.

По-перше, щоб побудувати модель зв’язку між погодними умовами та вибором одягу, нам потрібно зібрати достатню кількість даних про погоду та інформацію про типи одягу, який люди обирають у конкретних погодних умовах. Для цього можна використовувати історичні дані про погоду, а також опитування та опитування користувачів.

Після того можна приступити до розробки математичної моделі. Одним із підходів є використання моделей лінійної або нелінійної регресії для прогнозування типу одягу на основі погодних параметрів. Наприклад, можна використовувати такі фактори, як температура, вологість і швидкість вітру, як незалежні змінні, а тип одягу – як залежну змінну.

Крім того, можна розглянути імовірнісну модель, яка враховує статистичний розподіл вибору одягу залежно від погодних умов. Наприклад можна використовувати статистичні методи Байєса для моделювання ймовірності вибору певного типу одягу за певних погодних умов.

Такі моделі можна покращувати та оптимізувати за допомогою методів машинного навчання, які можуть враховувати більш складні зв’язки між погодними параметрами та вибором одягу. Наприклад, нейронні мережі можна використовувати для автоматичного виявлення нетривіальних закономірностей у даних і побудови точних моделей.

В рамках цієї роботи було обрано метод, який полягає у зберіганні різних типів одягу в базі даних, після чого, відповідно до погоди, з бази отримується повен комплект одягу. В нашому випадку використовується база даних –

MySQL. Це потужна система керування базами даних (СУБД), яка використовується для зберігання та керування великими обсягами даних. В рамках цього проекту MySQL можна використовувати з кількох причин, які слід враховувати:

- відкритість та безкоштовність;
- надійність і стабільність;
- простота використання;
- підтримка стандартів;
- масштабованість.

MySQL є відкритою системою. Це означає, що його можна використовувати безкоштовно або з відкритим вихідним кодом. Це робить її доступним для широкого кола користувачів і проектів на будь-якому рівні бюджету. Сервіс роками використовується великими організаціями та вебпроєктами завдяки своїй надійності та стабільності. Підходить для завдань, які вимагають великих обсягів даних або високих вимог до продуктивності. Має дуже простий синтаксис, з яким легко працювати. Це робить його ідеальним вибором для новачків у роботі з базами даних, а також для швидкої розробки проектів. Також він підтримує багато стандартів SQL і сумісний з багатьма іншими інструментами та бібліотеками. Це дозволяє легко інтегрувати MySQL в існуючі проекти або змінювати існуючий код для використання MySQL. Можливість масштабування як горизонтально (додавання нових серверів), так і вертикально (збільшення ресурсів на існуючих серверах) – ще одна перевага MySQL. Це робить його популярним вибором для вебпроектів, які очікують збільшення даних і трафіку.

В рамках проєкту з розробки вебзастосунку, який надає персоналізовані рекомендації щодо одягу та музики на основі погоди, MySQL був ідеальним вибором для зберігання та керування даними про користувачів, уподобання, погодні умови та рекомендації. Це забезпечує надійність системи, ефективність і масштабованість, полегшує керування великими обсягами даних і забезпечує швидкий доступ до необхідної інформації.

Для адміністрування бази даних було обрано PhpMyAdmin. Це потужний інструмент для адміністрування та управління базами даних MySQL через вебінтерфейс. Використання PhpMyAdmin має декілька переваг, які роблять його гарним інструментом для розробників і адміністраторів баз даних:

- зручний та інтуїтивно зрозумілий інтерфейс;
- можливості для адміністрування бази даних;
- підтримка безпеки;
- багатомовність;
- відкритий код та безкоштовність.

PhpMyAdmin має простий та зрозумілий інтерфейс, що дозволяє легко та швидко виконувати різноманітні операції з базою даних, такі як створення таблиць, зміна структури, додавання записів, виконання запитів тощо. Він надає широкий спектр інструментів для адміністрування бази даних, таких як керування користувачами та їх привілеями, створення резервних копій, відновлення даних, перегляд та аналіз вмісту таблиць, оптимізація таблиць та індексів, моніторинг запитів та використання ресурсів. Також PhpMyAdmin забезпечує різні заходи безпеки, такі як автентифікація користувачів, шифрування даних та захист від SQL-ін’єкцій та інших атак.

PhpMyAdmin є відкритим програмним забезпеченням, що дозволяє безкоштовно використовувати його без обмежень. Крім того, відкритий код дає можливість розробникам вносити зміни та вдосконалювати програму відповідно до своїх потреб

Отже, використання PhpMyAdmin – це гарний вибір для адміністрування та управління базами даних MySQL, який забезпечує зручний та ефективний спосіб керування даними та забезпечує безпеку та надійність операцій з базою даних.

В базі даних була розроблена таблиця (рис. 2.1) спеціально для одягу, яка включає в себе такі стовпці, як: ідентифікаційний номер, тип одягу, зображення елемента одягу, назва. Також, для кожного елемента таблиці

враховуються такі дані про погоду: опади, температура, рівень ультрафіолетового випромінювання, погодні умови, швидкість вітру.

	id 1	clothing_type	image	name	precipitation_type	temperature_range	uv_index	weather_condition	wind_intensity
Delete	1	0	https://gepard-hunter.com/content/images/27/480x27...	Кепка	5	4	2	0	0
Delete	2	0	https://gepard-hunter.com/content/images/27/480x27...	Кепка	5	3	3	0	0
Delete	3	0	https://upload.wikimedia.org/wikipedia/commons/6/6...	Шапка	1	1	1	4	2
Delete	4	1	https://opticaluxor.ua/app/uploads/public/629/496/...	Сонцезахисні окуляри	5	3	3	0	0
Delete	5	1	https://img.joomcdn.net/709a790f83139390121a1442ab...	Вітрозахисні окуляри	1	1	1	4	2
Delete	6	2	https://borec.in.ua/5463-medium_default/kurtka-man...	Куртка	0	1	1	2	1
Delete	7	3	https://purplelama.com.ua/wp-content/uploads/2022/...	Светр	5	2	1	1	1
Delete	8	4	https://images.prom.ua/5453783032_w640_h640_futbol...	Футболка	5	3	1	0	0
Delete	9	5	https://borec.in.ua/5196-large_default/shtani-mant...	Штани	5	2	1	1	1
Delete	10	5	https://dodosocks.com/cdn-cgi/image/width=1400,hei...	Шорти	5	4	1	0	0
Delete	11	6	https://static.insalescdn.com/images/products/1/16...	Кросівки	0	2	1	1	1
Delete	12	6	https://sezon.ua/image/catalog/image/easyfoto/1619...	Шльопанці	5	3	1	0	0

Рисунок 2.1 – Вигляд таблиці Clothes в phpMyAdmin

Після того, як отримуються дані про погодні умови, створюється об’єкт, який містить всі елементи, які є в одязі. Далі було розроблено алгоритм, який порівнює кожен пункт і робить запит до бази даних для отримання необхідного елементу одягу. Порядок роботи алгоритму:

Крок 1. Спочатку створюється об’єкт, який буде містити весь одяг, який підбирається для користувача відповідно до погодних умов.

Крок 2. Для кожної категорії одягу (наприклад, головний убір, верхній одяг, светри тощо) викликається сервіс, щоб отримати список доступного одягу для кожної категорії.

Крок 3. Списки одягу сортуються відповідно до певних критеріїв, наприклад, за типом погоди, щоб найбільш підходящий одяг був на першому місці у списку.

Крок 4. Кожного списку одягу вибирається найпідходящіший елемент відповідно до поточних погодних умов і додавання його до контейнера одягу.

Крок 5. У залежності від погоди або інших умов може додаватися додатковий одяг, наприклад, сонцезахисні окуляри або зовнішній одяг для прохолодної погоди

Крок 6. На останок повертається контейнер, який містить всі вибрані елементи одягу для користувача залежно від погодних умов.

2.2 Аналіз музичних вподобань в залежності від погоди

Музика відіграє важливу роль у нашому житті та дозволяє нам виражати свої емоції, настрій та суб’єктивне сприйняття навколишнього світу. Останніми роками все більше досліджень показало, що музичні уподобання можуть змінюватися залежно від погодних умов. У цьому розділі буде розглянуто, як аналізувати музичні вподобання залежно від погоди та розробити підхід до визначення оптимального вибору музики для користувачів на основі погодних умов.

Першим кроком до аналізу музичних уподобань залежно від погоди є збір достатньої кількості даних. Дані про прослуховування музики можна збирати з потокових служб, таких як Spotify, Apple Music і YouTube, а дані про погоду можна отримати з відповідних джерел погоди;

Після збору даних можна проаналізувати зв’язок між музичними уподобаннями користувачів і погодними умовами. Статистичні методи, такі як кореляційний аналіз, можна використовувати, щоб визначити, які жанри музики є більш популярними за певних погодних умов;

За результатами аналізу можна розробити рекомендаційний алгоритм, який враховує погодні умови при виборі музики. Наприклад, у холодну та дощову погоду рекомендуються спокійні та затишні пісні, а в сонячну – бадьорі та енергійні треки;

Коли алгоритм розроблено, його необхідно перевірити на великих обсягах даних і користувачів. Також важливо постійно контролювати та оптимізувати алгоритми, щоб забезпечити їх ефективність та адаптуватися до мінливих погодних умов.

Аналіз музичних уподобань залежно від погоди може допомогти створити персоналізовані музичні рекомендації, які краще відповідають настрою та потребам користувача в конкретних погодних умовах.

В цій роботі використовується база даних, така ж, як і в пункті вище, для створення таблиці з музичними композиціям відповідно до погоди. Для роботи з музикою використовується відкрите API deezer.com.

Deezer (рис. 2.2) – це музичний стрімінговий сервіс, який надає доступ до мільйонів треків, альбомів і плейлистів. Це дозволяє користувачам слухати музику онлайн, створювати власні плейлисти, ділитися музикою з друзями та багато іншого.

Рисунок 2.2 – Логотип Deezer

У Deezer є API, який дозволяє розробникам створювати застосунки, що використовують дані та функції Deezer. Це дає можливість інтегрувати музичний контент Deezer у власні застосунки, вебсайти або сервіси.

Деякі основні функції, які можна реалізувати за допомогою API Deezer, включають отримання інформації про виконавців, альбоми, треки, плейлисти, а також можливість відтворення аудіо. Це дозволяє розробникам створювати різноманітні музичні застосунки, такі як музичні плеєри, музичні блоги, аналітичні інструменти та інше.

Щоб отримати доступ до API Deezer, розробники повинні зареєструватися на Deezer Developers і отримати ключ доступу API. Після цього вони можуть отримувати доступ до документації API та інструментів для розробки своїх застосунків.

Загалом, API Deezer відкриває широкі можливості для розробників у сфері музичних застосунків та сервісів, дозволяючи їм створювати інноваційні продукти на основі музичного контенту Deezer.

Deezer буде використано лише для отримання інформацію про пісні та можливість відтворення їх в нашому вебзастосунку. Зберігається ця інформація в спеціальній таблиці «Music» (рис. 2.3), яка містить такі стовпці: ідентифікаційний номер, назва, зображення, посилання на ресурс для відтворення пісні. Також, для отримання пісні спеціально для певних погодних умов, в таблиці є пункти, які відповідають за погоду такі, як: температура, кількість опадів, тип опадів, випромінювання, хмарність та швидкість вітру.

+ Options

	id	name	precipitation_type	preview	temperature_range	url	weather_condition	wind_intensity
Edit Copy Delete	1	Here Comes The Sun (Remastered 2009)	5	https://e-cdns-images.dzcdn.net/images/cover/aa94a...	3	https://cdns-preview-5.dzcdn.net/stream/c-5503df16...	0	0
Edit Copy Delete	2	Let It Snow! Let It Snow! Let It Snow! (with The B...	1	https://e-cdns-images.dzcdn.net/images/cover/d3c15...	1	https://cdns-preview-1.dzcdn.net/stream/c-16913b64...	4	0
Edit Copy Delete	3	Singing in the Rain	0	https://e-cdns-images.dzcdn.net/images/cover/93840...	2	https://cdns-preview-c.dzcdn.net/stream/c-c041ccad...	3	1
Edit Copy Delete	4	Stormy Weather	0	https://e-cdns-images.dzcdn.net/images/cover/793a7...	1	https://cdns-preview-1.dzcdn.net/stream/c-10c5117b...	5	2
Edit Copy Delete	5	Windy	0	https://e-cdns-images.dzcdn.net/images/cover/cc868...	3	https://cdns-preview-1.dzcdn.net/stream/c-15a6bbc9...	6	2
Edit Copy Delete	6	Winter Wonderland	1	https://e-cdns-images.dzcdn.net/images/cover/d2050...	1	https://cdns-preview-8.dzcdn.net/stream/c-8efa2734...	4	0
Edit Copy Delete	7	Walking On Sunshine	5	https://e-cdns-images.dzcdn.net/images/cover/1a81b...	2	https://cdns-preview-6.dzcdn.net/stream/c-6b8f3581...	1	1
Edit Copy Delete	8	Hot Fun in the Summertime	5	https://e-cdns-images.dzcdn.net/images/cover/68440...	4	https://cdns-preview-e.dzcdn.net/stream/c-ec36f827...	1	2

Рисунок 2.3 – Вигляд таблиці Songs в phpMyAdmin

В загальному, логіка алгоритму підбору пісні відповідно до погодних умов користувача схожа до алгоритму підбору одягу. В пісні зберігається погодні умови, під які вона була би вдалою, далі отримується погода від користувача, після чого відбувається сортування за схожими пунктами і користувачу повертається пісня з найбільшою кількістю співпадінь.

2.3 Технологія розробки серверної частини Spring Boot

Для розробки серверної частини було розглянуто багато різноманітних технологій, таких як Spring Boot, Django, Laravel. Саме для цього проєкту було обрано фреймворк Java, а саме – Spring Boot. Spring Boot – це популярний фреймворк для розробки серверних застосунків на мові програмування Java. Він вбудовує у себе багато стандартних бібліотек та інструментів, що спрощує процес розробки та підвищує продуктивність розробника.

Spring Framework складається з кількох модулів, які надають широкий спектр послуг:

- інверсія контейнерів керування;
- аспектно-орієнтоване програмування;
- доступ до даних;
- управління транзакціями;
- MVC;
- автентифікація та авторизація;
- віддалене керування;
- тестування.

Під час розробки вебпрограми, яка надає персоналізовані рекомендації щодо одягу та музики на основі погоди, використання Spring Boot дає кілька важливих переваг, так як він забезпечує зручний і ефективний спосіб створення вебзастосунків. Вбудовані стандарти та готові компоненти дозволяють розробникам швидко створити робочий прототип або MVP. Також Spring Boot поєднує різноманітні модулі та бібліотеки, які можна використовувати для різноманітних завдань, таких як робота з базами даних, розробка вебсервісів та інтеграція з іншими сервісами.

Spring Boot дозволяє легко масштабувати вебзастосунок у міру зростання даних і робочого навантаження. Spring Cloud дозволяє впроваджувати мікросервісні архітектури та розподілені проекти, а також

містить такі функції безпеки, як Spring Security, які захищають вебпрограми від потенційних загроз і атак.

Як можна побачити з рисунка 2.4, принцип роботи Spring Boot застосунку дуже простий, що значно полегшує розробку. Сам принцип полягає у використанні сервісів та моделей для бази даних. Тобто, для комфортної, налагодженої роботи з базою даних використовується Java клас-модель, який містить в собі поля, які стануть майбутніми стовпцями в таблиці базі даних. Далі у кожного такого класу є свій репозиторій і сервіс: репозиторій – напряму працює з базою даних, відправляє запити та отримує дані звідти, сервіс в свою чергу – спосіб відокремити весь функціонал, який може бути зайвим і призведе до випадкових помилок під час розробки.

Отже, коротко принцип роботи Spring Boot застосунку можна описати наступним чином: клієнт відправляє запит на контролер, з контролеру запит йде до сервісу, з сервісу до репозиторію, з репозиторію до бази даних і в зворотному порядку інформація повертається до користувача, обробляючись на цьому шляху. Використання такої системи називається – інкапсуляцією, що є одним з основних принципів ООП і вважається гарною практикою.

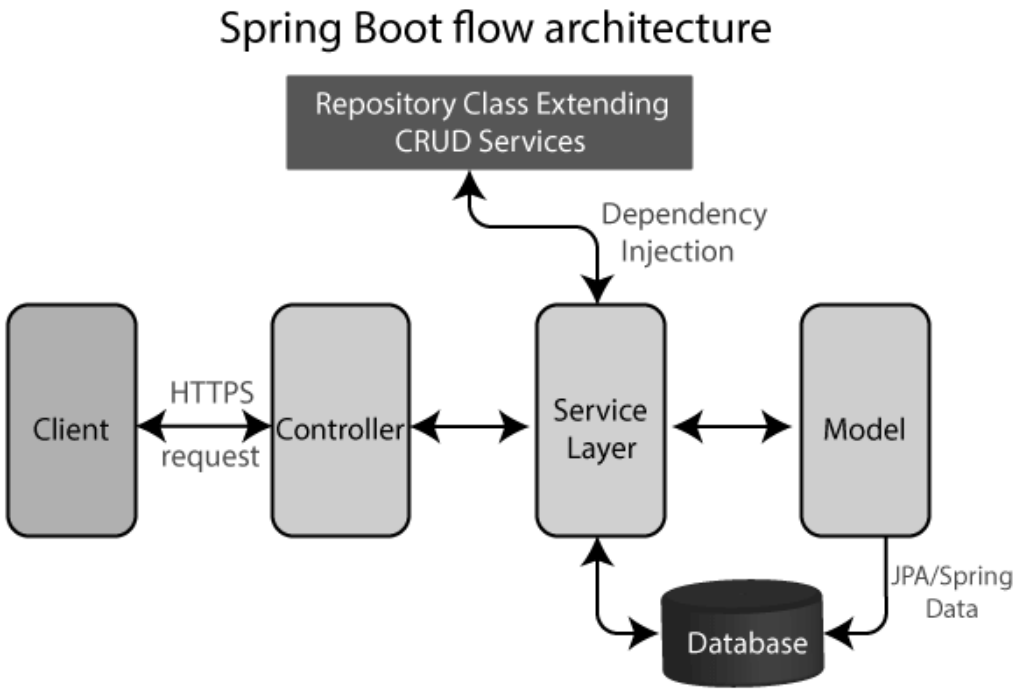

Рисунок 2.4 – Принцип роботи будь-якого Spring Boot застосунку

Тому використання Spring Boot у цьому випадку дозволяє швидко розробляти надійні, масштабовані та безпечні застосунки. Він надає потужний інструментарій для реалізації бізнес-логіки та інтеграції зовнішніх служб, спрощуючи процес створення та підтримки програм.

2.4 Обґрунтування вибору стеку для фронтенду

Вибираючи технологічний стек для інтерфейсної розробки, важливо ретельно розглянути всі можливі варіанти та вибрати той, який найкраще відповідає потребам проекту. Вибір правильного стека може значно вплинути на продуктивність, швидкість розробки та якість кінцевого продукту.

Крім того, правильний стек може сприяти підвищенню якості фінального продукту. Використання сучасних та ефективних технологій дозволяє створювати вебзастосунки, які працюють швидко, мають зручний інтерфейс та відповідають сучасним стандартам безпеки та стійкості.

Для проєкту з розробки вебсистеми для надання рекомендацій ввідносно погоди, було обрано найпростіший варіант – чистий HTML, CSS та JavaScript. Так як в нашому випадку немає необхідності у використанні складних функцій, нам треба лише отримувати координати користувача та виводити йому інформацію, можна обрати цей варіант, який буде швидшим за будь-які інші технології.

HTML – основа вебсторінок і визначає їх структуру та зміст. Це мова розмітки, яка використовується для створення текстового та мультимедійного вмісту на вебсторінках. HTML використовує теги для визначення різних елементів сторінки, таких як заголовки, параграфи, зображення, посилання тощо.

CSS – відповідає за зовнішній вигляд і стиль вебсторінок. Це мова стилів, яка дозволяє визначати кольори, шрифти, розміри, відступи, анімацію

та інші аспекти вигляду елементів HTML. CSS дозволяє зробити вебсторінки більш привабливими та професійними.

JavaScript – є мовою програмування, яка використовується для створення інтерактивних та динамічних елементів на вебсторінках. Вона використовується для додавання функціональності, такої як валідація форм, анімація, взаємодія з користувачем, динамічне оновлення даних без перезавантаження сторінки та інше.

Усі ці технології – HTML, CSS та JavaScript - використовуються спільно для створення вебсторінок та вебзастосунків. HTML визначає структуру сторінки, CSS відповідає за її вигляд, а JavaScript - за інтерактивність та динаміку. Вони є найбільш поширеними та основними технологіями веброзробки і є основою будь-якого вебпроекту.

Використання чистого HTML, CSS та JavaScript спрощує розробку фронтенду. Вони є базовими технологіями для розробки вебінтерфейсів і дозволяють швидко створювати та редагувати вебсторінки без необхідності вивчення складних фреймворків або бібліотек. Чистий HTML, CSS та JavaScript дозволяють створювати легкі та швидкозавантажувані сторінки, що особливо важливо для користувачів, які відкриватимуть вебзастосунок на мобільних пристроях або з повільним Інтернет-з’єднанням. Використання чистого HTML, CSS та JavaScript дає нам повний контроль над дизайном та поведінкою вебсторінок. Також використання чистого HTML, CSS та JavaScript дозволяє уникнути залежності від зовнішніх бібліотек та фреймворків. Це спрощує процес розробки, робить код більш простим та легким для розуміння та підтримки

Чистий HTML забезпечує кращу оптимізацію для пошукових систем, що дозволить вебзастосунку краще індексуватися та відображатися у пошукових результатах;

Отже, вибір чистого HTML + CSS + JS для фронтенду нашого вебзастосунку обґрунтований його простотою розробки, швидкістю

завантаження, гнучкістю та незалежністю від зовнішніх бібліотек, а також оптимізацією для SEO.

2.5 Вибір інтегрованого середовища розробки IntelliJ IDEA

Для роботи з Spring Boot було обрано інтегроване середовище розрбки IntelliJ IDEA. Це IDE, розроблене компанією JetBrains для програмування на мовах Java, Kotlin, Groovy, Scala та інших. IntelliJ IDEA відома своєю потужністю, гнучкістю та розширюваністю, які роблять його популярним інструментом серед розробників усього світу.

IntelliJ IDEA має кілька тарифів для користувачів: звичайний та Ultimate. Для розробки Spring Boot необхідно використовувати maven, він доступний в версії Ultimate, яку JetBrains надають безкоштовно для студентів.

IntelliJ IDEA надає розширену підтримку для різних мов програмування, таких як Java, Kotlin, Groovy, Scala, JavaScript, TypeScript, HTML, CSS, SQL та багатьох інших. Це дозволяє розробникам працювати з різноманітними технологіями в рамках одного інструменту.

Редактор коду IntelliJ IDEA має велику кількість функцій, які полегшують написання коду, такі як автодоповнення, автоматичне вирівнювання, підсвічування синтаксису, швидкі генерації коду та інші. Він також підтримує різні шаблони коду та стандарти оформлення, що допомагає зберігати стиль коду команди.

IntelliJ IDEA надає ряд інструментів для рефакторингу коду, які дозволяють змінювати його структуру та організацію з мінімальними зусиллями. Це включає в себе перейменування, витягування методів, розбиття та об’єднання методів, оптимізацію імпортів та інші операції. Інтеграція з іншими інструментами. IntelliJ IDEA легко інтегрується з іншими інструментами розробки, такими як системи контролю версій (Git, SVN),

засоби збирання проектів (Maven, Gradle), системи автоматичної збірки (Jenkins, TeamCity) та багато інших;

Також IntelliJ IDEA підтримує розширення через плагіни, які дозволяють розробникам налаштовувати та розширювати функціональність IDE під їхні потреби.

Інтегроване середовище розробки IntelliJ IDEA є потужним та універсальним інструментом, який допомагає розробникам створювати високоякісний код швидше та ефективніше.

Хоча IntelliJ IDEA є потужним та універсальним інструментом для розробки програмного забезпечення, він також має деякі недоліки, які варто врахувати. По-перше, він може вимагати значних ресурсів комп'ютера, особливо при роботі з великими проектами або використанні деяких розширень. Це може призводити до повільної роботи програми на менш потужних пристроях або працювати в значний обсяг оперативної пам'яті.

По-друге, Інтерфейс IntelliJ IDEA містить велику кількість функцій та опцій, що може перенавантажувати новачків або тих, хто рідко використовує певні функції. Навчання всіх можливостей та опцій може зайняти час.

По-третє, IntelliJ IDEA доступна у двох версіях: Community Edition (безкоштовна) та Ultimate Edition (платна). Ultimate Edition надає розширені можливості, такі як підтримка Java EE, Spring, інші мови програмування та інші інструменти. Вартість ліцензії може бути значним фактором для деяких користувачів.

Також через широкий функціонал та постійні оновлення, користувачам може знадобитися час для вивчення нових можливостей та оволодіння новими інструментами. Якщо ресурси комп'ютера обмежені, робота з IntelliJ IDEA може бути повільною або нестабільною, особливо при роботі з великими проектами.

Не зважаючи на ці недоліки, IntelliJ IDEA залишається одним з найпопулярніших інтегрованих середовищ розробки завдяки своїй потужності, гнучкості та широкому спектру функцій.

2.6 Використання технології MAMP

Як описано в розділах вище, для бази даних було обрано MySQL та PhpMyAdmin, але для того, щоб тестувати роботу вебзастосунка, необхідно мати ще один сервер, на якому буде розташована база даних. В рамках цього проєкту було обрано технологію MAMP, як сервер для баз даних.

MAMP – це популярний набір програмного забезпечення для розробки вебзастосунків на мові PHP та баз даних MySQL на операційних системах MacOS та Windows. Обираючи MAMP для розробки та роботи з базою даних, є декілька ключових факторів, які роблять його привабливим вибором:

- простота встановлення та конфігурації;
- швидкість та продуктивність.;
- підтримка багатьох мов програмування;
- зручне керування базою даних.

MAMP забезпечує швидке та просте встановлення і налаштування середовища розробки на локальному комп’ютері. Це дозволяє швидко розпочати роботу над проектом без необхідності витрачати час на складні налаштування.

MAMP надає швидке та ефективне середовище для розробки вебзастосунків. Він включає в себе оптимізовані версії серверного програмного забезпечення, такі як Apache, MySQL та PHP, що дозволяє працювати з великими обсягами даних без затримок.

Хоча MAMP спеціалізується на розробці на мові PHP та базі даних MySQL, він також підтримує інші мови програмування та технології, що робить його універсальним інструментом для різних типів вебпроектів.

Також він має вбудований інтерфейс для керування базою даних MySQL, який дозволяє легко створювати, змінювати та керувати базами даних та їх таблицями безпосередньо з вебінтерфейсу.

2.7 Моделювання вебсервісу

Для розробки вебсервісу було змодельована діаграму класів за допомогою вебзастсоунку draw.io (рис. 2.5).

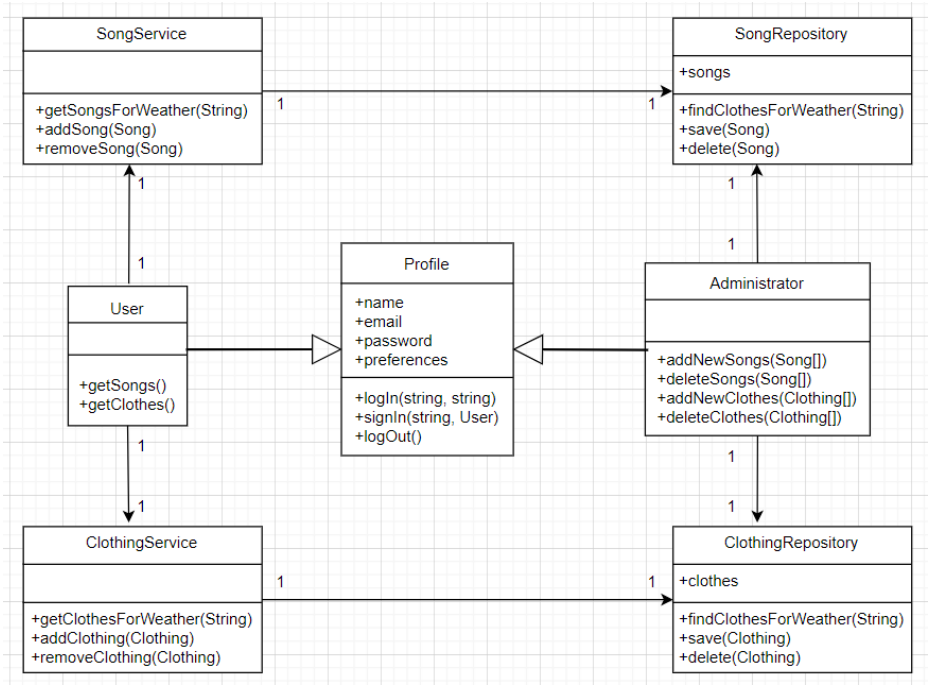

Рисунок 2.5 – Діаграма класів вебсервісу

На діаграмі способів використання (рис. 2.6) зображено акторів та усі їхні можливі дії у сервісі.

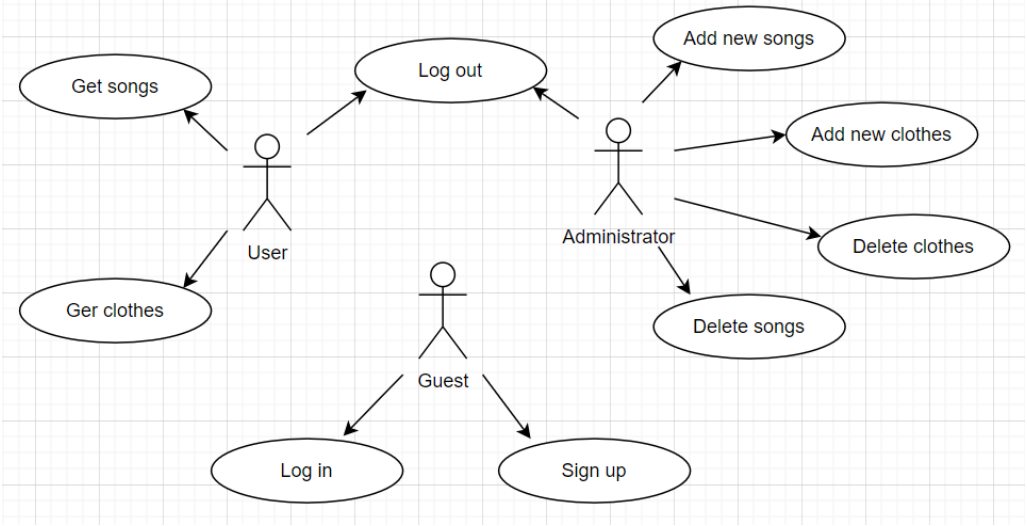

Рисунок 2.6 – Діаграма варіантів використання вебсервісу

Опис акторів системи:

− «Guest» – користувач, який ще не війшов до свого облікового запису. Йому доступні функції реєстрації та входу до системи;

− «User» – користувач, який виконує запит для пошуку пісень та одежі відповідно до погодних умов. Йому доступні функції отримання пісень та одягу, а також можливість виходу з свого профілю;

− «Administrator» – користувач, який може додавати нові пісні та одяг, або видаляти вже наявні пісня чи одяг з бази даних. Йому доступні функції додавання та видаляння пісень та одягу, а також можливість виходу з свого профілю.

Опис варіантів використання (прецедентів) системи:

− Log in – цей варіант використання ініціює «Guest». Він забезпечує можливість входу до свого акаунту;

− Sign up – цей варіант використання ініціює «Guest». Він забезпечує можливість реєстрації нового акаунту;

− Log out – цей варіант використання може ініціювати як «User» так і «Administrator». Він забезпечує можливість виходу з акаунту;

− Get songs – цей варіант використання ініціює «User». Він забезпечує можливість пошуку пісень відповідно до поточних погодних умов;

− Get clothes – цей варіант використання ініціює «User». Він забезпечує можливість пошуку одягу відповідно до поточних погодних умов;

− Add new songs – цей варіант використання ініціює «Administrator». Він забезпечує можливість додавання нових пісень до бази даних;

− Add new clothes – цей варіант використання ініціює «Administrator». Він забезпечує можливість додавання нового одягу до бази даних;

− Delete songs – цей варіант використання ініціює «Administrator». Він забезпечує можливість видалення пісень з бази даних;

− Delete clothes – цей варіант використання ініціює «Administrator». Він забезпечує можливість видалення одягу з бази даних;

2.8 Розробка алгоритму надання персоналізованих рекомендацій щодо одягу на основі зібраних даних про погоду

Для реалізації надання рекомендацій щодо одягу, який би підійшов під погоду було розроблено наступний алгоритм:

Крок 1. Отримати детальні дані про поточну погоду та прогноз про її найближчі зміні.

Крок 2. За допомогою інформації про поточну температуру визначити колір та теплоту одягу. Чим нижча температура – тим більш теплий одяг та темніший колір будуть підбиратися.

Крок 3. Проаналізувати наявності та імовірність дощу чи снігу. У випадку опадів порекомендувати парасольку чи водонепроникну одежу.

Крок 4. Якщо хмарність дуже мала, а температура вища за 25 градусів за Цельсієм – порекомендувати сонцезахисні окуляри та головний убір.

2.9 Розробка алгоритму надання персоналізованих рекомендацій щодо музики на основі зібраних даних про погоду

Для реалізації надання рекомендацій щодо музики для поточної погоди було розроблено наступний алгоритм:

Крок 1. Отримати детальні дані про поточну погоду та прогноз про її найближчі зміні.

Крок 2. За допомогою інформації про поточну хмарність визначити динамічність пісні. При малій хмарності рекомендувати більш динамічні пісні.

Крок 3. За допомогою інформації про поточні опади визначити настрій пісні. При відсутності опадів пісні будуть підбиратися більш веселі та легкі; у випадку дощу пісні будуть більш меланхолічні.

Крок 4. Обрати пісні, що відповідають отриманому опису. При виборі композицій на пріоритетність впливає їхня популярність.

3 РЕАЛІЗАЦІЯ ПОСТАНОВКИ ЗАДАЧІ ТА ТЕСТУВАННЯ

3.1 Розробка програмного забезпечення для серверної частини

3.1.1 Розробка каркасу системи

Для повноцінної роботи Spring Boot застосунку необхідно розробити правильну структуру, в яку входять такі пункти, як налаштування maven, підключення необхідних залежностей для роботи, створення структури директорій, базове налаштування Spring та створення основних класів для роботи.

Для вебзастосунку були обрані такі залежності:

- spring-boot-starter-thymeleaf;
- spring-boot-starter-web;
- spring-boot-devtools;
- spring-boot-starter-test;
- spring-boot-starter-data-jpa;
- mysql-connector-java;
- lombok;
- jackson-databind;
- json;
- jakarta.servlet-api;
- gson;
- jsoup.

Завдяки цим бібліотекам відкриваються для розробки такі функції, як підключення HTML файлів до проєкту(thymeleaf), налаштування та робота з базами даних(data-jpa, mysql-connector), полегшення в створенні нових класів та робота з ними(lombok), надсилання запитів до серверів та отримання відповіді(jsoup), парсинг даних(json, gson).

Далі необхідно створити директорії, в яких знаходитимуться майбутні класи. Загалом в Spring Boot проєкті повинні бути такі основні директорії:

– Controllers;
– Models;
– Repositories;
– Services;
– Resources.

В цих директоріях будуть класи, які працюватимуть з базою даних, обмінюватимуться з клієнтом та містити клієнтські файли.

Далі для роботи необхідно налаштувати вхідну точку в застосунок – main.

Лістинг 3.1 Вхідна точка в застосунок:

```
@SpringBootApplication
public class WeatherStyleApplication {
   public static void main(String[] args) {
      SpringApplication.run(WeatherStyleApplication.class, args);
   }
}
```

В даному випадку вказано клас, який є основним в проєкті і від якого залежатиме правильний запуск застосунку.

Для роботи з базою даних необхідно вказати налаштування бази даних в спеціальному файлі – application.properties. В ньому вказується адреса бази даних, логін та пароль, також, додаткова інформація, яка необхідна для роботи з базою даних.

Лістинг 3.2 Налаштування бази данх:

```
spring.jpa.hibernate.ddl-auto = update
```

```
spring.datasource.url =
   jdbc:mysql://${MYSQL_HOST:localhost}:3306/weatherstyle
spring.datasource.username =    root
spring.datasource.password =    root
spring.jpa.show-sql=true
spring.main.allow-circular-references=true
spring.jpa.properties.hibernate.connection.characterEncoding=UTF-8
spring.jpa.properties.hibernate.connection.useUnicode=true
server.port=5050
```

На цьому основні налаштування Spring закінчуються і необхідно переходити до наступного кроку – створення моделей для бази даних. Для веб застосунку для надання персональних рекомендацій щодо одягу і музики відносно погоди необхідно всього дві моделі (дві таблиці) – songs та clothes, в яких зберігатимуться пісні та одяг відповідно.

Лістинг 3.3 Модель Song в Spring Boot:

```
@Table(name = "songs")
@Entity
@AllArgsConstructor
@NoArgsConstructor
@Data
public class Song {
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private Long id;
   private String name;
   private String preview;
   private String url;
   private PrecipitationType precipitationType;
```

```
    private TemperatureRange temperatureRange;
    private WeatherCondition weatherCondition;
    private WindIntensity windIntensity;
}
```

Як можна побачити в цій моделі використовуються аннотації:

– @Table – назва таблиці в базі даних;

– @Entity – клас є моделлю бази даних;

– @AllArgsConstructor – створення конструктора зі всіма полями;

– @NoArgsConstructor – створення конструктора без параметрів;

– @Data – створення геттерів та сеттерів;

– @Id – поле позначається ідентифікатором;

– @GeneratedValue – ідентифікація буде від 1 і кожен рядок буде мати n+1 id.

Також, в цьому класі використовуються об’єкти enum-класів Java:

– PrecipitationType – містить інформацію про опади;

– TemperatureRange – містить інформацію про температуру;

– WeatherCondition – містить інформацію про погоду в загальному;

– WindIntensity – містить інформацію про швидкість вітру.

Для роботи з цією моделлю необхідно створити інтерфейс-репозиторій, який буде обмінюватись з базою даних.

Лістинг 3.4 Інтерфейс-репозиторій для обміну з базою даних:

```
public interface SongRepository extends JpaRepository<Song, Long> {
}
```

І для того, щоб працювати з цим інтерфейсом розроблено сервіс, який містить в собі всі необхідні методи для роботи.

Лістинг 3.5 Сервіс для роботи з інтерфейсом *SongRepository*:

```
@Service
@Slf4j
@RequiredArgsConstructor
public class SongService {
    private final SongRepository songRepository;

    public Song getById(Long id){
        return songRepository.findById(id).orElse(null);
    }

    public List<Song> getAll(){
        return songRepository.findAll();
    }

    public Song save(Song song){
        return songRepository.save(song);
    }

    public void delete(Song song){
        songRepository.delete(song);
    }
}
```

Аннотації:

– Service – клас є сервісом;

– Slf4j – логування всіх дій з базою даних;

– RequiredArgsConstructor – генерує конструктор, який приймає всі обов’язкові поля класу як аргументи.

Методи:

- getById – отримати пісню за її id;
- getAll – отримати всі пісні;
- save – зберегти/оновити пісню в базі даних;
- delete – видалити пісню з бази даних.

По такому ж принципу розроблена і система для одягу.

Лістинг 3.6 система для одягу:

```
@Entity
@Table(name = "clothing")
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Clothes{
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private Long id;
   private String name;
   private String image;
   private ClothingType clothingType;
   private PrecipitationType precipitationType;
   private TemperatureRange temperatureRange;
   private UVIndex uvIndex;
   private WeatherCondition weatherCondition;
   private WindIntensity windIntensity;
}
```

3.1.2 Розробка методів для отримання погодної інформації

Для отримання інформації від користувача необхідно розробити контролер, який буде обмінюватись з ним. У випадку з цим застосунком, це буде RestController.

Лістинг 3.7 реалізація RestController:

```
@RestController
@RequiredArgsConstructor
@RequestMapping("/api")
public class APIController {
    private final SongService songService;
    private final ClothingService clothingService;

    @GetMapping("/get-info")
    public DataToUser getInfo(@RequestParam Double latitude,
@RequestParam Double longitude){
        WeatherDTO userWeather = WeatherHelper.getWeather(latitude,
longitude);
        System.out.println(userWeather);
        Song song = MusicHelper.getSongByWeather(userWeather,
songService);
OutfitDTO outfit =
        ClothingHelper.getOutfit(userWeather,clothingService);
        System.out.println(outfit);

        return ConverterDTO.convertToData(userWeather, song, outfit);
    }
}
```

Метод *getInfo* отримує від користувача його координати, після чого викликає метод *WeatherHelper.getWeather()*, якому передає координати користувача. Метод *getWeather()* в свою чергу використовує відкрите API – OpenWeatherMap. Це онлайн-платформа, яка надає погодні дані та сервіси з прогнозу погоди для різних місць по всьому світу. Їх API дозволяє розробникам отримувати доступ до цих погодних даних та інтегрувати їх у свої програми, сайти або застосунки.

Ключові можливості та функції OpenWeatherMap API:

– поточна погода. Можна отримувати поточні погодні дані для конкретного місця, включаючи температуру, вологість, швидкість вітру та інші параметри;

– прогноз погоди. API дозволяє отримати прогноз погоди на певний період часу у майбутньому для обраного місця;

– історичні дані. Отримати історичні дані про погоду;

– карти погоди. OpenWeatherMap API також має можливість генерувати карти погоди для відображення на сайті або застосунку;

– астрономічні дані. отримати інформацію про сонячні та місячні засічення, фази місяця та інші астрономічні дані.

Щоб використовувати OpenWeatherMap API, потрібно зареєструватись на їх вебсайті та отримати API-ключ. Після цього можна здійснювати запити до API, використовуючи цей ключ.

Лістинг 3.8 Запит до OpenWeatherMap:

```
String apiUrl = "https://api.openweathermap.org/data/3.0/onecall?lat=" +
    latitude + "&lon=" + longitude + "&exclude=minutely&appid=" +
    apiKey;
URL url = new URL(apiUrl);
HttpURLConnection connection = (HttpURLConnection)
    url.openConnection();
connection.setRequestMethod("GET");
```

На основі координат робиться запит по API до OpenWeatherMap, де збираються дані, після чого відправляються користувачу в Json форматі. Коли дані отримуються, вони автоматично розпарсюються в клас Weather, в якому зберігаються, доки їх не отримає користувач. Після парсингу, відповідь повертається в метод контролера, з якого був викликаний, – *getInfo*, де далі вже на їх основі обирається одяг та пісня.

3.1.3 Розробка алгоритмів для отримання рекомендацій з одягу

Як згадувалось вище, в вебзастосунку використовуються enum-класи, які відповідають за деяку погодну інформацію. В цьому пункті вони будуть розглянуті і описані, відповідно до їх роботи з класами, які працюють з базою даних. Всього є шість класів: *ClothingType*, *PrecipitationType*, *TemperatureRange*, *UVIndex*, *WeatherCondition* та *WindIntensity*;

ClothingType – містить інформацію про вид одягу, який зберігається в базі даних, та має такі варіації:

- HEAD – головний убір;
- EYEWEAR – окуляри;
- OUTER – верхній одяг;
- SWEATER – кофта;
- T_SHIRT – футболка;
- BOTTOMS – нижня частина;
- FOOTWEAR – взуття.

PrecipitationType – містить інформацію про опади та містить такі варіанти:

- RAIN – дощ;
- SNOW – сніг;
- HAIL – град;

– ICE – лід;

– WET_SNOW – мокрий сніг;

– NONE – без опадів.

TemperatureRange – містить інформацію про температуру, яку встановлює в таких об’єктах:

– VERY_COLD – дуже холодно;

– COLD – холодно;

– MODERATE – середня;

– WARM – жарко;

– VERY_WARM – дуже жарко.

UVIndex – містить інформацію про УФ-індекс та має такі варіації:

– LOW – низький;

– MEDIUM – середній;

– HIGH – високий;

– VERY_HIGH – дуже високий;

– EXTREMELY_HIGH – екстремально високий.

WeatherCondition – містить інформацію про тип погоди та має такі варіації:

– SUNNY – сонячно;

– CLOUDY – хмарно;

– FOG – туман;

– RAIN – дощ;

– SNOW – сніг;

– THUNDERSTORM – гроза;

– WINDY – вітер.

WindIntensity – містить інформацію про вітер та має такі варіації:

– CALM – спокійний;

– MODERATE – помірний;

– STRONG – сильний;

– STORMY – бурховий.

Отже, на моменті, коли отримується відповідь від OpenWeatherMap, дані перетворюються на об’єкти цих enum-класів для подальшої роботи з алгоритмами сортування одягу та пісень. В даному пункті буде описана робота з алгоритмами одягу.

Для того, щоб підібрати необхідний одяг, з того ж контролера, що описувався вище, відбувається виклик методу *ClothingHelper.getOutfit()*, якому передається об’єкт з погодою.

Лістинг 3.9 виклик метода *ClothingHelper.getOutfit*:

```
OutfitDTO outfit = ClothingHelper.getOutfit(userWeather, clothingService);
```

Після чого в цьому методі створюється загальний об’єкт класу для повного образу – *OutfitDTO* та з бази даних надходить інформація про всі елементи одягу, які в ній присутні.

Лістинг 3.10 отримання повної інформації про елементи одягу:

```
OutfitDTO outfitDTO = new OutfitDTO();
List<Clothing> headList =
   clothingService.getByClothingType(ClothingType.HEAD);
List<Clothing> outerList =
   clothingService.getByClothingType(ClothingType.OUTER);
List<Clothing> sweaterList =
   clothingService.getByClothingType(ClothingType.SWEATER);
List<Clothing> tShirtList =
   clothingService.getByClothingType(ClothingType.T_SHIRT);
List<Clothing> bottomsList =
   clothingService.getByClothingType(ClothingType.BOTTOMS);
List<Clothing> footList =
   clothingService.getByClothingType(ClothingType.FOOT);
```

Для сортування одягу відповідно до погодних умов, використовується компаратор з Java Collection. У Java компаратор (Comparator) - це інтерфейс, який дозволяє порівнювати об’єкти та визначати їх порядок сортування. В основному використовується для сортування списків об’єктів за визначеними критеріями.

Основні методи інтерфейсу Comparator:

– compare(T obj1, T obj2): цей метод приймає два об’єкти для порівняння та повертає ціле число: від’ємне число, якщо obj1 менше, ніж obj2.0, якщо obj1 рівний obj2. позитивне число, якщо obj1 більше, ніж obj2;

– equals(Object obj): цей метод перевіряє, чи є данний об’єкт рівним іншому об’єкту Comparator.

Для порівняння елементів одягу було розроблено власний компаратор, який порівнює по кількості співпадінь enum-об’єктів одягу та погоди. Основним методом цього класу є метод countMatches, який рахує кількість співпадінь та повертає їх, після чого відбувається сортування від більшого, до меншого.

Лістинг 3.11 реалізація підрахунку кількості співпадінь одягу та погоди:

```
private int countMatches(Clothing clothing) {
    int matches = 0;
    if (clothing.getPrecipitationType() ==
        weatherDTO.getPrecipitationType()) {
            matches++;
        }
    if (clothing.getTemperatureRange() ==
        weatherDTO.getTemperatureRange()) {
            matches++;
        }
    if(clothing.getUvIndex() == weatherDTO.getUvIndex()){
        matches++;
```

```
    }
    if (clothing.getWeatherCondition() ==
        weatherDTO.getWeatherCondition()) {
            matches++;
        }
    if (clothing.getWindIntensity() == weatherDTO.getWindIntensity()) {
        matches++;
    }
    return matches;
}
```

Після сортування даних відбувається створення образу об’єкта OutfitDTO.

Лістинг 3.12 Створення образу об’єкта OutfitDTO:

```
outfitDTO.setHead(headList.get(0));
if(userWeather.getWeatherCondition() == WeatherCondition.SUNNY)
outfitDTO.setGlasses(clothingService.getByName("Сонцезахисні
окуляри"));
else if(userWeather.getWeatherCondition() == WeatherCondition.SNOW)
outfitDTO.setGlasses(clothingService.getByName("Вітрозахисні
окуляри"));
if(userWeather.getTemperature() < 10 ||
userWeather.getWeatherCondition() == WeatherCondition.RAIN)
outfitDTO.setOuter(outerList.get(0));if(userWeather.getTemperature() < 20)
outfitDTO.setSweater(sweaterList.get(0));
outfitDTO.setShirt(tShirtList.get(0));
outfitDTO.setBottoms(bottomsList.get(0));
outfitDTO.setFoot(footList.get(0));
```

Як можна зрозуміти з коду вище, в об’єкт *outfitDTO* встановлюється по одному елементу одягу з усіх можливих. Також, є перевірка на те, чи є необхідність в сонцезахисних окулярах – якщо в *WeatherCondition* встановлений об’єкт *SUNNY*, значить рекомендуємо користувачу взяти сонцезахисні окуляри, якщо ж погода сніжна (*SNOW*) – вітрозахисні окуляри. Схожим чином працює і верхній одяг: якщо температура менше 10 або на вулиці йде дощ, рекомендуємо одягти верхній одяг.

Після того, як всі дані відсортовані і зібрані, вони повертаються назад до методу контролера, з якого були викликані.

3.1.4 Розробка алгоритмів для отримання рекомендацій з музики

Завдяки enum-класам, описаним вище, працює і алгоритм з музикою. Для кожної пісні встановлюються свої рекомендації, відповідно до яких пісню рекомендує користувачу.

Лістинг 3.13 Виклик метода *MusicHelper.getSongByWeather*:

```
Song song = MusicHelper.getSongByWeather(userWeather, songService);
```

В цьому методі відбувається схожий процес, як і з алгоритмом одягу, але більш полегшений: отримується список пісень, сортується відповідно до об’єктів enum-класів, береться пісня з найбільшою кількістю співпадінь і повертається в контролер.

Лістинг 3.14 Пошук пісні з найбільшою кількістю співпадінь:

```
List<Song> songs = songService.getAll();
SongComparator comparator = new SongComparator(userWeather);
Collections.sort(songs, comparator);
return songs.get(0);
```

Для сортування пісень розроблено власний компаратор – *SongComparator*, який містить в собі два методи: *compare* та *countMatches*.

Лістинг 3.15 Реалізація *SongComparator*:

```
public class SongComparator implements Comparator<Song> {
    private WeatherDTO weatherDTO;
    public SongComparator(WeatherDTO weatherDTO) {
        this.weatherDTO = weatherDTO;
    }
    @Override
    public int compare(Song song1, Song song2) {
        int matchesSong1 = countMatches(song1);
        int matchesSong2 = countMatches(song2);
        int matchesComparison = Integer.compare(matchesSong2,
    matchesSong1);
        if (matchesComparison != 0) {
            return matchesComparison;
        }
        return song1.getName().compareTo(song2.getName());
    }
    private int countMatches(Song song) {
        int matches = 0;
        if (song.getPrecipitationType() ==
            weatherDTO.getPrecipitationType()) {
                matches++;
            }
        if (song.getTemperatureRange() ==
            weatherDTO.getTemperatureRange()) {
                matches++;
            }
```

```
        if (song.getWeatherCondition() ==
            weatherDTO.getWeatherCondition()) {
                matches++;
            }
        if (song.getWindIntensity() == weatherDTO.getWindIntensity()) {
            matches++;
        }
        return matches;
    }
}
```

Працює схожим чином до компаратора одягу – рахується кількість співпадінь до погоди, та сортується від більшого до меншого, після чого повертається список в метод, з якого був викликаний.

Після того, як метод контролера – *getInfo* отримав дані про погоду, дані про музичні рекомендації та рекомендації одягу, він об’єднує всі ці дані в один об’єкт – *DataDTO*.

Лістинг 3.16 Реалізація *SongComparator*:

```
@Data
public class DataDTO {
    private double temperature;
    private double uvi;
    private int clouds;
    private double wind_speed;
    private double popi;
    private String rain;
    private String songName;
    private String songUrl;
    private String songPreview;
```

```
    private OutfitDTO outfit;
}
```

В цьому об’єкті зберігаються дані про погоду, рекомендовану пісню та образ, які необхідно передати користувачу. Створення об’єкта класу *DataDTO* відбувається в класі *ConverterDTO*, який перетворює всі дані в один об’єкт.

Лістинг 3.17 реалізація методу *convertToData*:

```
public static DataToUser convertToData(WeatherDTO userWeather, Song
    song, OutfitDTO outfit){
    DataToUser dataToUser = new DataToUser();
    dataToUser.setTemperature(userWeather.getTemperature());
    dataToUser.setUvi(userWeather.getUvi());
    dataToUser.setClouds(userWeather.getClouds());
    dataToUser.setWind_speed(userWeather.getWind_speed());
    dataToUser.setPopi(userWeather.getPop());
    dataToUser.setRain(userWeather.getRain() > 0 ? "Так" : "Ні");

    dataToUser.setSongName(song.getName());
    dataToUser.setSongUrl(song.getUrl());
    dataToUser.setSongPreview(song.getPreview());
    dataToUser.setOutfit(outfit);
    return dataToUser;
}
```

І після цих маніпуляцій дані відправляються до користувача, де потім виводяться в HTML через JavaScript.

3.1.5 Огляд методів для обміну з фронтендом

Загалом для роботи з сервером розроблено три контролери, один з яких розглянуто в попередніх підрозділах. Окрім нього є такі контролери:

– ClothingAPIController;

– SongAPIController.

Ці два контролери мають по одному методу для завантаження нових даних до бази даних, у випадку з першим контролером – створення нових елементів одягу, у випадку з другим – створення нових пісень.

Лістинг 3.18 Метод *ClothingAPIController*:

```
@PostMapping("/create")
public void create(@RequestBody ClothingDTO clothingDTO){
   if(clothingDTO != null){
      Clothing clothing = new Clothing();
      clothing.setName(clothingDTO.getName());
      clothing.setImage(clothingDTO.getImage());
clothing.setClothingType(ClothingType.valueOf(clothingDTO.getClothingTy
pe()));
clothing.setPrecipitationType(PrecipitationType.valueOf(clothingDTO.getP
recipitationType()));
clothing.setTemperatureRange(TemperatureRange.valueOf(clothingDTO.ge
tTemperatureRange()));
clothing.setUvIndex(UVIndex.valueOf(clothingDTO.getUvIndex()));
clothing.setWeatherCondition(WeatherCondition.valueOf(clothingDTO.get
WeatherCondition()));
clothing.setWindIntensity(WindIntensity.valueOf(clothingDTO.getWindInten
sity()));
      clothingService.save(clothing);
   }  }
```

Отже, це POST метод, який отримує в аргументах об’єкт класу *ClothingDTO*, після чого збирає дані з цього об’єкту і на його основі створює новий рядок в таблиці бази даних – *Clothes*.

Метод *SongAPIController* працює інакше: він отримує об’єкт класу *SongDTO*, після чого перевіряє, чи є в ньому посилання на пісню, якщо є, то він робить запит до Deezer і розпаршує ці дані для об’єкту *Song*, який в майбутньому буде рядком в таблиці Songs.

Лістинг 3.19 Запит до Deezer:

```
String endpoint = "https://api.deezer.com/track/" + songDTO.getId();
HttpClient client = HttpClient.newHttpClient();
HttpRequest request = HttpRequest.newBuilder()
    .uri(URI.create(endpoint))
    .build();
```

Після того, як отримує результат в форматі Json – request, він розпаршується наступним кодом.

Лістинг 3.20 Обробка результату запиту:

```
if (response.statusCode() == 200) {
    Gson gson = new Gson();
    JsonObject jsonObject = gson.fromJson(response.body(),
JsonObject.class);
    System.out.println("code 200");
    if (jsonObject.has("title")) {
        System.out.println("has title");
        String title = jsonObject.get("title").getAsString();
        String link = jsonObject.get("preview").getAsString();
        if (jsonObject.has("album")) {
            System.out.println("has album");
```

```
            JsonObject albumObject =
        jsonObject.get("album").getAsJsonObject();
            if (albumObject.has("cover_big")) {
                System.out.println("has cover big");
                String preview = albumObject.get("cover_big").getAsString();
                Song song = new Song();
                song.setName(title);
                song.setUrl(link);
                song.setPreview(preview);
        song.setPrecipitationType(PrecipitationType.valueOf(songDTO.getPrecipita
        tionType().toUpperCase()));
        song.setTemperatureRange(TemperatureRange.valueOf(songDTO.getTempe
        ratureRange().toUpperCase()));
        song.setWeatherCondition(WeatherCondition.valueOf(songDTO.getWeather
        Condition().toUpperCase()))
        song.setWindIntensity(WindIntensity.valueOf(songDTO.getWindIntensity().t
        oUpperCase()));
                songService.save(song);
                System.out.println(song.toString());
            }
        }
      }
        } else {
           System.out.println("Помилка: " + response.statusCode());
        }
```

З цього коду можна зробити такий порядок дій:

– перевірка, чи прийшла відповідь від сервера Deezer (перевірка на код 200);

– перевірка на те, чи є в цій відповіді назва пісні(*jsonObject.has("title")*);

– перевірка на те, чи є назва альбому(**jsonObject.has(“album”)**);

– перевірка на те, чи містить відповідь зображення пісні(*albumObject.has(“cover_big”)*).

Якщо по всім пунктам отримується позитивна відповідь, створюється об’єкт класу Song, який заповнюється цими даними та даними про погоду від адміністратора, та зберігається у вигляді нового рядка в таблиці бази даних – Songs.

3.2 Розробка клієнтської частини вебзастосунку

3.2.1 Верстка головної сторінки

Для розробки клієнтської сторони було обрано чистий HTML + CSS + JavaScript, як було сказано вище. По дизайну розроблено три головні блоки на сторінці: блок з погодою, блок з рекомендацією по музиці, блок з рекомендаціями по одягу.

Лістинг 3.21 Код HTML:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial
scale=1.0">
    <title>WeatherStyle</title>
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min
css">
    <link rel="stylesheet" href="styles.css">
</head>
```

```
<body>
<header class="text-center mb-4">
   <h1 class="display-4">WeatherStyle</h1>
</header>
<div class="container" id="container">
</div>
<script type="module" src="../js/main.js"></script>
</body>
</html>
```

З цього коду можна зробити наступні висновки:

– весь код генерується з JavaScript;

– для стилів використовується Bootstrap.

У світі веброзробки існує безліч інструментів і фреймворків, які обіцяють полегшити життя програмістів і забезпечити швидкий та ефективний розвиток вебсайтів. Один із таких невід'ємних інструментів – Bootstrap. Bootstrap є відкритим і безкоштовним фреймворком для розробки вебсайтів і вебзастосунків, який став вкрай популярним серед розробників усього світу.

Переваги використання Bootstrap очевидні:

– швидкість розробки. Bootstrap надає готові компоненти і стилі, які можна легко використовувати у ваших проектах. Це значно прискорює процес розробки, оскільки не потрібно писати кожен стиль або компонент з нуля;

– адаптивність. Bootstrap надає масштабовану сітку (grid system), яка дозволяє вебсайту автоматично адаптуватися до різних розмірів екранів, включаючи мобільні пристрої. Це дозволяє створювати зручні та користувацько-орієнтовані застосунки для будь-яких пристроїв;

– консистентність. Використання Bootstrap забезпечує консистентний вигляд і поведінку вашого вебсайту на різних платформах і браузерах. Це робить проект більш професійним і допомагає уникнути проблем з відображенням;

– розширюваність. Bootstrap має велику спільноту користувачів і велику кількість розширень та плагінів, які можна легко інтегрувати у проекти. Це дозволяє розширювати функціональність вебсайту з мінімальними зусиллями;

– документація. Bootstrap має добре структуровану та зрозумілу документацію, яка допомагає новачкам швидко освоїти фреймворк і використовувати його на повну потужність.

Отже, використання Bootstrap є доцільним для будь-якого веброзробника, оскільки він дозволяє ефективно розвивати проекти, забезпечуючи при цьому високу якість та професійний вигляд вебсайтів і застосунків.

Як сказано вище, код має три блоки, отже, блок для погоди містить в собі елемент section, в якому має набір тексту для відображення погоди.

Лістинг 3.22 Блок для погоди:

```
<section class="weather-card">
   <div class="card">
      <div class="card-body">
         <h2 class="card-title">Погодні умови</h2>
         <p class="card-text">Температура: C</p>
         <p class="card-text">Хмарність: p>
         <p class="card-text">Швидкість вітру: км/год</p>
         <p class="card-text">Ультрафіолетовий індекс: p>
         <p class="card-text">Ймовірність опадів: p>
         <p class="card-text">Чи йде дощ: Ні</p>
      </div>
   </div>
</section>
```

Блок з піснею містить в собі зображення альбому, назву пісні та плеєр для прослуховування пісні.

Лістинг 3.23 Блок з піснею:

```
<section class="song-card">
   <div class="card">
      <img src= class="card-img-top song-img" alt="Пісня" style="max"
width: 250px">
      <div class="card-body">
         <h2 class="card-title></h2>
         <audio controls>
            <source src=‘‘ type="audio/mpeg">
            Your browser does not support the audio element.
         </audio>
      </div>
   </div>
</section>
```

Блок з виведенням рекомендацій щодо одягу більш складний, так як він повинен мати динамічний розмір, в залежності від кількості елементів одягу, і виглядає наступним чином.

Лістинг 3.24 Блок з виведенням рекомендацій:

```
<h2 class="card-title">Одяг відповідно до погодних умов</h2>
<div class="row" id="outfit-row">
</div>
```

Сам блок має вигляд контейнера, в якому динамічно виводяться інші контейнери.

Лістинг 3.25 Контейнер блоку:

```
<div class="card">
   <img src=‘‘ class="card-img-top">
   <div class="card-body">
      <h2 class="card-title"></h2>
   </div>
</div>
```

Цей елемент містить в собі зображення та назву елемента одягу. Таким чином зверстана головна сторінка вебзастосунку. Як тільки юзер відкриває застосунок, він надає свої координати серверу, після чого на сервері відбувається обробка даних та повертається відповідь користувачу. Більш детально про те, як працює JavaScript код в наступному пункті.

3.2.2 Розробка JavaScript коду для обміну з бекендом

Для коректної роботи фронтенду використовується динамічний вивід елементів з коду JavaScript. При переході користувача на вебзастосунок, отримується його геолокація та передається до сервера.

Лістинг 3.26 Код отримання геолокації:

```
let options = {
   enableHighAccuracy: true,    timeout: 5000,
   maximumAge: 0
};

if ("geolocation" in navigator) {
```

```
    navigator.geolocation.getCurrentPosition(successCallback,
errorCallback, options);
} else {
    console.log("Геолокація не підтримується у вашому браузері");
}

function successCallback(position) {
    let latitude = position.coords.latitude;
    let longitude = position.coords.longitude;
    console.log("Широта: " + latitude + ", Довгота: " + longitude);

    const url = `/api/get-info?latitude=${latitude}&longitude=${longitude}`
    fetch(url,{
        method: 'GET',
    })
```

Цей код використовує метод *navigator.geolocation.getCurrentPosition()* для отримання точних координат звідки робиться запит. Після чого через fetch робить запит до бекенду і очікується результат у Json форматі, який після цього оброблюється.

Лістинг 3.27 Обробка результату методу отримання точних координат:

```
.then(response => response.json())
.then(data => {})
```

Після того, як дані з сервера надходять до скрипту, відбувається перевірка, чи вони не пусті. Далі створюється контейнер, в який виводяться всі дані про погоду та музику.

Лістинг 3.28 Обробка результату методу отримання точних координат:

```
const container = document.getElementById("container")
container.innerHTML = `
<section class="weather-card">
    <div class="card">
        <div class="card-body">
            <h2 class="card-title">Погодні умови</h2>
            <p class="card-text">Температура:
${data.temperature.toFixed(2)}°C</p>
            <p class="card-text">Хмарність: ${data.clouds}</p>
            <p class="card-text">Швидкість вітру: ${data.wind_speed}
км/год</p>
            <p class="card-text">Ультрафіолетовий індекс: ${data.uvi}</p>
            <p class="card-text">Ймовірність опадів: ${data.popi}%</p>
            <p class="card-text">Чи йде дощ: Ні</p>
        </div>
    </div>
</section>

<section class="song-card">
    <div class="card">
        <img src='${data.songPreview}' class="card-img-top song-img"
alt="Пісня" style="max-width: 250px">
        <div class="card-body">
            <h2 class="card-title">${data.songName}</h2>
            <audio controls>
                <source src='${data.songUrl}' type="audio/mpeg">
                Your browser does not support the audio element.
            </audio>        </div>
```

```
    </div>
</section>
`
```

Для виведення користувачу елементів одягу використовується інша система, з перевірками на те, чи є той чи інший елемент одягу в отриманих даних. Якщо є, то цей елемент виводиться.

Лістинг 3.29 Приклад роботи системи:

```
const head = data.outfit.head
const rowBlock = document.getElementById('outfit-row')
if(head){
    const headBlock = document.createElement('div')
    headBlock.className = 'col-md-4'    headBlock.innerHTML = `
    <div class="card">
        <img src='${head.image}' class="card-img-top">
        <div class="card-body">
            <h2 class="card-title">${head.name}</h2>
        </div>
    </div>
    `
    rowBlock.appendChild(headBlock)
}
```

Отже, з цього коду можна зрозуміти, що є перевірка на те, чи не пустий об'єкт для головного убору, а далі вже звичайний вивід в HTML, як і в коді вище.

На цьому інтерфейс користувача закінчується. Для того, щоб користувач міг отримати нову інформацію, йому необхідно оновити сторінку. Якщо ж погода зміниться, зміняться і дані.

3.3 Тестування програмного забезпечення

Тестування вебзастосунків є важливою складовою розробки, оскільки воно допомагає впевнитися в якості та надійності програмного забезпечення перед випуском його в продакшн. Тестування включає в себе створення та виконання тестових сценаріїв, що перевіряють різні аспекти застосунку, від функціональності до продуктивності та безпеки. Воно допомагає виявити та усунути помилки та дефекти ранніх стадіях розробки, що зменшує витрати на виправлення проблем у майбутньому. Тестування також сприяє покращенню якості коду та збільшенню довіри користувачів до продукту, що відображається на його популярності та успіху на ринку. В цілому, тестування є важливою практикою, яка сприяє розробці високоякісного та надійного програмного забезпечення.

У випадку з вебзастосунком, що надає рекомендації відносно погоди, тестування відбувалось завдяки підстановки різних координат. Тобто з Google Мар беруться координати, далі ці дані підставляються при відправленні запиту на сервер і аналізується інформація. До аналізу інформації входять такі етапи:

– перевірка погоди в регіоні з різних джерел;

– перевірка отриманої інформації та те, які саме об’єкти enum-класів обираються;

– аналіз пісні з бази даних, чи підходить вона за всіма параметрами;

– аналіз одягу, чи підходить він за всіма параметрами;

– загальна оцінка роботи сервера.

З рисунку 3.1 можна зробити висновки, що на вулиці доволі прохолодна температура та йде дощ; велика хмарність та доволі сильний вітер. Далі відбувається перевірка та аналіз пісні, яку рекомендує сервер. Результатом цього має бути підібрана пісня та одяг відповідно до погодних умов. Пісня, наприклад, може бути пов’язана з дощем, а одяг має бути доволі теплим та водонепроникним, скоріш за все темних відтінків, переважно чорного та сірого кольорів.

Рисунок 3.1 – Виведення інформації про погоду для користувача

Відповідно до погоди (рис. 3.1) було обрано пісню «Singing in the Rain» (рис. 3.2). Для точного аналізу необхідно отримати всі дані про пісню з бази даних.

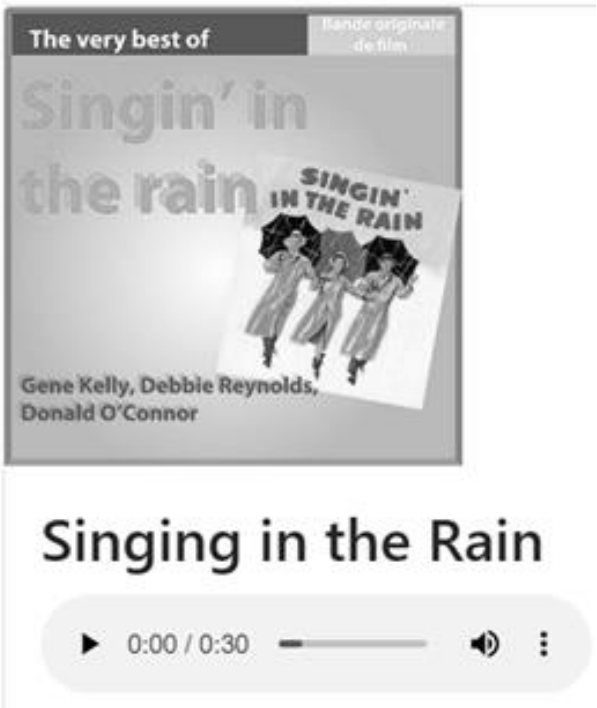

Рисунок 3.2 – Пісня рекомендована сервером

Після отримання інформації відбувається перевірка, чи підходить ця пісня за всіма параметрами погоди за enum-класами. В даному випадку сервер спрацював ідеально та надав пісню, яка мала найбільшу кількість співпадінь з базою даних, та містила такі дані enum-класів, як низька температура чи дощ.

Точно таким методом відбувається і аналіз одягу, який рекомендує сервер. На рисунку 3.3 можна побачити, які елементи одягу рекомендує сервер. Після чого відбувається аналіз кожного елементу в базі даних.

Рисунок 3.3 – Рекомендації одягу відповідно до погодних умов

Вищевказаний аналіз підтверджує важливість тестування вебзастосунків для забезпечення їх якості та надійності перед випуском у продакшн. Цей процес, дозволяє виявляти та усувати помилки на ранніх стадіях розробки, зменшуючи витрати на подальше виправлення. Конкретний приклад аналізу погодних умов та відповідних рекомендацій, який включає перевірку інформації з різних джерел, аналіз музичних та одягових варіантів, ілюструє ефективність тестування у підтримці функціональності та коректності вебзастосунків.

ВИСНОВКИ

У цій кваліфікаційній роботі було проведено аналіз предметної області та постановка задачі з метою розробки вебзастосунку для надання персональних рекомендацій щодо одягу та музики відносно погоди. Під час аналізу були розглянуті сучасні тенденції в галузі рекомендаційних систем, технологічні інновації в аналізі погодних умов, підходи до аналізу погодних даних та їх вплив на емоційний стан користувачів, а також аналіз індивідуальних вподобань користувачів.

На основі проведеного аналізу були сформульовані основні завдання та цілі розробки вебзастосунку, а також поставлено задачі для досягнення визначених цілей дослідження. Для реалізації поставлених завдань та досягнення цілей були обгрунтовані вибрані методи та технології, такі як моделювання залежності між погодними умовами та вибором одягу, аналіз музичних вподобань в залежності від погоди, технологія розробки серверної частини Spring Boot, вибір стеку для фронтенду та інші.

Під час реалізації поставлених задач було розроблено програмне забезпечення для серверної та клієнтської частини вебзастосунку, здійснено тестування програмного забезпечення, що дозволило перевірити його функціональність та відповідність поставленим вимогам.

Отже, результатом даної роботи є розроблений вебзастосунок, який забезпечує користувачів персоналізованими рекомендаціями щодо одягу та музики відносно погодних умов, а також відповідає сучасним тенденціям в галузі рекомендаційних систем та технологічним інноваціям в аналізі погодних даних.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. McSherry, D. (2019). "Personalized Weather Recommendations Using Machine Learning." Medium.

2. Kantor, J. (2021). "Building a Weather App with Spring Boot and Angular." Baeldung.

3. Smola, A., & Narasimhan, H. (2020). "Machine Learning for Personalized Recommendations." arXiv preprint arXiv:2006.13025.

4. Gupta, P. (2018). "Building a Weather App with React." freeCodeCamp.

5. Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to Information Retrieval. Cambridge University Press.

6. Lee, J., & Lee, K. (2017). "A Survey of Recommender Systems Research." Korean Journal of Applied Statistics, 30(1), 169-203.

7. Zhang, Y., & Mao, K. (2019). "A Survey of Weather Data Analysis and Forecasting Methods." IEEE Access, 7, 43483-43501.

8. Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A. R., Jaitly, N., ... & Kingsbury, B. (2012). "Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups." IEEE Signal Processing Magazine, 29(6), 82-97.

9. Вечірська, І. Д., Черноусова, М. С., & Вечірська, А. Д. (2021). Побудова логічної мережі для опису процесу кореспонденції балансових рахунків обліку депозитів суб’єктів господарської діяльності та фізичних осіб.

10. Вечірська, І. Д., Гончаров, І. Е., & Хамітов, Т. М. (2015). Побудова логічної мережі для діагностики та управління надзвичайними ситуаціями.

11. Вечірська, І., Кобилін, О., Прокоп'єв, С., Вечірська, А., & Кучеренко, М. (2022). BUILDING A LOGICAL NETWORK FOR SOLVING THE PROBLEM OF CAR RENTAL BY MEANS ALGEBRA OF FINITE PREDICATES. *Computer systems and information technologies*, (2), 78-87.

12. Gorokhovatskiy, V. A., Vechirska, I. D., & Chetverikov, G. G. (2016). Method for building of logical data transform in the problem of establishing links between the objects in intellectual telecommunication systems. Telecommunications and Radio Engineering, 75(18).

13. Вечірська, А., & Вечірська, І. (2021). Аналіз застосування інструментарію алгебри предикатів для оцінення якості складної системи. *Збірник наукових праць ΛΌГОΣ*.

14. Vechirska, A., Shyrokorad, K., & Vechirska, I. (2022). VISUAL MODELING OF CHILD REGISTRATION TO KINDERGARTEN PROCESS. *Collection of scientific papers «SCIENTIA»*, (May 27, 2022; Stockholm, Sweden), 73-75.

15. Vechirska, A., Shyrokorad, K., & Vechirska, I. (2022). VISUAL MODELING OF PURCHASE PROCESS MANAGEMENT IN THE ELECTRONICS ONLINE STORE. *Матеріали конференцій МНЛ*, (3 червня 2022 р., м. Львів), 189-192.

16. Vechirska, A., Shyrokorad, K., & Vechirska, I. (2023). SOLVING THE PROBLEM OF CHOOSING THE BEST ASSEMBLY PLAN FOR SOFTWARE DEPLOYMENT IN THE CLOUD ENVIRONMENT USING THE ANALYTIC HIERARCHY PROCESS. *Матеріали конференцій МНЛ*, (23 червня 2023 р., м. Дніпро), 126-129.

17. Chetverikov, G., Puzik, O., & Vechirska, I. (2016, September). Multiple-valued structures of intellectual systems. In *2016 XIth International Scientific and Technical Conference Computer Sciences and Information Technologies (CSIT)* (pp. 204-207). IEEE.

18. Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., & Fei-Fei, L. (2014). "Large-scale Video Classification with Convolutional Neural Networks." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1725-1732.

19. Zhou, Y., Wilkinson, D., Schreiber, R., & Pan, R. (2018). "Large-Scale Parallel Collaborative Filtering for the Netflix Prize." International Conference on Algorithmic Applications in Management, 337-348.

20. Lam, H., Ho, H., & Tompkin, J. (2011). "Weather-informed Clothing Recommendations." Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 1083-1092.

21. An, J., & Cho, S. (2015). "Combining Multiple Recommendations for Clothing Style Coordination." Expert Systems with Applications, 42(22), 8604-8614.

22. Lam, H., & Riedl, J. (2007). "Shilling recommender systems for fun and profit." Proceedings of the 16th international conference on World Wide Web, 393-402.

23. Bao, S., & Intille, S. (2004). "Activity recognition from user-annotated acceleration data." Pervasive Computing, 3001, 1-17.

24. Breiman, L. (2001). "Random forests." Machine learning, 45(1), 5-32.

25. Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.

26. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

27. Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer Science & Business Media.

28. Murphy, K. P. (2012). Machine Learning: A Probabilistic Perspective. MIT press.

29. Sutton, R. S., & Barto, A. G. (2018). Reinforcement Learning: An Introduction. MIT Press.

30. Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. O'Reilly Media, Inc.