Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____ННЦ ЗФН_____
(повна назва)

Кафедра _____Програмної інженерії_____
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

_____другий (магістерський)_____
(рівень вищої освіти)

Дослідження методів класифікації тексту та їх застосування у інформаційних технологіях
(тема)

Виконав: студент 2 курсу, групи ПЗСзм-18-1
спеціальності _121- Інженерія програмного забезпечення_____
(код і повна назва спеціальності)
освітньо-професійної програми _Програмне__
_____забезпечення_систем_____
(повна назва освітньої програми)

_____Вірстюк С.Ю._____
(прізвище, ініціали)

Керівник _____проф. Шубін І.Ю._____
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри, проф. _____ З.В.Дудар

2019 р.

Харківський національний університет радіоелектроніки

Факультет _Комп'ютерних наук_____

Кафедра _Програмної інженерії_____

Рівень вищої освіти _____другий (магістерський)_____

Спеціальність _121– Інженерія програмного забезпечення_____
(код і повна назва)

Освітньо-професійна програма _Програмне забезпечення систем_____
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 20 ___ р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

Студентові _Вірстюку Сергію Юрійовичу_____
(прізвище, ім'я, по батькові)

1. Тема роботи _Дослідження методів класифікації тексту та їх застосування у інформаційних технологіях_____

затверджена наказом по університету від «___» _____ 2019 р № __Стз__
заповнюється вручну після отримання наказу

2. Термін подання студентом роботи до екзаменаційної комісії
10 грудня 2019 р.

3. Вихідні дані до роботи __*проаналізувати існуючі алгоритми, що використовуються для вимоги до веб-сервісу, що буде розроблено, середовище проектування Microsoft VisualStudio 2017, мова розробки C#, сервер IIS Express*_____

4. Перелік питань, що потрібно опрацювати в роботі _*мета роботи, аналіз проблемної галузі і постановка задачі, опис запропонованих варіантів оптимізації, використовувані методи та алгоритми, опис розробленої програмної системи, опис застосованих оптимізацій, аналіз можливих застосувань*_____

5. Консультанти розділів роботи

Найменування розділу	Консультант (посаду, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Спецчастина	проф. Шубін І.Ю.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1.	Аналіз предметної галузі	10 жовтня 2019 р.	
2.	Огляд існуючих методів	27 жовтня 2019 р.	
3.	Проектування та розробка ПЗ	15 листопада 2019 р.	
4.	Підготовка пояснювальної записки	25 листопада 2019 р.	
5.	Спецчастина	26 листопада 2019 р.	
6.	Підготовка презентації та доповіді	30 листопада 2019 р.	
7.	Попередній захист	10 грудня 2019 р.	
8.	Нормоконтроль, рецензування	11 грудня 2019 р.	
9.	Занесення диплома в електронний архів	12 грудня 2019 р.	
10.	Допуск до захисту в зав. кафедри	14 грудня 2019 р.	
* заповнюється вручну після виконання чергового пункту			

Дата видачі завдання 2019 р.

Студент _____
(підпис)

Керівник роботи _____ __проф. Шубін І.Ю._____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до атестаційної роботи: 86 с., 37 рис., 3 додатки, 22 джерела.

ЗНАННЯ, АЛГОРИТМ, СИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ, МОДЕЛЮВАННЯ, ПРОГРАМНА БІБЛІОТЕКА, РІВНЯННЯ.

Об’єктом дослідження є методи розробки систем підтримки прийняття рішень.

Метою даної роботи є моделювання методів та розробка програмної системи, що надає публічний API до алгоритму нейромережевої класифікації тексту.

Методи розробки базуються на методах математичного моделювання та методах моделювання нейронних мереж.

У результаті роботи розроблена математична модель веб-сервісу для бізнесу зацікавленого у покращенні ефективності свого відділку кадрів, а також для рекрутингових агенцій, що мають великий потік листівок на свої електронні адреси.

KNOWLEDGE, ALGORITHM, DECISION SUPPORTING SYSTEMS, MODELING, SOFTWARE LIBRARY, EQUATIONS.

The object of the study is the methods of developing decision support systems.

The purpose of this work is to model methods and develop a software system that provides a public API to the neural network algorithm for text classification.

Development methods are based on mathematical modeling and neural network modeling methods.

As a result, a mathematical model of a web service was developed for a business interested in improving the efficiency of its HR department, as well as for recruiting agencies with a large flow of leaflets to their email addresses.

ЗМІСТ

ВСТУП

Успішними і конкурентоспроможними на сучасному ринку є компанії, що використовують всі резерви для підвищення ефективності своєї діяльності і зменшення неефективних витрат. Автоматизація простих бізнес-процесів є одним із найперспективніших та ефективних методів отримання найвищих показників праці та прибутку. Саме тому нові технології, які спроможні вирішити цю задачу, користуються попитом на ринку вже сьогодні.

Компанії усіх сфер так чи інакше працюють з інформаційними системами. Вони допомагають збільшити прибуток за рахунок збільшення клієнтської бази та зменшення витрат. Усередині цих систем використовується багато складних інтелектуальних алгоритмів, що спочатку були відкриті та доведені науковими дослідженнями. Автоматизація аналітики, виконання складних розрахунків, логістичні задачі, задачі розпізнавання і т.д. – завдяки автоматизації цих речей, компанії економлять чималий процент бюджету.

Розвиток компанії, як правило, означає потребу в збільшенні кількості персоналу, а також, в свою чергу, високий попит на вакантні місця. Із зростанням кількості вакансій, відділу кадрів стає все складніше працювати з потоком резюме, що надходять до компанії. Беручі до уваги сучасний стан розвитку інформаційних технологій в нашій країні, можна побачити чимало відкритих для будь-кого шляхів ознайомлення з вакантними місцями у різноманітних компаніях, що збільшує кількість претендентів на ці самі вакантні місця, адже майже кожен користується сучасними технологіями для пошуку інформації.

Одразу можна побачити декілька можливих шляхів вирішення цієї безумовно важливої проблеми, наприклад збільшити відділ кадрів або почати співпрацювати з кваліфікованим агентством кадрів. До певного моменту ці варіанти можуть показувати себе досить непогано, але зі збільшенням потоку відгуків, людський фактор може зіграти злий жарт і, наприклад, буде пропущений відгук на вакансію надзвичайно досвідченого та талановитого спеціаліста, якого,

можливо, топ-менеджмент чекав дуже багато. Все це недоотриманий прибуток, або навіть збитки для компанії. Жоден керівник не буде задоволений таким станом справ. Саме тут відповідальні люди починають міркувати над оптимізацією цього процесу, та закономірно приходять до необхідності хоча б часткової автоматизації цього бізнес-процесу, адже це найоптимальніший шлях.

Хоч на даний момент і не можливо повністю позбавитися людського фактору у цьому бізнес-процесі, але все ж таки можна перекласти частину обов’язків на обчислювальні машини. Найперспективнішим варіантом є автоматична класифікація резюме по відношенню до відкритих вакантних місць та вимог до них. Це допоможе співробітникам відділу кадрів працювати більш ефективно, адже завдяки автоматизації інформація буде структурованою по вакансіям і відслідковувати нові відгуки буде набагато простіше.

Отже тема атестаційної роботи є актуальною. Метою даної роботи є проектування та розробка програмної системи, що складається з REST-застосування, що надає публічний API до алгоритму нейромережевої класифікації тексту [1], який може бути використаний у подальшому іншими застосунками-клієнтами, та веб-клієнту, який надає основні функції для класифікації резюме претендентів на вакантні місця. Даний продукт повинен бути популярним здебільшого серед середнього та великого бізнесу, адже саме ця аудиторія є найбільш зацікавленою у автоматизації цього бізнес-процесу та зменшенні витрат за рахунок цього. Але, звісно, системою можуть користуватися і невеликі компанії. Отже, можна сказати, що потенціальний ринок даного проекту достатньо великий.

1 АНАЛІЗ СТАНУ РОЗВ'ЯЗАННЯ ПРОБЛЕМИ ТА ОБҐРУНТУВАННЯ ЦІЛЕЙ ДОСЛІДЖЕННЯ

1.1 Аналіз інформаційних лінгвістичних технологій

Поняття інформаційних технологій у лінгвістиці. Одним з основних призначень мови є його використання для передачі інформації між людьми. Тому, говорячи про мову, неможливо залишити без уваги й поняття інформації. Інформація в повсякденнім розумінні трактується як відомості про стан справ у навколишньому світі, його властивостях процесах, що протікають у ньому, і т.п. [3]. У спеціальних науках, що вивчають інформацію, це поняття визначається трохи інакше: як послідовність сигналів або символів деякого алфавіту, кодуючи деяке повідомлення без обліку значеннєвого змісту цього повідомлення (у теорії передачі інформації) або як зміст, який отриманий із зовнішнього миру й дозволяє адекватно реагувати живому організму (або технічній системі) на навколишнє середовище (у кібернетиці) [4].

Узагальнюючи різні визначення інформації, можна запровадити наступне трактування цього поняття: інформація – це свідчення про навколишній світ, передані людиною, живими організмами або технічними системами для регулювання своєї поведінки в навколишньому середовищі.

Роль інформації в сучасному суспільстві винятково велика. Інформація, кодуєма за допомогою мови, перетворюється в знання; знання ж передаються від покоління до покоління, тем самим забезпечуючи наступність суспільних підвалин.

Інформація може кодуватися вербально або невербально. Відмінність способів кодування інформації (аудітивний, тактільний, візуальний, і т.ін.) обумовлює множину засобів її представлення:

- тексти;
- малюнки, кресленники, фотографії;
- світлові або звукові сигнали;

– електричні й нервові імпульси;

– жести й міміка;

– заходи й смакові відчуття;

– хромосоми, за допомогою яких передаються в спадщину ознаки й властивості організмів, і т.д.

Способів вистави інформації, як показують приклади, досить багато. Але оскільки людей може сприймати інформацію лише за допомогою власних органів почуттів, целесообразно класифікувати види інформації саме на цьому основані. По тому, якими органами почуттів сприймаються і якою сигнальною системою закодовані відомості про навколишній світ, можна виділити звукову, смакову, тактильну й візуально-символічну інформацію. Саме останні два види інформації є найбільш значимими для сучасної людини, при цьому якщо в XX в. людина мала справу в основному візуально-образної, то в XXI сторіччі найбільш значимої стає візуально-символічна інформація.

Символ – це знак, що позначає деякий предмет або явище. У лінгвістиці символами вважаються в першу чергу слова, оскільки саме слово є мінімальною одиницею, здатною позначати предмети і явища навколишнього світу. В інформатиці символами вважаються головним чином букви, розділові знаки, цифри й інші знаки друкованого тексту, а також звукові знаки – фонеми – усного тексту, що є частинами алфавітів і фонетичних систем різних природних і штучних мов. Ці символи складаються в слова й вислови, що кодують передану інформацію.

Процеси, пов’язані з певними операціями над інформацією, називаються інформаційними процесами. У цей час над інформацією можна робити наступні операції:

– створювати ухвалювати комбінувати;

– зберігати передавати копіювати;

– шукати сприймати формалізувати;

– вимірювати використовувати ділити на частині;

– спрощувати руйнувати обробляти;

– збирати поширювати перетворювати.

Зв’язки з постійним збільшенням кількості використовуваної людьмі інформації на певному етапі розвитку суспільства потребовалось залучення спеціальних технічних засобів для її обробки й зберігання. Принципові зміни в способах фіксації й передачі інформації, пов’язані з винаходом нових технічних засобів одержали назву інформаційних революцій. Дослідниками виділяються три інформаційні революції [5]:

Третя інформаційна революція в значній мірі стимулювалася тим, що в середині XX ст. з’являються спеціальні науки, що вивчають інформацію: інформатика й кібернетика. Інформатика – це наука про накопичення, обробку й передачі інформації за допомогою ЕОМ. Наука про керування, зв’язок і переробці інформації називається кібернетикою.

Саме в рамках теорії інформації (математичної теорії зв’язків) для ілюстрації інформаційного обміну, здійснюваного за допомогою технічних засобів, К. Шенноном і У. Уивером була запропонована наочна модель.

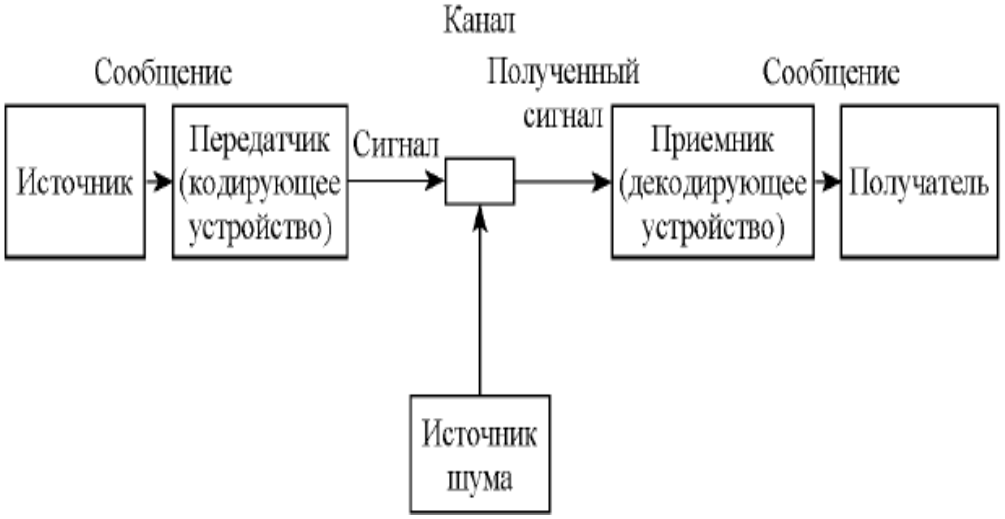

Рисунок 1.1 – Модель К. Шеннона і У. Уивера [6]

Особливо значимим для інформаційних технологій представляється вказівка в даній моделі на декодувальне обладнання, що кодує й, оскільки однієї з важливих завдань інформатики виявляється переклад інформації, закодованої в «людських» символах, в інформацію, зрозумілу комп’ютерам, і навпаки.

Комп’ютери в інформаційному обміні стають засобом кодування, обробки, зберігання й передачі більших масивів символьної інформації. Сукупність законів, методів і засобів одержання, зберігання, передачі, поширення й перетворення інформації за допомогою комп’ютерів одержало позначення «інформаційні технології».

При звуженні цього поняття для його використання в особою професійної сфері (лінгвістика) одержуємо комбінацію «інформаційні технології в лінгвістиці», що розуміється як сукупність законів, методів і засобів одержання, зберігання, передачі, розповсюдження й перетворення інформації про мову й закони його функціонування за допомогою комп’ютерів [8].

Одним із завдань відповідної області знання є співставлення способів кодування інформації людиною й комп’ютером. Під кодуванням у цілому розуміється процес представлення інформації у вигляді послідовності умовних позначок. Іншими словами, кодування – це зіставлення об’єктів і відносінь між ними із символами або словами якої-небудь мови [9].

У процесі кодування співвідношення слова (символу) і його значення звичайно називається семантикою, правила, що виражають простіші синтаксичні властивості слів і груп слів, що дозволяють перекладати й/або описувати правильні пропозиції мови – граматикою [10]. Комп’ютер може обробляти всі відомі види інформації, включаючи:

- числову;
- буквену (вербальну);
- графічну;
- звукову;
- відеоінформацію.

Інформація в комп’ютері представлена у двійковому коді, алфавіт якого складається із двох цифр (0 і 1). Так, числова інформація використовуваної людиною десяткової системи числення з’являється в ЕОМ у вигляді наступних комбінацій символів 0 і 1:

Для кодування комп’ютером вербальної інформації ізначально використовувався код ASCII (American Standard Code for Information Interchange). Для кодування одного символу в цьому коді потрібно 1 байт (або 8 бітів). У цілому в ASCII можна закодувати 256 символів, при цьому кожному символу ставиться у відповідність унікальний десятковий код від 0 до 255.

1.2 Аналіз методів виділення текстів

Але на сьогоднішній день такої проблеми з непорозумінням із-за мови немає. Це було спричинено тим, що з розвитком цивілізацій люди почали вивчати інші мови. Спочатку вивчали мову тільки так звана знать, але в сучасному світі кожна людина має змогу вивчати інші мови. Це допомагає у подорожах, у роботі, бізнесі, політиці і в інших галузях. Мова не є просто інструментом передачі інформації, це на багато більше — це розуміння культури тієї нації чию мову ми вивчаємо.

Вивчення іноземних мов на сьогодні не є чимось дивним або важким. Для цього протягом десятків років було розроблено багато методологій вивчення мов. Кожна методологія пропонує самі різноманітні способи вивчення, що говорить про те що люди можуть підібрати для себе той спосіб вивчення, який найбільш підходить для їх стилю життя. Також викладання іноземних мов на сьогоднішній день є дуже важливим процесом який дозволяє навчати людей із самих ранніх їх років. Наприклад викладання іноземних мов є в дитячих садках, школах та університетах, уже не говорячи про спеціалізовані школи, різні курси, приватні вчителі і т ін.

Зараз уже йде двадцять перше століття і чим воно може допомогти людям у вивченні іноземних мов? Двадцять перше століття прославилося бумом розвитку комп’ютерних та Веб технологій. Для вивчення мов було розроблено дуже багато додатків та сервісів. Всі вони надають велику масу контенту для вивчення, а

також допомагають різними вправами і т.ін. Але що відсутнє в даних додатках та сервісах? А відсутня змога вести свій словник, де не буде всього зайвого що зараз надають всі ці сервіси. Це начебто досить проста річ, але всі сервіси вирішують за користувачів який контент їм потрібен. Звісно, якщо людина тільки починає вчити нову мову, то вона собі не зможе підібрати потрібний контент для вивчення мови. Але коли людина має певний рівень, їй потрібно тільки те, що буде корисним тільки для неї особисто. Раніше люди вели свої словники, де записували слова, переклад, їх тлумачення, але зараз все це можна автоматизувати і реалізувати за допомогою програмних додатків.

Найбільш корисним буде те, якщо людина матиме змогу завжди отримати доступ до свого словника. А ще краще, якщо цей словник зможе зробити рутинну працю за людину, наприклад прочитати текст і підрахувати входження кожного слова в нього, щоб визначити які найбільш зустрічаються, та на які треба звернути увагу або вивчити в першу чергу. Наприклад ми хочемо прочитати книгу в оригіналі, але ми не отримаємо задоволення якщо постійно будемо дивитися в словник за новим зустрінутим словом. Найкращим виходом буде перед читанням вивчити нові слова які можуть бути в даній книзі. Тобто ми беремо книгу, віддаємо додатку і отримуємо всі слова із рейтингом який дозволить виявити нові незнайомі слова.

А як же вивчати слова? Ми ж як користувачі не будемо заучувати словник, для цього є кращий вихід — вправи. Найбільш ефективним видом вправ є знаходження перекладу до заданого слова. Ця вправа не потребує багато часу, а також є достатньо ефективною. З цих речей можна утворити достатньо зручний інструмент для вивчення або полегшення вивчення іноземних слів. А тепер візьмемо за основу те що це буде мобільний додаток, то це означає що у нас під рукою буде наш словник, наш перекладач, а також вправи які можемо виконувати в любий вільний час — все це стане не від’ємною частиною життя кожного, хто не байдужий до вивчення іноземних мов.

Отже розробка мобільного додатка є достатньо своєчасним рішення в даний час, так як даного функціоналу об’єднаного в одному додатку не має, що досить

сильно підвищує вірогідність того, що даний додаток стане досить популярним і корисним для всіх користувачів які вивчають іноземну мову

Інформаційно-технічна революція змінила характер і методи ведення бізнесу. Використання можливостей технічного обміну сьогодні дозволяє легше і швидше створювати і продавати пакети послуг споживачам, вирішувати завдання фінансово-операційного управління, маркетингового планування, підвищувати конкурентоздатність і кількість продаж.

Вже сьогодні успішність і ефективність бізнесу залежить від ефективності технологій, які допомагають підприємству перекласти найбрудніші частини роботи на комп'ютерну техніку. У цьому напрямку сфера інформаційних технологій розвивається з неймовірною швидкістю, тому потрібно постійно слідкувати за інноваціями і впроваджувати їх на своєму підприємстві.

З розвитком бізнесу майже завжди стає потреба у нових кадрах, адже забезпеченість і правильний підбір кадрів завжди були основними умовами ефективної роботи підприємства. Нові вакантні посади, звільнені старі робочі місця – на всі ці місця потрібно знайти людей і все це потребує ретельної перевірки та своєчасної обробки. Для цього створюють відділ кадрів або користуються послугами спеціальних агентств по пошуку потенційних робітників, такі агентства називають рекрутинговими. З часом даних про відгуки на вакансії накопичується все більше, потік резюме збільшується і виникає потреба у автоматизації цього бізнес-процесу. Основною частиною цього процесу є розподіл заявок по відкритим вакансіям.

Аналізуючи предметну галузь було виявлено, що не існує аналогічних систем для автоматизації саме цього бізнес-процесу. Найбільше підійшли до вимог програми поштових клієнтів, які також можна використовувати для автоматизації класифікації листівок, однак із певними доробками.

1.3 Виявлення проблем та актуалізація рішень

Говорячи про ринок інструментів для класифікації надісланих резюме, проаналізуємо аналоги. Відразу ж варто звернути увагу на велику кількість сервісів, які дозволяють сортувати і групувати листівки, які потрапляють до поштової скриньки. Істотні недоліки даних систем:

– велика узагальненість системи, що зменшує точність, а тому ефективність;

– примітивні алгоритми класифікації, де відсоток помилки великий.

Виходячи з цього, системи, які дозволяють усунути ці недоліки, матимуть велику перевагу на ринку. Аналізуючи методи класифікації тексту, можна виділити наївний баєсів класифікатор, класифікацію на основі правил та згорткові нейронні мережі.

Наївний баєсів класифікатор є одним з найбільш часто використовуваних класифікаторів. Він простий в реалізації та тестуванні, процес навчання досить ефективний в порівнянні з іншими більш складними класифікаторами. На даний момент є дуже популярним класифікатором для рішення суміжної із темою диплома задачі визначення та фільтрації спаму у поштових клієнт-серверних сервісах. На невеликих сукупностях текстових документів різниця між наївним баєсовим класифікатором і іншими набагато складнішими алгоритмами класифікації часто несуттєва, а іноді наївний баєсів класифікатор може виявитися і більш точним. Але із зростанням об’ємів інформації, баєсів класифікатор суттєво програє більш складним алгоритмам класифікації.

Класифікація на основі правил є традиційним рішенням задачі класифікації текстових повідомлень, особливо у поштових сервісах. Цей підхід може бути досить непоганим варіантом якщо ви працюєте з невеликою колекцією документів яку ви здатні охопити і ретельно проаналізувати, тому що ви чітко контролюєте правила за якими класифікатор приймає рішення. Але у цього підходу є і очевидні мінуси. Для того щоб вибрати значущі для класифікації слова

необхідно володіти експертними знаннями в предметній області. Також ніяк не завжди факт наявності або відсутності якого-небудь одного слова є вирішальним фактором для прийняття рішення.

Згорткові нейронні мережі – потужний інструмент машинного навчання, що націлений на ефективне розпізнавання і класифікацію. Успіх застосування згорткових нейронних мереж для зображень породив безліч спроб використання цього інструменту в інших завданнях. Експерименти на текстових даних великого обсягу показали, що згорткові нейронні мережі для задачі класифікації текстів дозволяють досягти рівня якості, аналогічного або кращого в порівнянні з традиційними методами. Основним недоліком нейронних мереж є те, що вони завжди дають приблизну відповідь. Вони не можуть вирішувати завдання, що містять послідовний ланцюжок кроків. Їх марно використовувати для обчислень. Але цей недолік не може проявити себе в даній задачі. Отже для даної задачі згорткові нейронні мережі будуть одним із кращих варіантів.

Для вирішення проблеми автоматичної класифікації резюме, що надходять на електронну скриньку, було розроблено веб-сервіс, в основі якого лежить ідея класифікація листів за допомогою нейронної мережі із пакету Tensorflow, які будуть використовувати словники Word2vec із тематичними наборами слів [12], що підібрані та автоматично поповнюються для кожної вакансії. Така система дозволяє усунути недоліки звичайних систем сортування листівок. Рішення проблеми із використанням машинного навчання робить дане рішення інноваційним і затребуваним на ринку.

Як вже було зазначено вище, сьогодні існує деяка кількість додатків для того, що дозволяють сортувати листи за заданими фільтрами. Був проведений детальний аналіз аналогів та виявлені їх переваги та недоліки. Деякі з них мають більш розширений перелік можливостей, деякі – менший. У переважній більшості, в них не можна настільки точно розподілити резюме за вакансіями. У процесі дослідження були виявлені додатки із схожим функціоналом, але повністю аналогічних до розробленого додатків.

Перший сервіс це «Gmail» – популярний у нинішній час поштовий клієнт який має функціонал сортування листів, але лише передбачені розробниками [2]. Існують мобільні рішення для цього продукту – для iOS та Android. Досить важливою перевагою даного програмного продукту є наявність додатків для декількох платформ та повноцінний, незалежний поштовий сервіс. У жовтні 2012 року служба «Gmail» стала найпопулярнішою в світі, обігнавши за кількістю унікальних користувачів конкуруючу «Hotmail» від Microsoft. Кількість користувачів «Gmail» перевищило 420 мільйонів чоловік. Недоліком системи у конкретній предметній галузі є обмеженість функцій по сортуванню листівок. На рис. 1.2 представлено інтерфейс сервісу «Gmail».

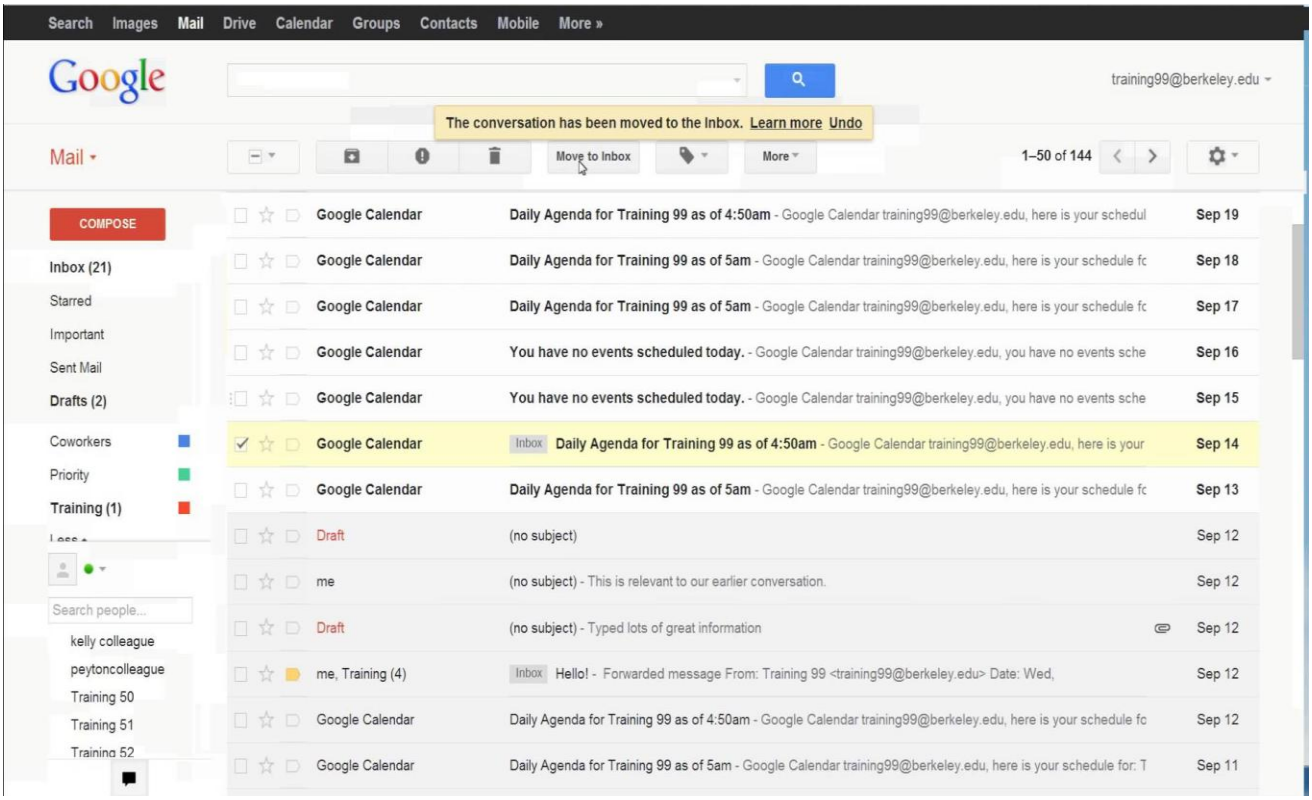

Рисунок 1.2 – Інтерфейс сервісу «Gmail»

Другий сервіс це «Outlook» – сервіс, який має додатки для усіх мобільних та десктопних платформ. Значною перевагою цього сервісу є гнучкі настройки правил сортування листівок. Також перевагою є підтримка роботи одночасно з багатьма поштовими скриньками та інтеграція між ними. Таким чином, користувач може розподіляти листівки між скриньками за заданими правилами. У

цьому сервісі користувач має можливість використовувати будь-які акаунти. Недоліком даної системи є все ж недостатня точність та самостійність сортування листівок, адже використовується розглянутий раніше алгоритм класифікації на основі правил. Але якщо брати до уваги саме реалізацію цього алгоритму, то цей сервіс має найгнучкіший і зручний варіант. На рисунку 1.3 представлено інтерфейс сервісу «Outlook».

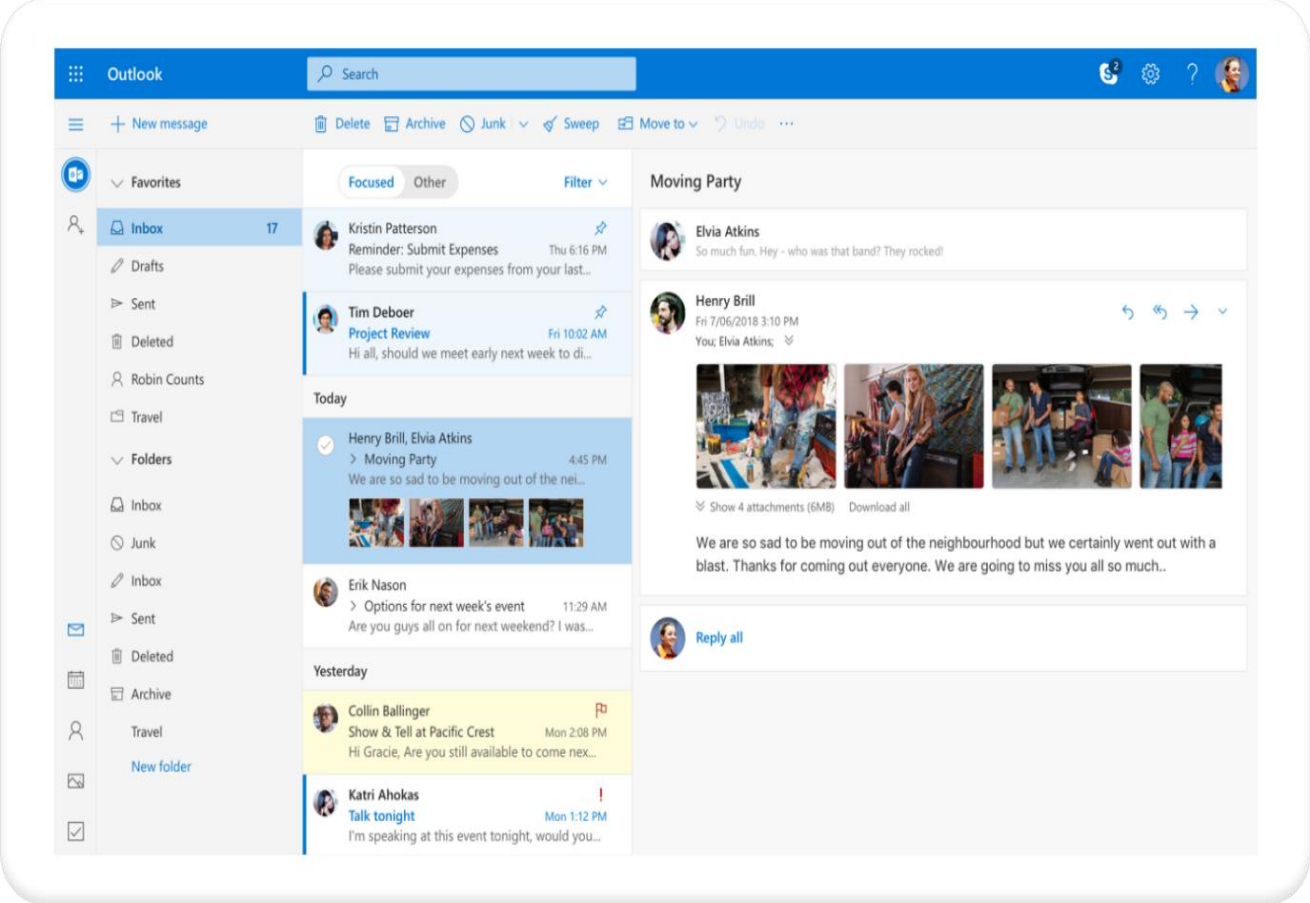

Рисунок 1.3 – Інтерфейссервісу «Outlook»

Наступним сервісом є «TheBat!». Це один з найвідоміших клієнтів для поштових скриньок із переносною версією [3]. Це дуже відомий і популярний в свій час поштовий клієнт із можливістю сортування листівок по папкам. У «TheBat!» можна налаштувати автоматичне сортування листів за заданими параметрами. Програма здатна пересортувати листи за відправником, адресатом, темою, текстом листа, тегам, розміром листа, пріоритетом, датою й іншими параметрами. Серед доступних дій — переміщення, копіювання, експорт, друк листів, видалення, автовідповідь, створення нагадування та запуск зовнішнього

застосунку. Можливо створювати загальні правила сортування, дійсні для декількох поштових скриньок. Але все це не зможе якісно вирішити проблему класифікаціїелектронних листів по вакансіям. До інших недоліків можна віднести застарілий інтерфейс, відсутність веб та мобільних додатків. На рисунку 1.4 представлено інтерфейс сервісу «TheBat!».

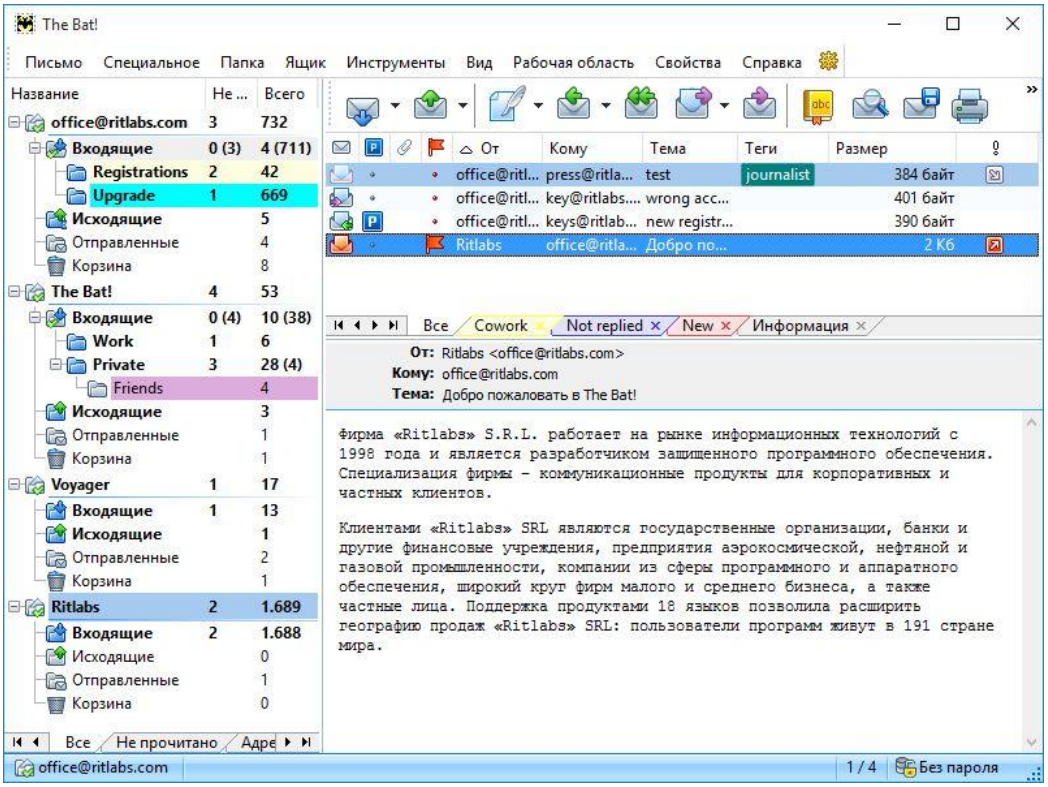

Рисунок 1.4 – Інтерфейс сервісу «Thebat!»

Із останніх open-source рішень, можна виділити «NylasMail 2.0» – поштовий клієнт із зручним та сучасним інтерфейсом. Велика добірка плагінів, включаючи засоби для автоматизованої фільтрації і класифікації пошти. Сумісний з більшістю поштових сервісів, включаючи Gmail, Yahoo, iCloud і Microsoft Exchange. Має повну підтримку операційних систем Windows та Linux, але нажаль не має адаптованої для мобільних пристроїв версії, що ускладнює роботу у сучасних реаліях.

Хоч цей продукт і має розвинуту систему класифікації завдяки розвинутій системі плагінів, все ж він дуже узагальнений щодо предметної галузі і тому буде складно створити класифікатор, що буде швидко та ефективно вирішувати такі задачі, як класифікація електронних листів за вакансіями. Але для звичайного

сортування електронних листів функціональність цього сервісу буде найбільш привабливою з усіх поштових клієнтів. На рис. 1.5 представлено інтерфейс сервісу «NylasMail 2.0».

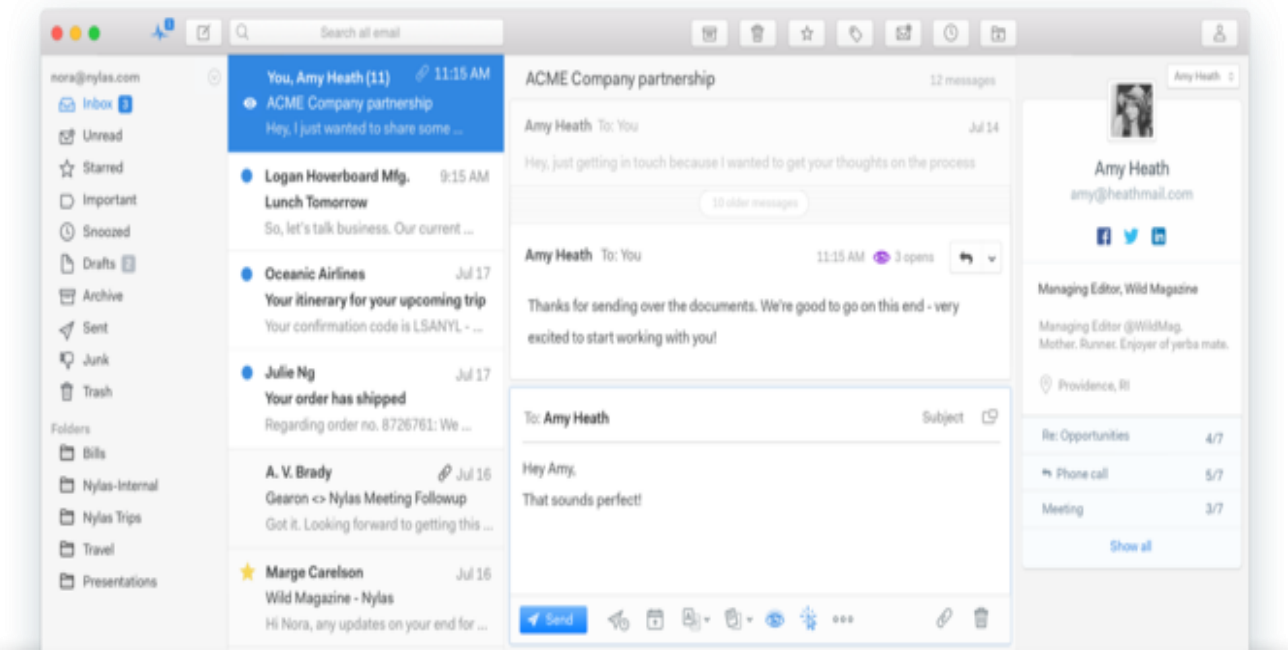

Рисунок 1.5 – Інтерфейс сервісу «NylasMail 2.0»

Після проведеного дослідження можна зробити висновок, що майже в усіх додатках досить примітивна реалізація класифікації чи сортування електронних листів, що не використовує складних алгоритмів для класифікації тексту. Це зумовлює велику кількість помилок у результатах класифікації, що створе навіть більше проблем ніж коли класифікація не використовується.

1.4 Опис мети дослідження

У атестаційній роботі необхідно спроектувати та реалізувати веб-сервіс для забезпечення автоматичного розподілу листівок із резюме по вакансіям.

Основним завданням проекту є вирішення проблеми великих витрат часу на розбір та сортування листівок у кадрових відділках та рекрутингових агентствах

за допомогою нейромережевого підходу. Створення такого веб-сервісу допоможе сконцентруватися на дійсно важливих для бізнесу справах: оцінка придатності кандидату, організація та проведення співбесід, тощо.

Одна з головних ідей при розробці такої системи полягає у використанні структурованої методології при розробці застосування роботи з поштовими заявками на існуючі вакансії. Така система повинна поєднувати у собі всі переваги та виключати недоліки вже існуючих систем.

Інтерфейс програмної системи повинен мати сучасний дизайн, бути виконаним в одному кольоровому рішенні та мати інтуїтивно-зрозумілу систему навігації, керування та відстеження для користувача.

Важливим критерієм цього сервісу є точність класифікації електронних листів за вакансіями. Для цього потрібно забезпечити постійне оновлення моделі згорткової нейронної мережі. За допомогою постійного планового оновлення, нейронна мережа буде навчатися із поповненої бази векторів слів, що буде розширюватися за рахунок нових електронних листів з резюме, що допоможе збільшувати точність моделі із кількістю оброблених даних. Використання векторів слів допоможе зробити сервіс більш точним, адже вони дозволяють зрозуміти наскільки тісний зв’язок між словами та фразами.

Користувачами даного застосування можуть бути співробітники кадрових відділень агентств усіх розмірів підприємств та рекрутингових із великими потоками електронної пошти від кандидатів на вакансії різних підприємств. Однак, потенційними клієнтами даного застосування будуть компанії середніх та великих розмірів, і великі рекрутингові агенції. Для маленьких компаній цей сервіс може бути засобом для економії, адже буде відстрочено збільшення відділку кадрів, якщо вакансій і заявок на них стане більше, а це означає відсутність потреби оплачувати додаткову робочу силу.

2 ОПИС ПРОВЕДЕНИХ ТЕОРЕТИЧНИХ ДОСЛІДЖЕНЬ

2.1 Алгоритми пошуку у корпусах текстів

Національні корпуси текстів існують також для немецкого, китайського, фінської й інших мов. Одним з перших відомих корпусів є Брауновский корпус (Brown Corpus), створений в 1963 р. у Брауновском університеті (США) для побудови частотного словника американського варіанта англійської мови. Його обсяг становив 1 млн слів. Создатели корпуса (У. Френсис і Г. Кучері) розробили строгу процедуру відбору текстів: у корпус увійшли 500 фрагментів прозаіческих текстів, створених американськими авторами й надрукованих у1961 р., по 2000 слововживань кожний. Тексти представляли собою 15 найпоширеніших жанрів інформативної й художественной прози [14].

Пошук у корпусі відповідно до запиту користувача забезпечується за допомогою спеціальних програм – корпусних менеджерів. Корпусний менеджер (англ. Corpus manager) – це специалізована пошукова система, що включає програмні середовища для пошуку даних у корпусі, одержання статистичної інформації й надання результатів користувачеві в зручній формі [15]. Результати пошуку звичайно видаються у вигляді конкорданса (тому корпусні менеджери ще називають конкордансіорами), де шукана одиниця представлена в її контекстному окруженіі з представленням частотних характеристик окремих одиниць мови, граммем і т.ін.

Таким чином, корпус, що представляє собою розмічене зібрання текстів з обсягом слів не менш 100 млн, дає широкі можливості як для прикладних (робота над принципами автоматичної розмітки), так і для дослідницьких цілей.

Комп’ютерна лексикографія являє собою розділ прикладної лінгвістики, націлений на створення комп’ютерних словників, лінгвістичних баз даних і розробку програм підтримки лексикографічних робіт.

Основними завданнями традиційної й комп’ютерної лексикографії є визначення структури словника й зон словникової статті, а також розробка принципів складання різних видів словників.

Словник традиційно визначається як організоване зібраня слів з коментарями, у яких описуються особливості структури й/або функціонування цих слів [4]. Електронний (автоматичний, комп’ютерний) словник – це збори слів у спеціальному комп’ютерному форматі, призначене для використання людиною або, що є складовою частиною більш складних комп’ютерних програм (наприклад, систем машинного перекладу). Відповідно, різняться автоматичні словники користувача людини (АСКП) і автоматичні словники для програм обробки тексту (АСПОТ) [4].

Автоматичні словники, призначені для кінцевого користувача, найчастіше є комп’ютерними версіями звичайних словників, наприклад:

– Оксфордський словник англійської мови (www.oed.com);

– автоматичний тлумачний словник англійської мови видавець ства «Коллинз» (www.mycobuild.com);

– автоматичний варіант «Нового великого англо-російського словника» під ред. Ю.Д. Апресяна і Є.М. Медниковой.

Автоматичні словники такого типу практично повторюють структуру словникової статті звичайних словників, однак вони мають функції, недоступні своїм прототипам, наприклад, використовують сортування даних по полях словникової статті (порівн. Відбір усіх прикметників), проводять автоматичний пошук усіх вокабул, що мають у тлумаченні певний семантичний компонент, і т.д. [4].

Автоматичні словники для систем машинного перекладу, автоматичного реферування, інформаційного пошуку і т.д. (АСПОТ) по інтерфейсу й структурі словникової статті суттєво відрізняються від АСКП. Особливості їх структури, сфера охоплення словникового матеріалу задаються тими програмами, які з ними взаємодіють. Такий словник може містити від однієї до сотні зон словарної статті.

Надзвичайно різноманітні й області лексикографічного опису: морфологічна, лексична, синтаксична, семантична і т.ін. [4, 5].

Структура традиційного словника зазвичай включає наступні компоненти:

– введення, що пояснює принципи користування словником і даю щее інформацію про структуру словникової статті;

– словник, що включає одиниці словника: морфеми, лексеми, слово форми або словосполучення; кожна така одиниця з відповідаю щим коментарем являє собою словникову статтю;

– покажчики (індекси);

– список джерел;

– список умовних скорочень і алфавіт [4, 5, 6].

В електронних словниках з названих компонентів обовязковим є, мабуть, лише словник, в онлайн-словниках нерідко є також алфавіт із закладеними за кожною буквою гіпер-посиланнями, ведучими до тексту словникової статті. Практично в кожному електронному словнику, пропонованому на диску (оффлайн-словник) або в Інтернеті (онлайн-словник) є функція автоматичного пошуку, що дозволяє значно заощаджувати зусилля користувача при роботі зі словником.

Відмінність електронних словників від «паперових» стосується також їх мультимедійності й гіпертекстуальності: ці властивості виражено в електронних словниках у значно більшому ступені, чому в друкованих. Так, гіперпосилання можуть бути закладені за кожним елементом словникової статті або пунктом програмного меню словника. Це дає користувачеві додаткові можливості по пошукові й швидкому перехід до необхідної словникової інформації, дозволяє знайти синоніми й антоніми до заданого слова, слова тієї ж семантичної групи, парадигми відмінювання й дієвідміни і т.ін.

Гіперпосилання дозволяють також легко зв’язувати різні словники один з одним, так що в підсумку онлайн- або офлайн-словники є колекціями або порталами словників. Одержавши необхідну інформацію, наприклад, про значення слова, користувач одним натисканням посилання може перейти до коментарів цього слова в інших словниках і довідатися особливості його

тлумачення в спеціальних галузях знання (термінологічні словники) або одержати додаткову лінгвістичну інформацію про його форму.

Окремі електронні словники мають також додаткові можливості, наприклад, електронний багатомовний словник ABBYY Lingvo (© 2008 ABBYY) надає функцію навчання (ABBYY Lingvo Tutor), що дозволяє запам'ятовувати слова, відібрані по конкретній темі й представлені парами: російське й іноземне слів, становити нові словники й словникові картки, зберігати ре зультати навчання у файл і т.д.

У підсумку структура електронного словника в значній мірі відрізняється від структури словника друкованого, хоча основна частина словника – словник зі словниковими статтями – продовжує складати ядро словника в обох випадках.

Структура словникової статті досить типова й звичайно включає наступні зони словникової статті, актуальні як для традиційної, так і для комп'ютерної лексикографії:

- лексичний вхід (вокабула, лема);
- зона граматичної інформації;
- зона стилістичних позначок;
- зона значення;
- зона фразеологізмів;
- зона етимології;
- зона прикладу й джерела прикладу.

Правда, можна виділити зони словникової статті, обов'язкові для всіх словникових одиниць, і факультативні зони. Обов'язковою зоною словникової статті для різних видів словників є лише лексичний вхід, усі інші зони залежать від типу словника: на приклад, для тлумачного словника необхідна зона значення, а для орфоепічного вона необов'язкова. Зона фразеології відсутня у коментарях слів, не використовуваних у стійких комбінаціях, а наявність зони прикладу і його джерела залежить від принципів, що є в основі створення словника.

Кількість зон словникової статті комп'ютерного словника звичайні але перевищує кількість зон словникової статті «паперового» словника, що обумовлене значними

ресурсами пам’яті й високою швидкістю обробки цифрової інформації сучасними комп’ютерами. Але обсяг пропонованої словникової інформації повинен відповідати виду словника: якщо читачеві потрібно вимова, то «зайва» інформація про переклад слова, що перевіряється, або його контекстних значеннях буде тільки заважати користувачеві.

Класифікацію комп’ютерних словників можна здійснювати на тих же принципах, що й класифікацію звичайних словників. Традиційно виділяються лінгвістичні, енциклопедичні й попередні (лінгвокраїнознавчі й термінологічні) словники. У лінгвістичних словниках описуються самі слова – їх значення, особливості вживання, структурні властивості, сполучуваність, з відношення з лексичними системами інших мов і т.д. В енциклопедічних словниках описуються поняття, факти й реалії світу, тобто екстралінгвістична інформація. Проміжний тип словників включає інформацію й лінгвістичного, і екстралінгвістичного роду [5].

Серед лінгвістичних словників можна виділити декілька їх видів [6]:

– розумні, що мають метою тлумачення (пояснення) значень слів і їх вживання в мові, що включають дескриптивні й нормативні словники, які, крім того, можуть бути загальними й частками, серед останніх виділяються, наприклад, фразеологічні словники, словники іноземних слів і т.д.;

– словники-тезауруси, що відрізняються розташуванням словникової статті, яке підлегло не алфавітному, а тематичному принципу, наприклад, тезаурус ідіоматики включає семантичне поле «ВІДХІД, ВІД’ЇЗД, ВТЕЧА», яке поміщена в категорію «РУХ», семантичне поле «ДАВНО» розміщено в категорію «ЧАС» і т.д. [6];

– двомовні (перекладні) словники, наприклад, «Англо-російський словник» В.К. Мюллера (1-е видання з’явилося в 1943 р.), «Французско-російський словник активного типу» під ред. В.Г. Гака й Ж. Триомфа й ін.;

– асоціативні словники, об’єктом яких є сфера асоціативних відносин у лексиці; словникова стаття такого словника включає лексему-стимул і список упорядкованих по частоті й алфавіту (із вказівкою частоти) реакцій, отриманих у

психолінгвістичному експерименті, наприклад: «Асоціативний тезаурус сучасної мови» [19];

– історичні й етимологічні словники, що надають інформацію про історію слів, починаючи з певної дати протягом деякого періоду, із вказівкою виникнення слів і значень, їх відмиранні й видозміні, або такі, що пояснюють походження слів;

– словники язикових форм, які фіксують особливості форми слів і в яких тлумачення значень відсутні або грають допоміжну роль, наприклад, орфографічні й орфоепічні, словотворчі й морфемні (показують, як слова складаються з морфем і інвентаризують їх), граматичні (інформація з кожного слова, що дозволяє по будувати будь-яку граматично правильну форму), інверсійні словники;

– словники мовного вживання: словники труднощів і сочетаемости слів;

– ономастикони: антропонимічні словники й топонімічні словники;

– нетрадиційні, що піддають словниковому опису нетіпичні лінгвістичні об’єкти, наприклад, «Словник політичних метафор» А.Н. Баранова і Ю.Н. Караулова [5], словники поетичних метафор, епітетів, авторські словники й словники конкордансів.

Комп’ютерні словники звичайно створюються на базі корпусів текстів з використанням засобів автоматичної обробки й пошуку словникових одиниць. Для цього залучаються спеціальні програми – бази даних, комп’ютерні картотеки, програми обробки тексту, які дозволяють автоматично формувати словникові статті, зберігати словникову інформацію й обробляти її. Так, створення електронного словника, включає наступні етапи [4, 14]:

– формування корпуса текстів і паралельне створення словника;

– автоматичне формування корпуса прикладів;

– написання словникових статей;

– уведення словникових статей у базу даних (БД);

– редагування словникових статей у БД;

– коректура тексту в БД;

– породження тексту словника й формування оригінал-макета;

– друк (розповсюдження) словника.

Звичайно, наведений опис процесу створення електронного словника може коректуватися залежно від його виду, дослідницьких принципів і інших факторів, порівняльні коментарі розробників електронного історичного словника [8]. Але у будь-якому випадку використання комп'ютерів і вже готових корпусів текстів у комп'ютерній лексикографії дозволяє зменшити кількість етапів у процесі створення електронного словника й зекономити час практично на кожному з них. Так, замість створення словникової картки в комп'ютерній лексикографії використовуються бази даних. Записи баз даних дають можливість автоматично сортувати масив по обраних пари метрам, відбирати потрібні приклади, поєднувати їх у групи і т.д. Спеціалізованих програмних оболонок для лексикографічних цілей на ринку практично немає. Для цих цілей цілком підходять сучасні бази даних. Для пошуку прикладів творці словників можуть використовувати комп'ютерні програми побудови конкордансів, наприклад, DIALEX. Для створення оригінал-макета (верстки) словників залучаються видавницькі системи типу Page-Maker або Winword, які дозволяють приписувати стилі зонам словникових статей, алфавітізацію, створення покажчиків і т.д. [8].

2.2 Алгоритми складання словників

Приклад спеціалізованої комп'ютерної програми, призначеної для комп'ютерних лексикографічних робіт, є «Програма автоматизованого складання й обробки словників» [6]. Електронні словники мають позитивні сторони не тільки у процесі їх створення, але й у процесі використання. Зокрема, виділяються наступні переваги у використанні електронних словників [7]:

– електронні словники дозволяють по-різному представити зміст словникової статті (різні «проекції» словника), у тому числі за допомогою

різноманітних графічних і мультимедійних засобів, які не використовуються у звичайних словниках;

– у видаваній інформації знаходять висвітлення різні технології комп’ютерної лінгвістики, наприклад морфологічний, синтаксичний аналіз, повнотекстовий пошук, розпізнавання і синтез звуку й т.п.;

– стає можливим швидко одержати інформацію, яка втримується десь у надрах словника й безпосередньо відповідає тому запиту, який сформульований користувачем у зручній для нього формі;

– електронний словник дозволяє швидко реагувати на зміни в мові й світі, і випуск кожної наступної його версії або внесення змін в онлайн.

3 АНАЛИЗ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

3.1 Опис розроблених алгоритмів

В ході розробки моделі майбутньої програмної системи було спроектовано діаграми, які найкраще показують структуру системи, її поведінку та функціонал.

Діаграма використання (Use Case) має одного актора (див. рис.3.1). На діаграмі відображено всі можливі дії користувача в розроблюваному додатку.

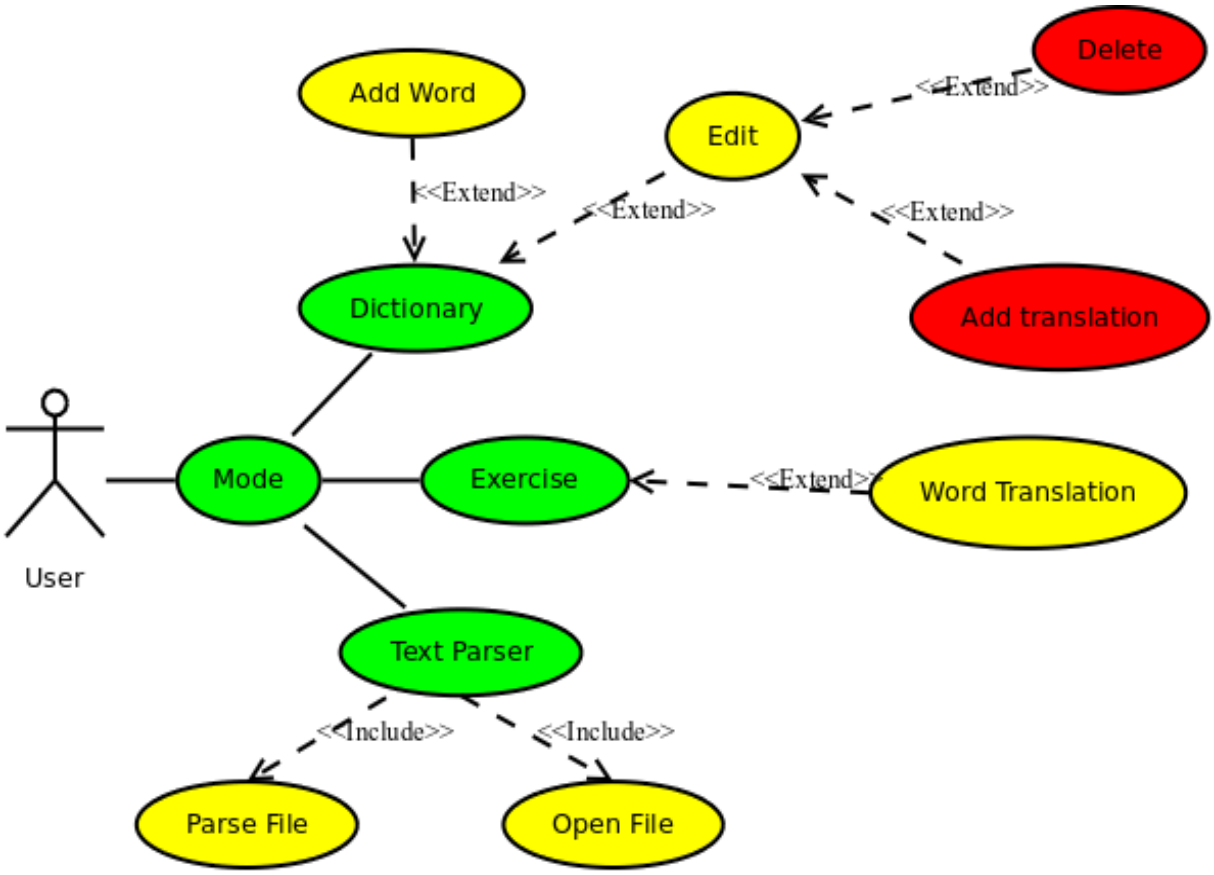

Рисунок 3.1 – Діаграма варіантів використання

Для розробки мобільного додатку було вирішено розділити код та класи на декілька рівнів. Найвищий рівень ієрархії включає такі пакети як: core, dependencies, ui. Тобто на найвищому рівні ми одразу вирішили що вся логіка буде зосереджена під пакетом core. Під пакетом dependencies будуть зосереджені всі залежності які ми будемо потребувати в деяких класах. І останній пакет з

даного рівня є пакет з реалізацією графічного інтерфейсу користувача, а також всі необхідні утилітні класи для реалізації інтерфейсу користувача.

Наступним рівнем ієрархії додатку є розділення core на наступні під-пакети: commonutils, local, webpart. Пакет commonutils включає в собі всі необхідні, загальні утилітні методи, які використовуються у різних частинах додатку. Пакет local включає в собі всі функції яким не потрібне підключення до мережі Інтернет, тобто вони виконуються локально на мобільному пристрої. Даний пакет підрозділяється ще на декілька під-пакетів як: dataaccess, database, exercises, parsing. Пакет dataaccess маэ в собі тільки ті функції, які працюють з файлами на мобільному пристрої, наприклад як відкриття та зчитування інформації із текстового файлу. Пакет database включає в себе тільки функції по роботі з локальною базою даних, а також підрозділяється ще на два пакети як dao та domain. Пакет dao включає в себе так звані дао класи, які реалізують стандартний набір функцій по роботі з базою даних, тобто зчитування, видалення, створення та редагування сутностей. Пакет domain включає в собі POJO класи, які представляють сутності бази даних.

Пакети exercises та parsing включають в себе функції вправ та розбору текстів відповідно. Останній під-пакет пакета local є пакет webpart, який включає в себе всі функції по роботі із віддаленим сервером.

Наступним після пакета local йде пакет ui. Так як додаток розроблюється для мобільної платформи Android, то це значить що в будь-якому випадку ми матимемо справу із наступними класами які допоможуть реалізувати графічний інтерфейс користувача, а саме: Activity, Dialog, Fragment, AsyncTask. Ці класи є основними будівельними матеріалами графічного інтерфейсу на даній платформі. Activity представляє із себе головне вікно додатку. Раніше кожен новий екран в додатку був представлений окремим Activity, але на сьогодні є механізм який дозволяє робити все це за допомогою так званих фрагментів. Fragment це саме та частина вікна додатку, з якою користувач має пряму взаємодію. Усі фрагменти розміщуються у головному класі Activity, це дає змогу економити ресурси, тому що не потрібно створювати важкі вікна (Activity), а створювати тільки більш легкі

фрагменти (Fragment) які розміщуються в головному вікні і без проблем можуть бути замінені.

Клас Dialog дозволяє створювати діалогові вікна, які надають змогу надати необхідну інформацію користувачеві. Наприклад при обробці результаті запиту користувач зможе спостерігати за теперішнім станом додатку через ProgressDialog.

При розробці інтерфейсу користувача потрібно створити клас AsyncTask. Оскільки операційна система Android була спроектована так, що головний потік бере на себе відповідальність за роботу з графічним інтерфейсом користувача, тому ми не можемо виконувати тяжкі задачі в головному потоці додатку, бо це може негативно сказатися на продуктивності графічного інтерфейсу аж навіть до повного його зависання. Тому щоб цього не трапилося ми повинні використовувати паралельні потоки, які будуть робити тяжку роботу і не завдавати шкоди графічному інтерфейсу. Але з побічних потоків ми не можемо напряму звертатися до графічного інтерфейсу, тобто до головного потоку, бо це є обмеженнями самої платформи. Для цього і було розроблено AsyncTask який надає змогу реалізувати цей зв’язок між головним та побічними потоками.

Також в пакеті ui є під-пакети як utils та interfaces. Перший включає в себе всі необхідні функції утиліт для роботи з графічним інтерфейсом. Другий же пакет включає в себе інтерфейси, які необхідні для взаємодії між графічним інтерфейсом та бізнес-логікою додатку. На рис. 3.2 наведено усю основну структуру пакетів додатку.

Вся структура додатку роздроблена на маленькі частини, що зменшує плутанину і покращує загальну структуру додатку. Це відгороджує нас від створення бізнес-логіки додатку в класах графічного інтерфейсу. Тому більшою частиною ми досягли створення додатку який реалізую паттерн MVC, точніше більш адаптовану його версію MVP.

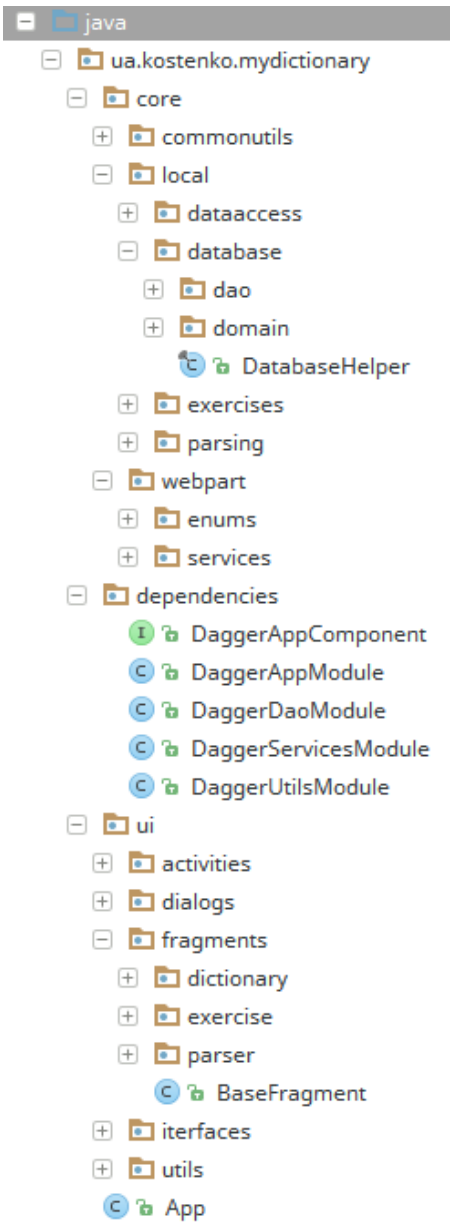

Рисунок 3.2 — Структура класів додатку

Це також є залежним від даної платформи. Дана платформа дозволяє робити всю розмітку графічного інтерфейсу користувача в xml файлах, а обробка всіх подій відбувається в класах Activity та Fragment, що більше підходить для паттерна Model View Presenter, де так званий контролер тісно зв’язаний із графічним інтерфейсом. Тобто в даному випадку View передає всі події до Presenter’а, який в свою чергу вже має певну логіку для обробки сирих даних і передачі їх до моделі. Звісно дана платформа не надає змогу чистої реалізації із коробки, але все ж таки та модель яка лежить в основі побудови Android додатків найбільш ближча за своєю поведінкою та ідеєю до патерна MVP.

3.2 Вдосконалений алгоритм класифікації тексту

В першу чергу треба чітко розділити з чого складається даний додаток. Перша і головна частина це мобільний додаток, про який вже було неодноразово згадано. Друга і не менш важлива частина це веб-додаток. Веб-додаток робить ту частину роботи, яка не помітна користувачеві на перший погляд, але насамперед без веб-додатку користувач не зміг би отримати переклад потрібного слова або фрази. Тому вся ця задача була сконцентрована у веб-додатку, що дає декілька загальних переваг. Однією із цих переваг є те, що на веб-сервері ми можемо зберігати для подальшого використання запрошені користувачем слова. Так як даний додаток використовує зовнішній API, то це робить додаткову затримку при очікуванні перекладу, тому набагато краще якщо раз переклад був отриманий із зовні, то його потрібно залишити для подальшого використання, це дозволить зменшити для всіх наступних аналогічних запитів затримку отримання відповіді.

Також веб-сервер бере на себе задачу розбору відповідей від зовнішніх API та приводить їх до того виду, який вже можна буде опрацювати на мобільному додатку. І остання досить важлива річ це те, що при зміні API перекладача, користувач мобільним додатком ніколи не помітить, що щось змінилося, бо мобільний додаток є незалежним від того API який використовує веб-сервер.

Для створення веб частини даного додатку було вирішено обрати JavaEE та Spring Framework. Це дозволяє в досить короткі строки розробити веб частину додатку, яку в подальшому буде досить легко змінювати. Обрання Spring Framework також нам надає ті можливості, які в чистому JavaEE не доступні, такі наприклад як інверсія управління, а конкретно впровадження залежностей, просте створення веб-сервісів за рахунок анотацій фреймворку, які беруть на себе більшу частину рутинної роботи, як наприклад реалізація методів doGet() та doPost() в сервлетах, а також фільтри які направляють запит до конкретного сервлету. Всі ці дії можуть бути замінені однією анотацією над класом або методом. І одна із найкращих речей у Spring Framework це робота з базами даними за допомогою

прошарку даного фреймворка, який бере на себе всі рутинні речі як встановлення підключення до бази даних, або приведення сирих типів до потрібних. Також спрощується написання Dao класів, бо фреймворк вже має шаблони які також спрощують написання цих класів.

В ролі бази даних було обрано NoSql базу даних MongoDb. Це ґрунтується на тому, що кожне слово може досить часто змінюватися і структуру даного об’єкта набагато легше змінювати ніж в реляційній базі даний, що може призвести до втрати всіх даних, чого не можу бути при використанні NoSql баз даних. Також NoSql бази даних дозволяють набагато простіше реалізувати повнотекстовий пошук, що являється дуже хорошою річчю при роботі з даними, які в першу чергу являються текстовими. До речі говорячи, всі дані які поміщаються в базу даних MongoDb називаються документами.

В ролі хостинг сервісу було обрано сервіс Openshift. Він дозволяє завантажувати веб-додатки на різні серверах-додатків (application server). В конкретному випадку було обрано в ролі серверу-додатків Tomcat 7. Це досить сучасний представник свого класу, який дозволяє використовувати всі сучасні інструменти для розробки, окрім можливостей мови програмування Java 8. Також даний сервіс надає базу даних MongoDb завантажену на окремому сервері в локальній мережі разом із веб-додатком. Це гарантує що база даних не буде доступна зовні і буде використовуватися тільки даним додатком. В загальному вигляді ми отримали триярусну архітектуру додатку (див. рис. 3.3).

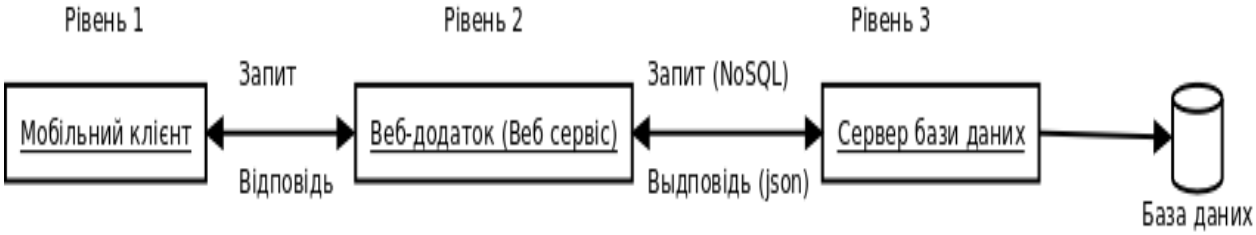

Рисунок 3.3 – Архітектура додатку

Структура класів була розділена також по різним рівнях, як це було зроблено в мобільному додатку (див. рис.3.4). Також по даній структурі можна виділити те що в нас є як dao так і сервіс через який беруться дані із дао. Це вже

говорить про те що в додатку реалізовано декілька рівнів (шарів) абстракції, тому можна також сказати що даний додаток використовує багатошарову архітектуру. Це надає змогу робити зміни в одному шарі, що не викличе додаткових проблем у других шарів, звісно якщо ми будемо оперувати тільки абстракціями.

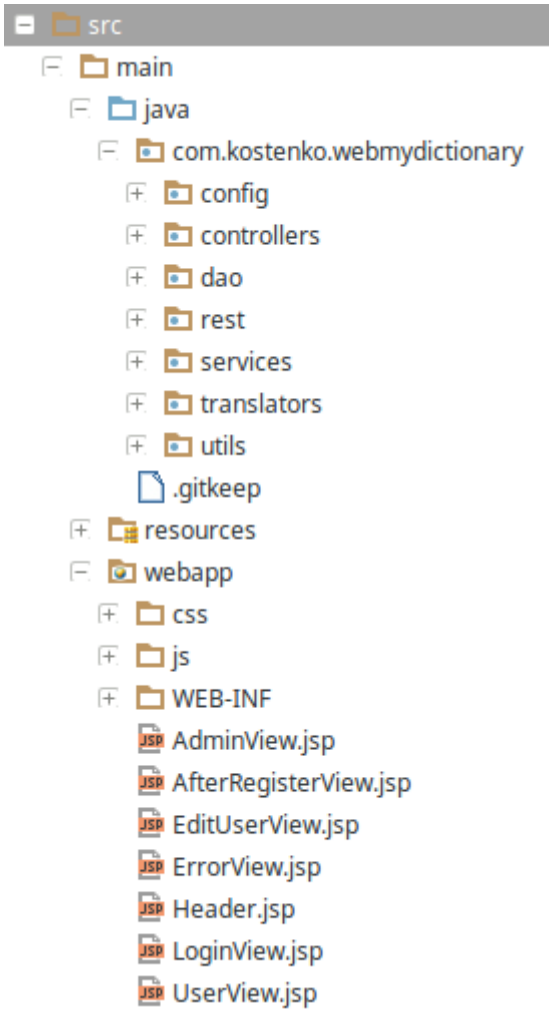

Рисунок 3.4 — Структура організації класів веб-додатку

Для зв’язку мобільного додатку з веб-додатком було обрано REST сервіс, а як тип відповіді сервера JSON документ. Це надає змогу спростити реалізацію як мобільного так і веб додатків, тому що до REST сервісу ми можемо робити звичайні Http запити як наприклад GET, POST, PUT і в відповідь отримувати JSON документ, який легко читається, а також і парситься.

Отже якщо зібрати все це в одне визначення, то даний додаток має триярусну, багатошарову архітектуру, з використання REST сервісу для обміну даними клієнтом.

3.3 Опис моделей

Моделювання – це метод дослідження явищ і процесів, що ґрунтується на заміні конкретного об’єкта досліджень іншим, подібним до нього (моделлю). Моделювання – це процес створення та дослідження моделі, а модель – засіб, форма наукового пізнання. Існує багато видів моделювання, але для розробки програмного забезпечення використовується математичне моделювання. Математичне моделювання – це моделювання, при якому модель являє собою систему співвідношень, що описують певні технологічні, економічні чи інші процеси [6].

UML – це уніфікована мова моделювання, яка використовується у парадигмі об’єктно-орієнтованого програмування та є невід’ємною частиною уніфікованого процесу розробки програмного забезпечення [6]. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, названої UML-моделлю. UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем.

UML може бути застосовано на всіх етапах життєвого циклу аналізу бізнес-систем і розробки прикладних програм. Різні види діаграм які підтримуються UML, дають можливість представити систему у такому вигляді з якого її можна легко перевести в програмний код об’єктно-орієнтованих мов програмування, що ще більш прискорює процес розробки. Саме тому для найбільш повного уявлення про програмну систему та її принципи дії, були побудовані наочні UML діаграми, котрі дають повний опис системи [6].

Для проектування програмної системи біло використано засоби проектування Microsoft Visio, Moqups.com, Creately.com та створено UML діаграми.

Під час моделювання системи була проведена побудова 10 діаграм, що цілком відображають роботу та функціонал майбутньої системи. Кожна діаграма відображує систему з конкретної сторони, і у сукупності це дає змогу отримати потрібну інформацію для розробки системи.

Діаграма варіантів використання. Для створення системи необхідно провести проектування поставлених задач, для цього проводиться моделювання системи та її окремих компонентів, з використанням UML діаграм [6]. У початковому аналізі було визначено основну структуру та поведінку системи взаємодію з нею з точки зору користувачів, зокрема зареєстрованих та не зареєстрованих користувачів. На рисунку 3.5 зображено діаграму використання, на якій представлено два актори: гість та авторизований користувач. Гість може зареєструватися або увійти до системи. Авторизований користувач може відредагувати інформацію профілю та змінити стан вакансії чи листівки.

У діаграмі варіантів використання виділено основний функціонал системи та вказано ролі користувачів, що будуть користуватись ним [7]. У системі наявні такі ролі користувачів: зареєстрований користувач та гість. Кожна з ролей має окремий набір функцій.

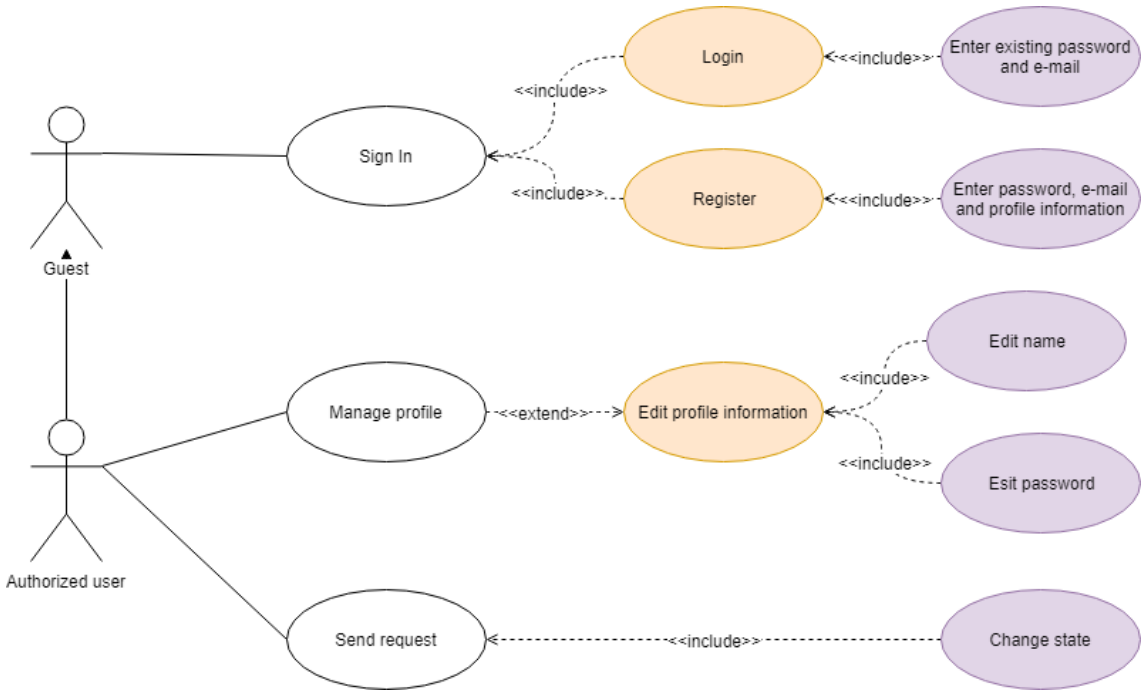

Рисунок 3.5 – Діаграма варіантів використання

Діаграма послідовностей представляє собою систему в цілому та те, як відбувається взаємодія між об’єктами на різних рівнях [8]. Послідовність дій протягом усієї роботи з системою, починаючи від реєстрації нового користувача і закінчуючи зміною стану вакансії зображено на діаграмі послідовності, що представлена на рисунку 3.6.

Також було розроблено діаграму послідовності, що відображає взаємодії об’єктів впорядкованих за часом. Зокрема, така діаграма відображає задіяні об’єкти та послідовність відправлених повідомлень між ними [8].

На ній можна побачити, що відбувається в системі поза очима користувача. Коли користувач відкриває вікно з даними, відображення надсилає запит до контролера для отримання даних, контролер в свою чергу надсилає запит до моделі, яка отримує дані з бази даних. Ці дані з бази надходять до відображення пройшовши обробку і відображаються користувачу. Коли до поштового серверу POP3 надходить нова листівка і приходить час автоматичного оновлення, дані

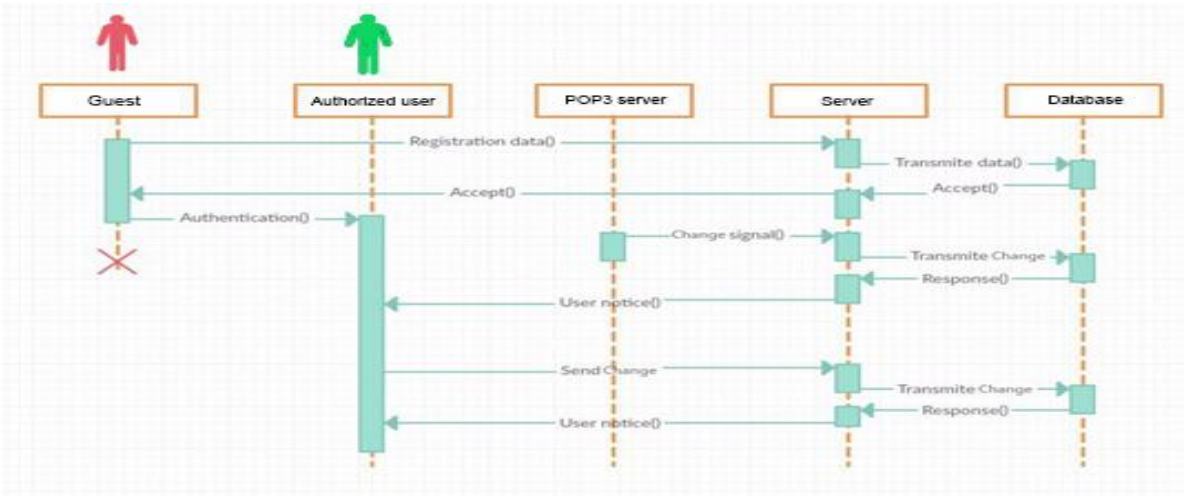

Рисунок 3.6 – Діаграма послідовності

через відображення відправляються до контролера, який їх валідує та відправляє до моделі. Модель спочатку обробляє дані, а потім зберігає їх до бази. Контролер в свою чергу повертає користувачу сторінку з результатом, або перенаправляє до початкової сторінки. Це загальна схема послідовності роботи будь-якого користувача з системою.

4 ОПИС РОЗРОБЛЕНОЇ ПРОГРАМНОЇ СИСТЕМИ

4.1 Проектування програмного додатку

В системі повинні бути реалізовані одна роль – користувача, адже кожен акаунт являє собою ізольовану систему, тож одна роль є достатньою умовою для функціонування системи. Користувачі мають можливість додавати вакансії, змінювати опис вакансій, контролювати їх стан, видаляти вакансії. Користувач має право додавати, редагувати, вмикати, вимикати та видаляти вакансії.

Для реалізації даної задачі необхідно реалізувати наступні функції:

– організувати систему реєстрації, аутентифікації і авторизації користувачів в системі;

– спроектувати і продумати зручний, інтуїтивно-зрозумілий користувачам інтерфейс;

– реалізувати функціонал класифікації електронної пошти за вакансіями за допомогою нейромережевого підходу;

– реалізувати функціонал перегляду класифікованої системою електронної пошти;

– реалізувати функціонал перегляду, додавання, вмикання/вимикання та видалення вакансій, перегляду та пошуку листівок для користувачів.

Окрім доступу до інформації необхідно також виконувати її модифікацію, і додавання нових даних, тому в інформаційній системі повинні бути реалізовані можливості редагування, видалення і додавання нових даних.

Вище наведені функціональні вимоги до системи для забезпечення керування вакансіями та листами підприємства.

Під час програмної реалізації функціональних вимог до програмного продукту потрібно врахувати такі обмеження:

– додати вакансію можна лише ввівши достатню для створення категорії класифікації інформацію;

– –одну електронну адресу можна прив’язати лише до одного аккаунту;

– один користувач може мати необмежену кількість вакансій;

– До нефункціональних вимог відносяться:

– в якості нейронної мережі повинна бути використана згорткова нейронна мережа із бібліотеки Tensorflow;

– в якості попереднього обробнику текстових даних повинна бути використана технологія Word2vec;

– серверна частина повинна бути реалізована за допомогою технології ASP.NET;

– в якості СУБД потрібно обрати Microsoft SQL Server;

– клієнт являє собою WEB-додаток, який розгорнутий на сервері IIS Express;

– написання коду за допомогою мов програмування C#, JavaScript та Typescript, бібліотек React, LINQ та Dapper, мов розмітки HTML та CSS;

– застосування повинно бути реалізоване з використанням цих фреймворків: ASP.NET Web APIFramework, ReduxFramework;

Під час реалізації нефункціональних вимог до програмного продукту потрібно врахувати такі обмеження:

– навчання нейронної мережі повинно бути швидким;

– серверна частина сайту повинна витримувати будь-які навантаження для забезпечення безперебійної роботи застосування;

– база даних не повинна містити обмежень щодо кількості можливих записів у одного користувача;

– унікальну ідентифікацію клієнтів необхідно здійснювати за допомогою перевірки e-mail адреси на унікальне значення.

Для класифікації текстових даних слід використовувати нейронну мережу з бібліотеки Tensorflow, яка навчається за допомогою даних електронних листівок оброблених за допомогою технології Word2vec.

Для зберігання даних слід використовувати базу даних. В якості СУБД слід обрати Microsoft SQL Server, що краще всього інтегрується з ASP.NET проектами, при використанні технологій Dapper та LINQ.

Усе це повинно підкріплюватись роботою технології React, як найбільш зручної та сучасної, що надає новий движок відображень для клієнт-серверних застосувань. Це спростить процес програмування (зробить його більш швидким). Компактний синтаксис зменшить кількість натискань клавіш, в той же час, покращуючи читабельність розмітки і коду. При обробці запитів фреймворк ASP.NETWeb API спирається на систему маршрутизації, яка зіставляє всі вхідні запити з визначеними в системі маршрутами, які вказують який контролер і метод повинен обробити даний запит. Вбудований маршрут за умовчанням передбачає триланкову структуру: controller, action, command/query.

4.2 Опис роботи програмної системи

Користувач вводить дані для реєстрації чи входу, клієнт їх обробляє та передає на сервер, який у свою чергу після обробки передає данні до бази даних, яка їх зберігає. У випадку зміни стану POP3-серверу сигнал передається на сервер, який обробляє подію, передає данні для збереження у базі даних та

надсилає повідомлення до користувача, що булаотримана нова листівка. Якщо користувач просто відкриває сторінку для перегляду, то виконується перша частина діаграми послідовності.

Тобто коли користувач просто відриває сторінку з даними, то контролер запрошує дані у моделі, яка отримує їх з бази та повертає, контролер обробляє дані та повертає дані для відображення.

Подальше редагування не відбувається і інша частина діаграми послідовності не відбувається. Після перевірки відповіді також відбувається занесення відповіді у базу даних, для подальшого використання.

Також було розроблено діаграму станів. Для моделювання поводження на логічному рівні в мові UML можуть використовуватися відразу кілька канонічних діаграм. На відміну від інших діаграм, діаграма станів описує процес зміни станів тільки одного класу, а точніше – одного екземпляра певного класу, тобто моделює всі можливі зміни в стані конкретного об’єкта. При цьому зміна стану об’єкта може бути викликана зовнішніми впливами з боку інших об’єктів або ззовні. Саме для опису реакції об’єкта на подібні зовнішні впливи й використовуються діаграми станів.

Головне призначення цієї діаграми – описати можливі послідовності станів і переходів, які в сукупності характеризують поводження елемента моделі протягом його життєвого циклу. Діаграма станів представляє динамічне поводження сутностей, на основі специфікації їхньої реакції на сприйняття деяких конкретних подій. Системи, які реагують на зовнішні дії від інших систем або від користувачів, іноді називають реактивними. Якщо такі дії ініціюються в довільні випадкові моменти часу, то це говорить про асинхронне поводження моделі.

На діаграмі зображений процес переходу станів сервера та користувача під час роботи програмної системи, що представлено на рисунку 4.1. Перехід між станами виконується в залежності від дій користувача. Треба зауважити, що на

діаграмі береться до уваги також можливість користувача в будь який момент припинити роботу з програмою.

Якщо користувач авторизується то система переходить у режим обслуговування користувача, потім, користувач може переводити систему у режим редагування профілю або відправки виклику. Ці процеси відбуваються доки користувач не вийде з системи.

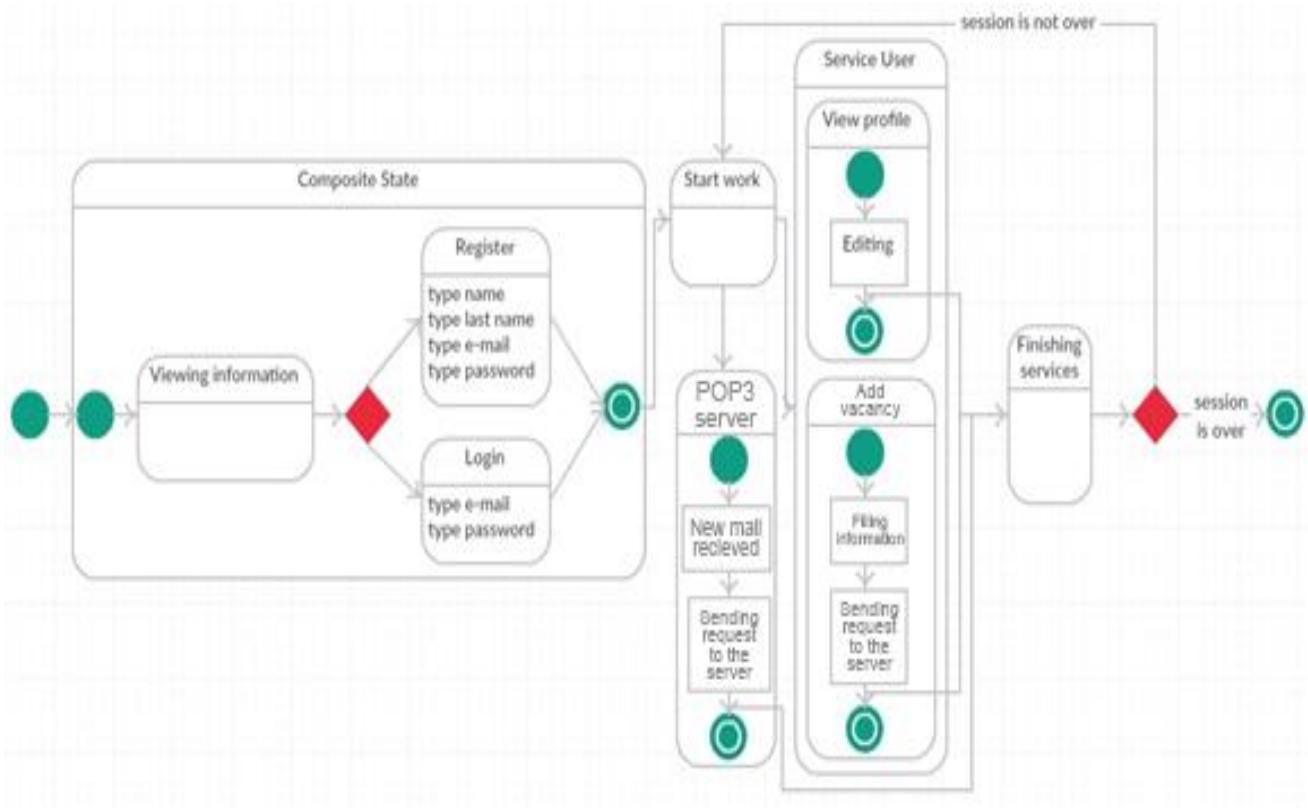

Рисунок 4.1 – Діаграма станів

Ще однією діаграмою, що було розроблено є діаграма активності системи [9]. На даній діаграмі відображено процес логіну користувача до системи, що представлено на рисунку 4.2.

В залежності від даних, які валідуються на сервері, клієнт залогіниться в систему, чи отримає помилку авторизації. Після авторизації клієнт може створити виклик чи редагувати дані профілю. Якщо валідація даних пройде не успішно клієнт повернеться до кроку авторизації.

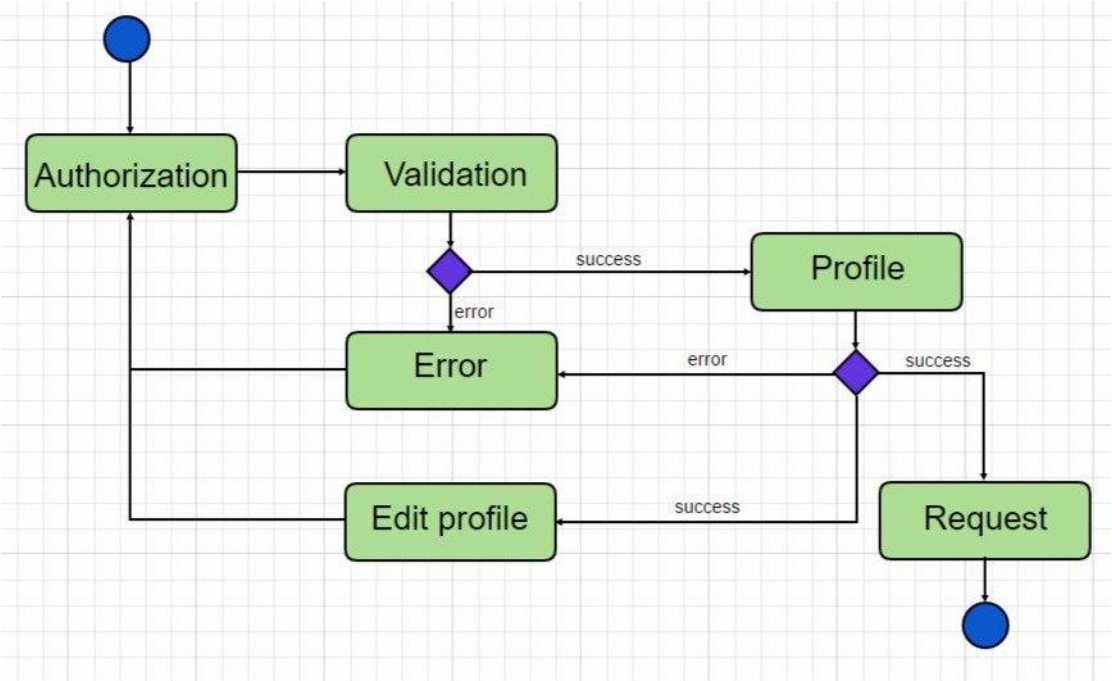

Рисунок 4.2 – Діаграма активності

Проект розроблений з використанням трирівневої архітектури, що базується на «Законі Деметри» (LawofDemeter, LoD), фундаментальною ідеєю якого є те, що об’єкт має мати якнайменше уявлення про структуру та властивості чого завгодно (включаючи власні підкомпоненти). Класична трирівнева система складається з наступних рівнів: рівень представлення, рівень бізнес-логіки та рівень доступу до даних.

Presentationlayer (рівень представлення) – це той рівень, з яким безпосередньо взаємодіє користувач. Цей рівень включає компоненти для користувача інтерфейсу, механізм отримання введення від користувача.

Businesslayer (рівень бізнес-логіки) – містить набір компонентів, які відповідають за обробку отриманих від рівня уявлень даних, реалізує всю необхідну логіку додатка, все обчислення, взаємодіє з базою даних і передає рівнем подання результат обробки.

Data Access layer (рівень доступу до даних) – зберігає моделі, що описують використовувані суті, також тут розміщуються специфічні класи для роботи з

різними технологіями доступу до даних. Тут також зберігаються репозиторії, через які рівень бізнес-логіки взаємодіє з базою даних.

При цьому треба зазначити, що крайні рівні не можуть взаємодіяти між собою, тобто рівень представлення не можуть безпосередньо звертатися до бази даних і навіть до рівня доступу до даних, а тільки через рівень бізнес-логіки. Графічне представлення архітектури можна побачити на рисунку 4.3.

Рисунок 4.3– Діаграма компонентів

Було розроблено діаграму компонентів (componentdiagram). Діаграма компонентів – статична структурна діаграма, показує розбиття програмної системи на структурні компоненти та зв’язки (залежності) між компонентами. У ролі фізичних компонентів можуть виступати файли, бібліотеки, модулі, виконувані файли, пакети тощо [10].

З діаграми видно, що в системі є компонент бази даних, серверна частина додатку та клієнтська частина веб-сервісу, що представлено діаграмою на рисунку 4.4.

Компонент «база даних» використовує Dapper, що потрібен для роботи репозиторіїв. Компонент «Сервер» використовує Controller для формування сторінок та обробки запитів від клієнта і обробки потрібних для відображення

даних, а також використовує бібліотеку Dapper. Веб-клієнт, в свою чергу, використовує бібліотеку React.

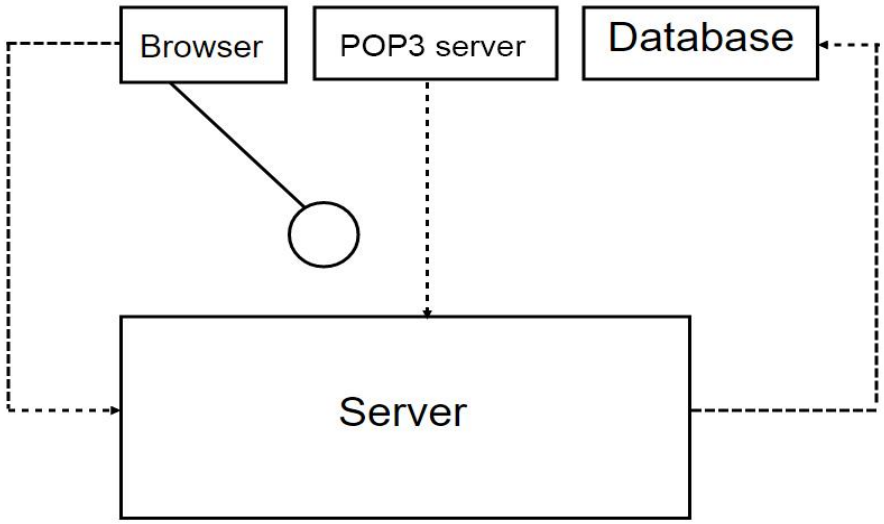

Рисунок 4.4– Діаграма компонентів

Треба відзначити, що на діаграмі зображені усі компоненти програмної системі та зв’язки між ними. Клієнтська частина додатку взаємодіє з сервером, а він в свою чергу взаємодіє з базою даних.

Також було розроблено діаграму розгортання, що була побудована відповідно до діаграми компонентів. На діаграмі розгортання зображено місцезнаходження окремих компонентів системи: на сервері додатку, сервері бази даних та поштовому сервері, що представлено на рисунку 4.5.

Було розроблено діаграму кооперації (CollaborationDiagram) – фундаментальне поняття в UML, що характеризує взаємодію об’єктів для реалізації деякої функції. Вона відображає те, як користувач впливає на систему та взаємодію об’єктів. Діаграма представлена на рисунку 4.6.

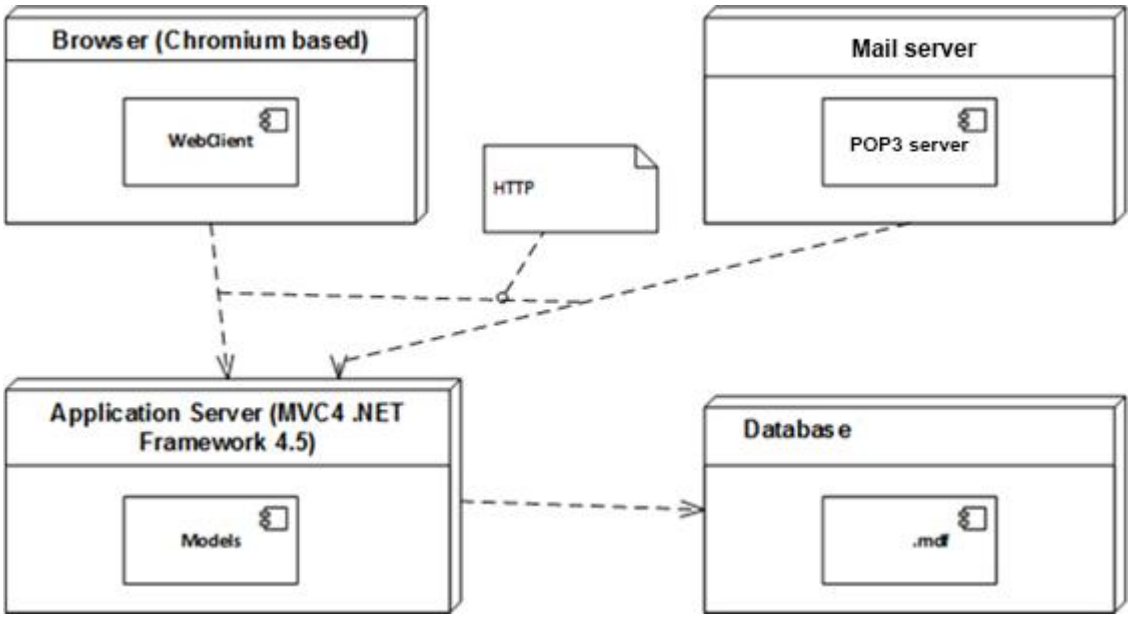

Рисунок 4.5 – Діаграма розгортання

На діаграмі відображено взаємодію основних елементів системи коли користувач змінює параметри вакансії. Користувач задає новий параметр (стан), данні заносяться до бази даних, у відповідності до унікального номера вакансії.

Рисунок 4.6 – Діаграма кооперацій

Архітектура даного програмного продукту розроблювалась у відповідності до структурованої методології розробки веб-сервісу класифікації текстових листівок за відкритими вакансіями за допомогою нейромережевого підходу.

З огляду на всі переваги і недоліки вже існуючих рішень необхідно використовувати всі переваги об’єктно-орієнтованого підходу в програмуванні для створення гнучкої архітектури програмного продукту. Це значно підвищить витрати на ранніх етапах розробки продукту, проте дозволить мінімізувати їх на стадії супроводу готового рішення. Досягнення гнучкості в архітектурі досягається побудовою концептуально правильною діаграми класів використовуючи основні парадигми ООП і SOLID принципи,що представлено на рисунку 4.7.

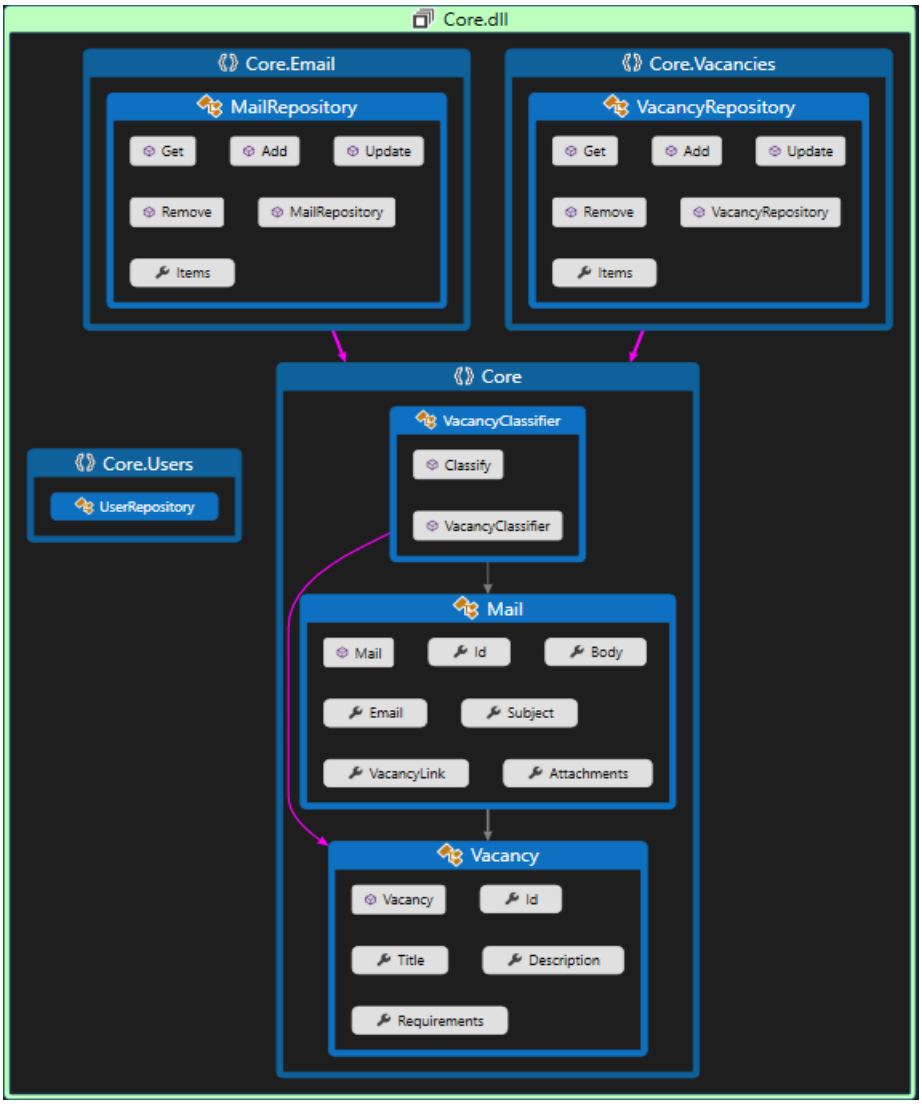

Рисунок 4.7 – Діаграма класів

Абстрактні класи та інтерфейси дозволяють додавати нові сутності шляхом виділення загальних методик і властивостей у вже існуючих (закладених) наборів поводжень, що представлено на рис. 4.8.

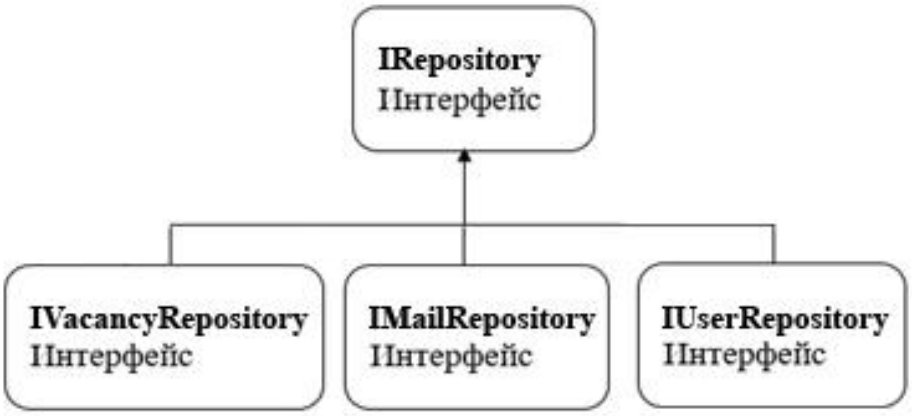

Рисунок 4.8 – Приклад реалізації інтерфейсу

4.3 Проектування бази даних

При проектуванні архітектури програмної системи, з метою підвищення безпеки розробленого програмного продукту, було прийнято рішення про розміщення бази даних поряд із бізнес шаром. У ній повинна міститись інформація про наявні вакансії та листівки.

Прийняття такого рішення, про розділення бази даних програмної системи, дозволить підвищити надійність та безпеку доступу до програмної системи (системи аутентифікації користувача).

Структуру основної бази даних представлено на рисунку 4.9. Фактично ця діаграма представляє собою і діаграму класів, так як використовувався підхід DatabaseFirst і моделі пов’язані з базою даних за допомогою репозиторіїв, тому робота з класами є робота з базою даних [11].

Розглянемо базу даних, таблиці якої відповідають наведеній структурі.

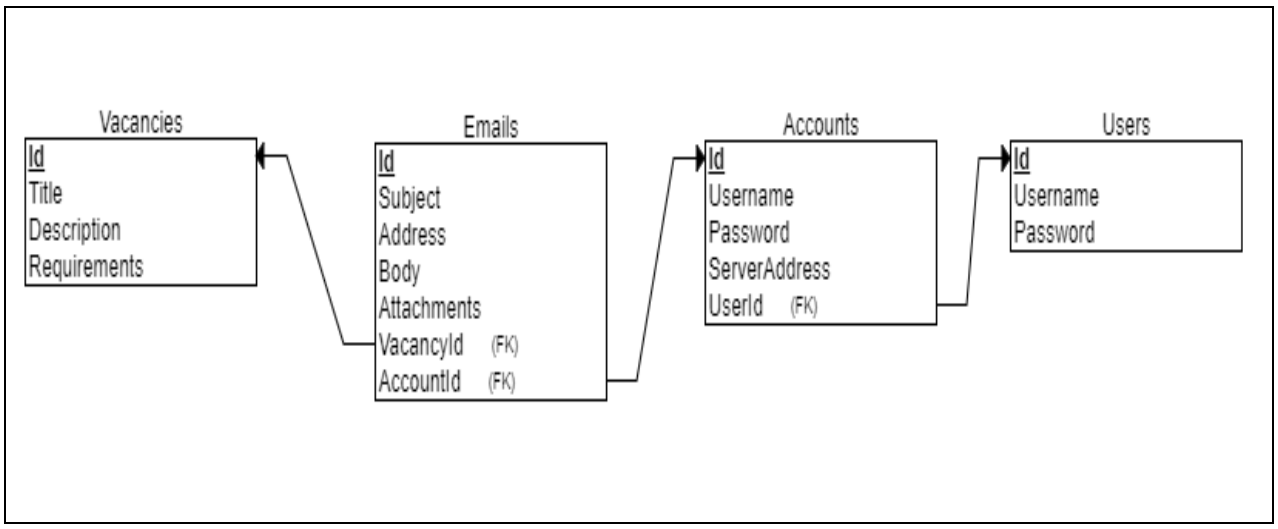

Рисунок 4.9 – Структура бази даних

Таблиця Vacancies буде зберігати дані про вакансії, зареєстровані у системі. Дані включають у себе унікальний номер, назву вакансії, опис, вимоги та її поточний стан (ввімкнений/вимкненій). Ці дані будуть використовуватися при відображенні користувачу стану вакансій.

Таблиця Emails буде зберігатинадійшовшілистівки. Дані включають у себе унікальний номер, тему листа, адресу, текст повідомлення, вкладення та номер вакансії до якої віднесено листівку. Ці дані будуть використовуватися при відображенні користувачу листівок.

Таблиця Users буде зберігати дані про користувачів системи. Дані включають у себе унікальний номер таоблікові дані. Ці дані будуть використовуватися при авторизації в системі та зв’язуванні аккаунтів.

Таблиця Accounts буде зберігати відключені поштові аккаунти. Дані включають у себе унікальний номер, облікові дані, дані про сервер пошти таномер користувача, що прив’язав аккаунт. Ці дані будуть використовуватися при перевірці пошти та віднесенні листівок до вакансій користувачів.

До бази також виконується безпосередньо підключення та запити під час проходження тестування [11]. Ця база даних відображає інформацію про вакансії та листівки із резюме для них.

4.4 Приклади найцікавіших алгоритмів і методів

Під час роботи над проектуванням веб-сервісу дотримувалися принципи SOLID.

Абревіатура SOLID була запропонована Робертом Мартіном, автором кількох книг, широко відомих в співтоваристві розробників. Ці принципи дозволяють будувати на базі ООП масштабовані і супроводжувані програмні продукти зі зрозумілою бізнес-логікою. Абревіатура розшифровується так:

- Singleresponsibility (принцип єдиної відповідальності);
- Open-closed (принцип відкритості / закритості);
- Liskovsubstitution (принцип підстановки Барбари Лісков);
- Interfacesegregation (принцип поділу інтерфейсу);
- Dependencyinversion (принцип інверсії залежностей).

Принцип єдиною обов’язки / відповідальності (singleresponsibilityprinciple) позначає, що кожен об’єкт повинен мати одну обов’язок і цей обов’язок повинна бути повністю інкапсульовані в клас. Всі його сервіси повинні бути спрямовані виключно на забезпечення цього обов’язку.

Принцип відкритості / закритості декларує, що програмні сутності (класи, модулі, функції і т.п.) Повинні бути відкриті для розширення, але закриті для зміни. Це означає, що ці сутності можуть міняти свою поведінку без зміни їх вихідного коду.

Принцип підстановки лісков (Liskovsubstitution) в формулюванні Роберта Мартіна: «функції, які використовують базовий тип, повинні мати можливість використовувати підтипи базового типу не знаючи про це».

Принцип поділу інтерфейсу (interfacesegregation) в формулюванні Роберта Мартіна: «клієнти не повинні залежати від методів, які вони не використовують». Принцип поділу інтерфейсів говорить про те, що занадто «товсті» інтерфейси необхідно розділяти на більш дрібні і специфічні, щоб клієнти маленьких інтерфейсів знали тільки про методи, які необхідні їм у роботі. У підсумку, при зміні методу інтерфейсу не повинні змінюватися клієнти, які цей метод не використовують.

Принцип інверсії залежностей (dependencyinversion) – модулі верхніх рівнів не повинні залежати від модулів нижніх рівнів, а обидва типи модулів повинні залежати від абстракцій; самі абстракції не повинні залежати від деталей, а ось деталі повинні залежати від абстракцій.

Для класифікації текстів було розроблено алгоритм класифікації текстових колекцій. У ньому використовується навчена на word2vec-векторах згорткова нейронна мережа Tensorflow. Word2vec – програмний інструмент аналізу семантики природних мов, що представляє собою технологію, яка заснована на дистрибутивної семантиці і векторному поданні слів. Цей інструмент був розроблений групою дослідників Google в 2013 році.

Робота цієї технології здійснюється наступним чином: word2vec приймає великий текстовий корпус в якості вхідних даних і зіставляє кожному слову вектор, видаючи координати слів на виході. Спочатку він створює словник, «навчаючись» на вхідних текстових даних, а потім обчислює векторне подання слів. Векторне подання ґрунтується на контекстній близькості: слова, що зустрічаються в тексті поруч з однаковими словами (а отже, мають схожий зміст), у векторному поданні матимуть близькі координати векторів-слів. Отримані вектори-слова можуть бути використані для обробки природної мови та машинного навчання.

Після обробки тексту Word2vec, отримані вектори класифікуються за допомогою нейронної мережі. Якщо нейронна мережа віднесла отримані дані до однієї з вакансій, то в базі даних Id вакансії додається до запису електронного листа в базі даних. На вхід нейронної мережі подається необхідна кількість векторів. Нейронна мережа має два прихованих шару. Перший відповідає за порівняння векторів, другий за визначення ваги – значення, що визначає ймовірність віднесення до певного кластеру. Вихідний шар віддає нам інформацію про результат класифікації. Алгоритм в цілому зображено на рисунку 4.10.

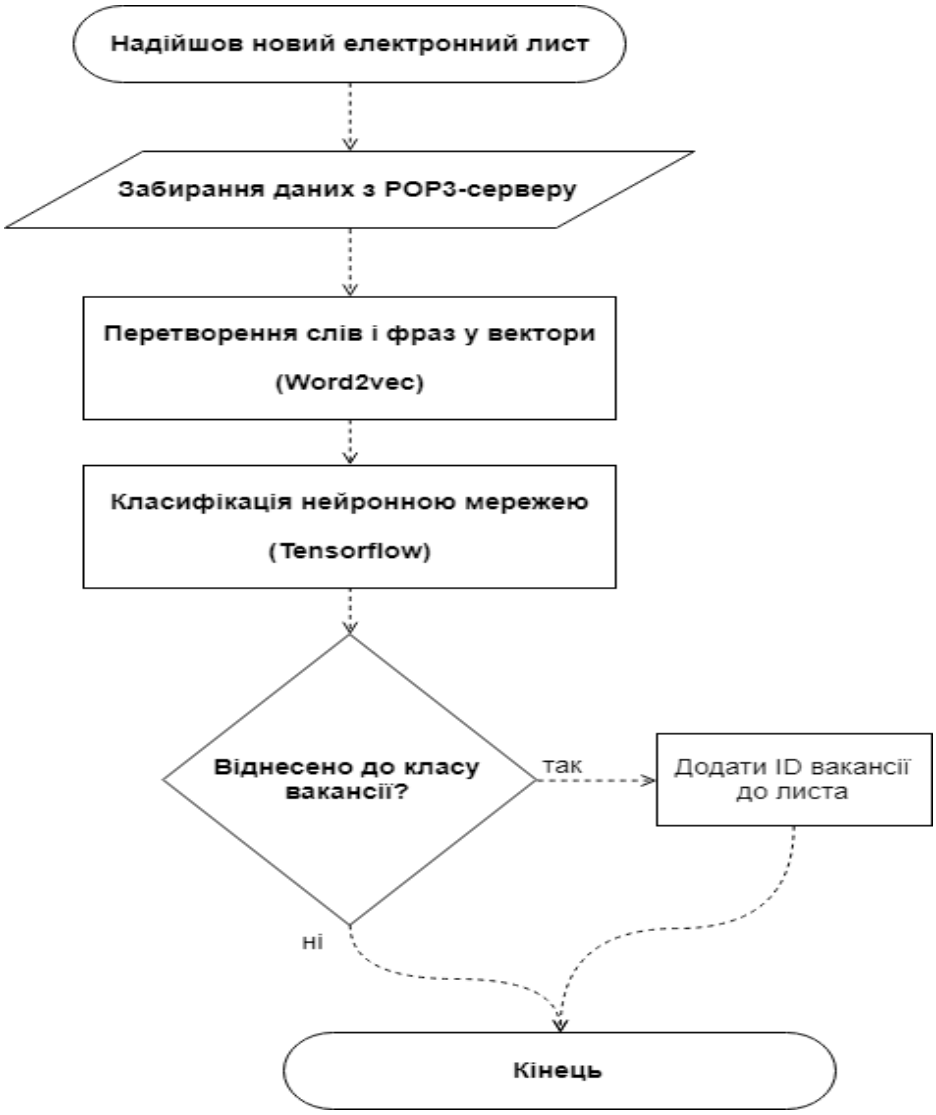

Рисунок 4.10 – Алгоритм класифікації текстових колекцій

В додаток до цього алгоритму, нейронна мережа проходить планове навчання кожен день, щоб навчитися на нових електронних листах.

4.5 Створення UI/UX

UX – це UserExperience (досвід користувача). Тобто це те, досвід (враження), який отримує користувач від роботи з інтерфейсом. Область відповідає за те, чи вдається йому досягти мети і на скільки просто або складно це зробити.

UI – це UserInterface (призначений для користувача інтерфейс). Це то, як виглядає інтерфейс і те, які фізичні характеристики набуває. Визначає, якого кольору буде «виріб», чи зручно буде людині потрапляти пальцем в кнопки, чи буде текст читабельним, тощо.

UX/UI дизайн – це проектування будь-яких призначених для користувача інтерфейсів, в яких зручність використання так само важливо як і зовнішній вигляд.

Перед початком проектування дизайну був зроблений аналіз сучасних тенденцій у сфері дизайну веб-сервісів, виділені популярні рішення та новаторські дизайнерські ходи. Огляд популярних сайтів за 2017 рік дозволив виділити основні підходи до проектування інтерфейсу користувача, що б відповідали таким параметрам як зручність у користування, так і привабливий графічний дизайн.

Тому, при проектуванні дизайну даної програмної системи було прийнято рішення дотримуватись наступних правил:

– світло падає згори;

– спочатку чорне і біле;

– збільшений білий простір;

– накладення тексту на картинки;

– використання сучасних популярних шрифтів.

Світло падає згори. Як відомо, наші екрани мають пласку поверхню, але ми докладаємо всіх зусиль, щоб будь-який елемент на ньому виглядав тривимірними, що представлено на рисунку 4.11.

Рисунок 4.11 – Застосування тривимірного ефекту до кнопки

Розглянувши даний приклад бачимо, що навіть в цій відносно «плоскій» кнопці є безліч деталей, пов’язаних зі світлом. У ненажатому стані (вгорі) у неї темний нижній край. Адже туди не падає сонячне світло. Верхня частина не натиснутої кнопки трохи світліша, ніж нижня. Це тому що вона імітує злегка вигнуту поверхню.

Для того щоб побачити сонячне світло, потрібно направити дзеркало вгору, так і в даному випадку – вигнута поверхня відображає трохи більше сонячного світла. У не натиснутому стані кнопка злегка відкидає тінь – її можна побачити в збільшеному вигляді. Натиснута кнопка зверху темніша, ніж знизу, тому що вона знаходиться на рівні екрану, і на неї потрапляє менше сонячного світла.

У реальному житті натиснуті кнопки теж темніші, тому що ми закриваємо світло рукою.

Це всього лише кнопка, але на ній вже є 4 невеликих ефекту від світла. Це і є основний принцип, що був пристосованих до потреб даного проекту.

Спочатку чорне і біле. Колір – це найскладніша частина графічного дизайну. Тому, перш за все, потрібно розробити усі елементі сайту, що мають білі та чорні кольори. Потім необхідно додати один яскравий колір, що суттєво відрізняється від чорно-білої гами інших елементів. Наприклад, це може бути помаранчевий, червоний, зелений колір. Він звертає увагу користувача саме на ті елементи, які є ключовими з точки зору маркетингу сайту.

Збільшений білий простір. Для того щоб інтерфейс виглядав привабливо та легко йому необхідно дозволити «дихати». Білий робочий простір сайту – загальноприйнята техніка. Вона дозволяє зменшити напругу на очі під час довготривалої роботи з сервісом та створює ілюзію розширення простору. Особливо це може бути корисним для користувачів, що працюють на пристроях з малими дисплеями.

Накладення тексту на картинки. У цьому правилі є багато проблем та ризиків. Перш за все фотографія повинна бути темна та не дуже контрастна. По-друге текст має бути білого кольору.

Це правило досягається декількома способами. Перший з них – умисне затемнення фотографії. Ще один простий та надійний спосіб – текст на фоні. Накресливши трохи прозорий чорний прямокутник потрібно розташувати текст білого кольору на ньому. Таким чином можна розташовувати будь-яку фотографію на фоні, не прибігаючи до штучного затемнення.

Ще одним популярним способом досягнення цього правила є розмиття фонового зображення. Не дивлячись на популярність такого методу він, як і перший спосіб, позбавляє користувача можливості побачити оригінальне зображення.

Виділення тексту як інструмент концентрації уваги. Секрет того, щоб текст виглядав одночасно красиво і доречно, полягає в застосуванні контрасту. До прикладу, можна зробити його більшим, але при цьому тонкими.

Використання сучасних популярних шрифтів. Сайти з дуже виразним «характером» можуть використовувати особливі шрифти. Але для більшості інтерфейсів потрібно щось чисте, просте і загальноприйняте. Найбільш популярними шрифтами останніх років є Roboto, OpenSans та HelveticaNeue.

5 ОПИС МОЖЛИВОСТІ ВИКОРИСТАННЯ ОТРИМАНИХ РЕЗУЛЬТАТІВ

5.1 Архітектура програмного продукту

Для реалізації веб-сервісу було вирішено використовувати клієнт-серверну архітектуру. Цей архітектурний підхід є одним із архітектурних шаблонів програмного забезпечення та є домінуючою концепцією у створенні розподілених мережевих додатків і передбачає взаємодію та обмін даними між ними [13]. Клієнт-серверна архітектура передбачає такі основні компоненти:

– набір серверів, які надають інформацію або інші послуги програмам, які звертаються до них;

– набір клієнтів, які використовують сервіси, що надаються серверами;

– мережа, яка забезпечує взаємодію між клієнтами та серверами.

У такій системі сервери є незалежними один від одного. Клієнти також функціонують паралельно і незалежно один від одного. Немає жорсткої прив’язки клієнтів до серверів. Більш ніж типовою є ситуація, коли один сервер одночасно обробляє запити від різних клієнтів, з другого боку, клієнт може звертатися то до одного сервера, то до іншого. Клієнти мають знати про доступні сервери, але можуть не мати жодного уявлення про існування інших клієнтів.

Для реалізації серверної частини було вирішено використовувати принципи проектування SOLID [14]. Усі ці принципи дуже важливі для побудови сучасного, надійного та масштабованого програмного продукту.

Для веб-клієнту, який реалізує веб-сервіс було обрано технологію React+Redux. Цю. Технологію було обрано, тому що вона наразі являється однією з найбільш функціональних та зручних для розробки веб-сервісів, актуальною і популярною серед розробників. Технологія Redux базується на шаблоні проектування Flux [15].

Підхід архітектури Flux поділяє програму на три основні частини: дії, сховища і уявлення. Розділення це здійснюється як на фізичному так і на логічному рівнях.

Використання Flux шаблону надає можливість будувати впорядковану програмну систему та дозволяє попередити та запобігти плутаниці в програмному коді, котра може виникнути в проектах значного масштабу [15]. Наглядна схема шаблону Flux представлена на рисунку 5.1.

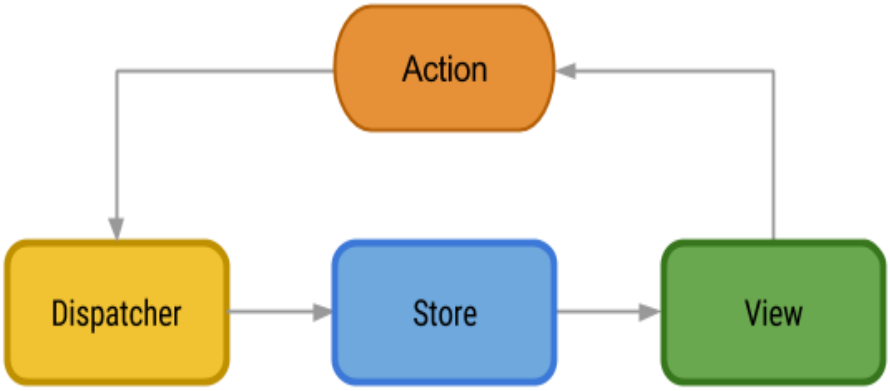

Рисунок 5.1 – Схема шаблону Flux

У моделі описуються основні сутності, використані в проекті. Контролер приймає і переадресовує запити користувачів завдяки своїм спеціальним методам, які обробляють запити і викликають певні відображення відповідних до запиту даних, отриманих з моделі. Представлення служить для відображення інтерфейсу сервісу.

Метою шаблону Flux є розробка гнучкого програмного дизайну, котрий під час подальшої підтримки програмної системи полегшує роботу та робить її швидшою і зрозумілішою. З'являється можливість повторного використання деяких компонентів програмної системи.

Flux дозволяє розробляти програми, різні аспекти яких(логіка вводу, бізнес-логіка та логіка інтерфейсу) розподілені, але досить тісно взаємодіють один з одним. Зв'язок між основними компонентами React+Redux також полегшує паралельну розробку. Розробник може підключати власний механізм уявлень,

політику маршрутизації URL-адресів, обробники параметрів методів дій й інші компоненти.

Redux також полегшує тестування проектів в порівнянні з веб-проектами на основі зі звичайним JavaScript та Jquery. Платформа React+Redux поділяє компоненти й активно використовує інтерфейси, що дозволяє тестувати окремі елементи поза решти структури.

React відзначає важливість дотримання чистої, відповідної стандартам розмітки. Вбудовані методи HTML помічника надають відповідні стандартам вихідні дані. Разом з платформою React можна використовувати додаткові функції з npm-репозиторіїв, наприклад спеціальні елементи інтерфейсу, вбудовані вирази, декларативні елементи управління, шаблони, прив’язку даних, локалізацію та багато інших аспектів.

Розглядаючи архітектуру системи більш детально слід приділити належну увагу рівню баз даних. Для взаємодії з БД користувачів було обрано технологію Dapper, яка дозволяє розробникам розробляти додатки для доступу до даних, що працюють швидко, адже вона використовує мінімальну кількість додаткових сутностей.

Для розробки БД за допомогою обрано стратегію DatabaseFirst («спочатку база даних»). Підхід DatabaseFirst дозволяє визначити модель з використанням запитів до бази даних, які будуть використані для створення репозиторіїв для роботи з базою даних [16].

Для зберігання інформації користувачів використовується база даних Microsoft SQL Server. SQL Server має вбудовану підтримку .NET Framework. Завдяки цьому, процедури бази даних, що зберігаються, можуть бути написані на будь-якій мові платформи .NET з використанням повного набору бібліотек, доступних для .NET Framework. На відміну від інших процесів, .NET Framework виділяє додаткову пам’ять і будує засоби керування SQL Server, не використовуючи вбудовані засоби Windows. Це підвищує продуктивність

порівняно із загальними алгоритмами Windows, оскільки алгоритми розподілу ресурсів спеціально налагоджені для використання у структурах SQL Server [17].

Оскільки проект є веб-сервісом, то він складається з серверної та клієнтської частини. Серверна частина виконує функцію обробки введених клієнтом даних та надання клієнту певної інформації відповідно до його запитів. База даних на сервері містить усю інформацію. Сервер надає інформацію з бази даних клієнту. Веб-клієнт, в свою чергу, виконає роль динамічної навігації між сторінками та взаємодії з клієнтом.

Для розробки серверної частини використовувалась технологія створення веб-сервісів ASP.NET Web API. Для роботи з базою даних, що будується на базі MS SQL Server, використовувалась технологія Dapper. Відповідно, мова програмування C# 6, яка вибрана через велику популярність і надійність .NET технологій та особисте бажання розробника. Для роботи з JSON (пакування запитів та розпакування відповідей) для використання Ajax, використовувалась технологія API Controller, що присутня в у збірці ASP.NET Web API [18].

Для розробки клієнтської частини використовувалися наступні мови та технології:

– HTML та CSS для розмітки та налаштування виду сторінок;

– JavaScript для динамічної роботи та інтерактивної взаємодії клієнта зі змістом сторінок;

– JavaScript бібліотека React, що полегшує створення уявлення;

– Bootstrap – набір CSS та JavaScript бібліотек, що надає способи для створення гнучких (Responsive) сайтів та містить набір іконок та елементів інтерфейсу.

Повертаючись до взаємодії сервера та клієнта, варто роз’яснити яку логіку роботи веб-сервісу було обрано. Ми маємо шар Контролера, який виконає роботу

по взаємодії шару Даних та шару Відображення, а також обробляє клієнтські запити сторінок.

Робота веб-сервісу виглядає таким чином: користувач звертається до сторінки, тоді метод контролеру, що відповідає за дану сторінку, починає свою роботу: збирає базові дані щодо сесії (авторизований користувач чи ні), дані для відображення. Після цього контролер повідомляє шару Подання, що дані готові для відображення й передає йому інформацію. Тепер в роботу включається шар Подання. Він, використовуючи передані шаром Контролера дані, формує шаблонну сторінку, завантажує необхідні для роботи клієнта React та CSS файли. Після цього в роботу включається клієнтський код. Він оброблює дані, що були передані сервером при генеруванні сторінки, готує та запускає клієнтську логіку та асинхронно завантажує (або бере з кеша даних) поточну сторінку. Після цього веб-сторінка готова до роботи з клієнтом. При переході між сторінками, клієнтський код у API Контролера необхідні дані та завантажує сторінки, отримуючи необхідні дані, чи відправляючи їх до Контролера [18].

Прикладом метода звичайного контролера може бути метод Login контролера StartController, що повертає шаблон сторінки. Метод контролера збирає всі необхідні для формування сторінки дані та повертає веб-сторінку. Після введення даних виконується метод контролера позначений [HttpPost], він оброблює введені дані та авторизує користувача або повертає помилку, якщо введені дані некоректні або користувача з такими даними не знайдено [19].

Більша частина роботи щодо вилучення інформації з бази даних, виконується в Моделі (шар Даних). Для кожної таблиці бази даних існує своя окрема модель, що відповідає за роботу з нею. Логіка роботи більшості моделей дуже очевидна, коли потрібно просто дістати дані з бази даних в чистому вигляді.

Логіка клієнтської частини створена на JavaScript. В клієнтському коді відбувається відгук на дії користувача. Тож JavaScript допомагає сконцентрувати деякі обчислення на клієнті, а також зробити веб-клієнт інтерактивним. Для

взаємодії з DOM веб-сторінки було обрано JavaScript бібліотеку jQuery. Вона значно полегшує й оптимізує роботу з DOM.

Для створення гнучкого та сучасного дизайну інтерфейсу була використана CSS та JavaScript бібліотека Bootstrap. З її допомогою була реалізована можливість однаково-добре переглядати сайт на різних розмірах екрану (Responsivedesign). В залежності від розмірів екрану, розміри та позиції елементів змінюються, щоб оптимально використовувати існуючу площу. Також, з цієї бібліотеки були використані іконки, кнопки та інші елементи інтерфейсу, на яких був побудований дизайн веб-сайту.

Отже, клієнтська частина реалізована так, щоб бути інтерактивною, відповідати потребам користувача, для чого були використані сучасні технології розробки.

Серверна частина виконує роботу щодо взаємодії в базою даних, обробки інформації для приведення її в необхідний вигляд, обробка інформації отриманої від клієнта, і, головне, спілкування з клієнтом шляхом запитів і відповідей. Клієнт, в свою чергу, виконує остаточне формування сторінки, спілкування з сервером через методи контролера та надає інтерфейсу користувача інтерактивності.

Інтерфейс користувача. Відкриваючи сайт, адміністратор бачить початкову сторінку з можливостями входу, після натискання кнопки авторизації він переходить на сторінку авторизації, яку зображено на рисунку 4.2. Ця сторінка передбачає введення електронної пошти та паролю для авторизації у системі адміністраторів та користувачів, користувачі авторизуються таким же чином. Після введення даних адміністратор або викладач натискає кнопку авторизації, а потім переходить на свою початкову сторінку, або отримує помилку авторизації при введенні некоректних даних.

5.2 Опис основних функцій веб-сервісу

Також веб-сервіс реалізує можливість авторизації користувачів за допомогою сторонніх сервісів (Google та Facebook). Для входу у веб-сервіс користувач повинен зареєструватися, ввести свої данні, додати пошту та створити пароль.

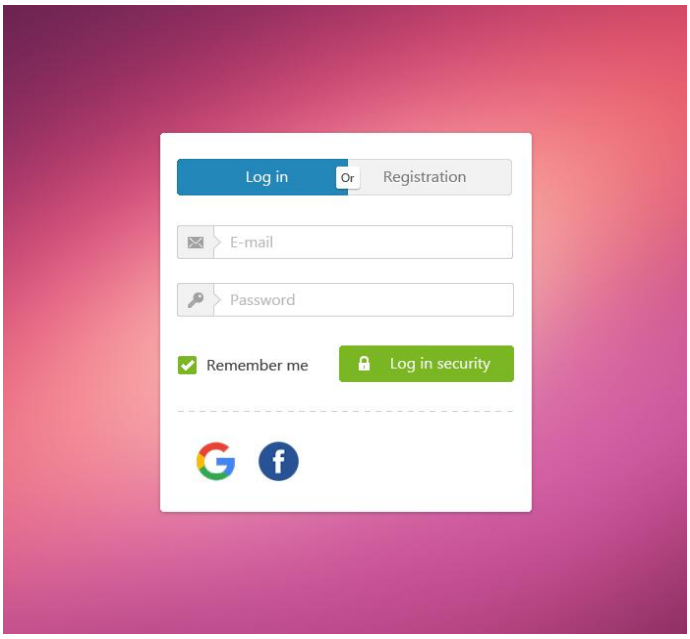

Рисунок 5.2 – Сторінка авторизації

Після авторизації користувач потрапляє на головну сторінку веб-сервісу, де проходить перегляд класифікованих електронних листів, на якій представлено основні можливості керування листами, надісланими до поштового серверу, які представлено на рисунку 5.3. Меню сторінки включає можливість перегляду та видалення листів, перегляд результату класифікації.

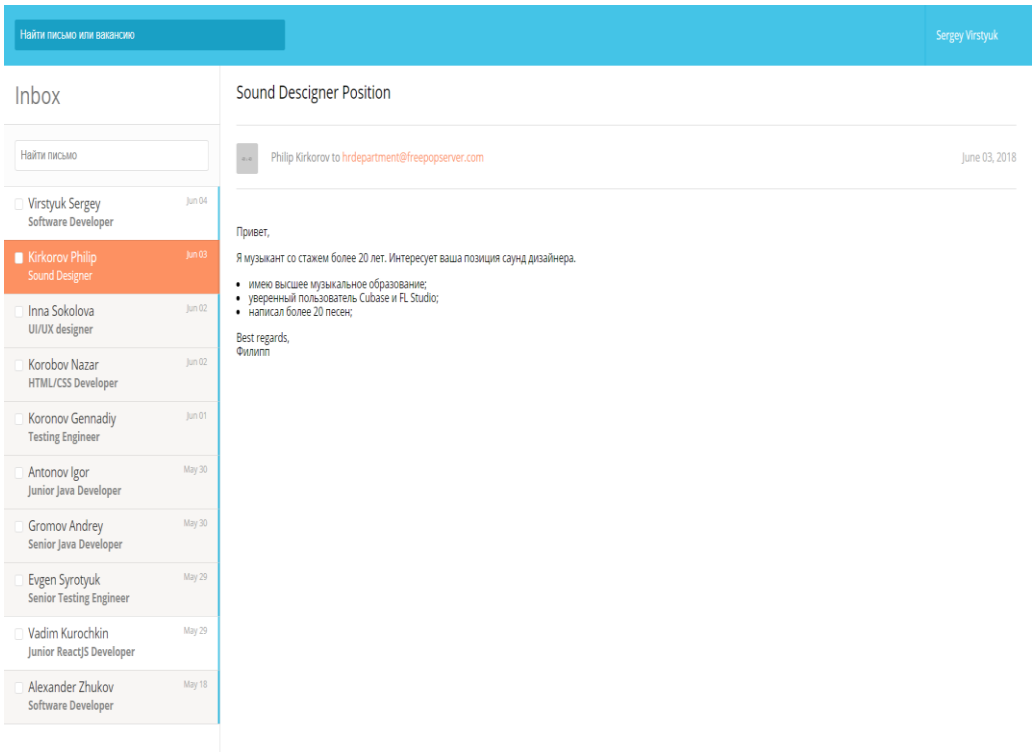

Рисунок 5.3 – Перегляд класифікованих електронних листів

Усі вакансії, що доступні наразі у системі, розміщені на окремій сторінці, що доступна з профілю користувача. Це спрощує орієнтування користувача у веб-сервісі, адже для кожної сутності використовується своє уявлення. На сторінці вакансій користувач може додавати, редагувати або видаляти інформацію відповідно обраної ним вакансії.

Також користувач може переглядати усі доступні вакансії для їх вмикання чи вимикання у системі класифікації. При створенні нової або редагуванні вже існуючої вакансії вона автоматично вмикається для класифікації.

Перегляд результатів роботи класифікації об’єднаний із сторінкою листівок, адже це найбільш ефективний підхід. Також користувачі можуть окремо переглянути сторінку із усіма наявними вакансіям, що представлено на рисунку 5.4.

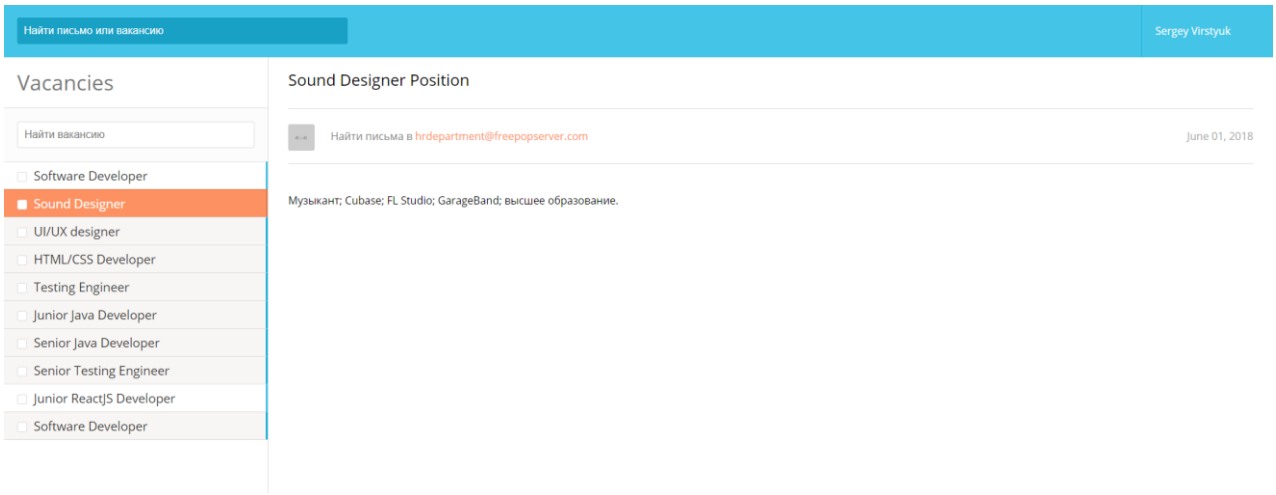

Рисунок 5.4 – Перегляд вакансій

Також є можливість підключити додаткові e-mailаккаунти, що представлено на рисунку 5.5.

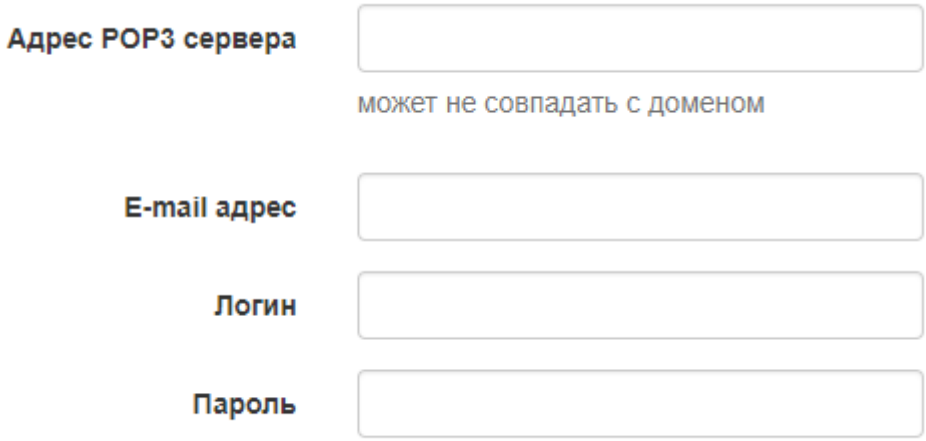

Рисунок 5.5 – Додавання нового поштового аккаунту

Отже, розглянуті інтерфейс користувача відповідає потребам до редагування даних. Можна змінити майже весь контент, що пов’язаний із профілем користувача. Оновлення даних проходить під час періодичних запитів до POP3-сервера.

ВИСНОВКИ

Під час написання атестаційної роботи був створений веб-сервіс для класифікації тексту за допомогою нейромережевого підходу. Проведений аналіз предметної області сприяв обранню кращих технологій для рішення поставленої задачі. Також, були спроектовані та розроблені функціональні та нефункціональні вимоги до системи.

Спроектовано та розроблено алгоритм класифікації текстових колекцій.

Під час проектування була розглянута діаграма варіантів використання, яка наглядно відображує функціональні можливості кожного типу користувачів. Також розроблена діаграма послідовності і активності, для кращого проектування послідовності процесів та дій. Основна база даних спроектована відповідно до вимог щодо збереження даних у системі, а саме користувачів та адміністратора.

Спроектовано та розроблено зручний інтерфейс користувача, що відкидає необхідність у додатковому навчанні користуванню системою. Також перевагою є те, що клієнтська частина веб-сервісу працює за технологією Flux та має гнучку архітектуру програмної системи, що дозволяє легше супроводжувати систему надалі, дороблювати інтерфейс і функціонал.

Сервер було розроблено з використанням мови програмування C#, середи розробки VisualStudio на програмній платформі .NET . Сервер розроблявся за технологією ASP.NET WEB API Framework. Готовий проект був розгорнутий для демонстрування локально в InternetInformationServices Express.

Одна з головних особливостей такої системи полягає у можливості достатньо точно класифікувати отримані повідомлення по завчасно заданим вакансіям із використанням алгоритму класифікації текстових колекцій.

Також програмний продукт передбачає можливість перегляду надісланих листів і їх категорій, та створених вакансій.

Розроблена система є корисним веб-сервісом для середнього та великого бізнесу зацікавленого у покращенні ефективності свого відділку кадрів, а також

для рекрутингових агенцій, що мають великий потік листівок на свої електронні адреси. Без сумнівів, цей сервіс не тільки економить час користувача, а ще й підвищує рівень якості роботи. При цьому були враховані всі вимоги, висунуті на початку виконання даного проекту.

Використання структурованої методології розробки систем для подорожування зробило процес розробки програмного продукту більш чітким, строгим та формалізованим. Завдяки цьому отриманий продукт повністю задовольняє поставленій задачі.

У ході роботи над проектом були дотримані усі правила написання «чистого коду» та рекомендації щодо проектування архітектури програмної системи. Також дотримані усі правила SOLID.

В подальшому розвиток системи можливий у декількох напрямках: додавання нового функціоналу, додавання підтримки додаткових СУБД, реалізація мобільних додатків на платформах Android та iOS.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Shukla, N. Tensorflow and Text Classification / N. Shukla– Indiana: WileyPublishing, 2010. – 212 p.

2. Gmail/ Gmail — Режим доступу: https://gmail.com– Загол. З екрану.

3. Outlook / Outlook– Режим доступу: https://outlook.com – Загол. З екрану.

4. TheBat! / TheBat– Режим доступу: https://www.thebat.com/– Загол. З екрану.

5. К. Вігерс. Розробка вимог до програмного забезпечення. – Російська редакція, 2004.

6. Діаграма прецендентів [Електронний ресурс] / Портал wikipedia.org − Режим доступу: https://uk.wikipedia.org/wiki/Діаграма_прецедентів – 10.11.2019 р. – Загол. З екрану.

7. Діаграма класів [Електронний ресурс] / Портал wikipedia.org − Режим доступу: https://uk.wikipedia.org/wiki/Діаграма_класів – 11.11.2019 р. –Загол. З екрану.

8. Діаграма активності / Портал wikipedia.org – Режим доступу: https://uk.wikipedia.org/wiki/Діаграма_активності – 18.11.2019 р. –Загол. З екрану.

9. Chapman N., Chapman J. Authenticationand AuthorizationontheWeb. 2012. – 246 с.

10. Діаграма кооперації / Портал wikipedia.org − Режим доступу: https://uk.wikipedia.org/wiki/Діаграма_кооперації – 10.11.2019 р. Загол. З екрану.

11. Виейра Р. Программирование баз данных Microsoft SQL Server 2005 для профессионалов / Р. Виейра – Диалектика, 2008. – 1072 с.

12. Fowler M., Scott K. UML distilled. 2d ed. 1999. – 282 с.

13. Smashing UX Design: FoundationsforDesigningOnlineUserExperiences 1st ed. 2010 – 456p.

14. Materialdesignguidelines. [Електронний ресурс] / Порталmaterial.io – Режим доступу: https://material.io/guidelines/ – 30.11.2019 р. – Загол. З екрану.

15. SOLID (объектно-ориентированное программирование) [Електронний ресурс] / Портал wikipedia.org − Режим доступу: https://uk.wikipedia.org/wiki/SOLID – 12.10.2019 р. –Загол. З екрану.

16. Freeman A. Pro ASP.NET MVC 5 5th ed. / A. Freeman – Apress, 2013. – 832 с.

17. CodeFirstMigrationsandDeploymentwiththeEntityFrameworkinan ASP.NET MVC Application/ The ASP.NET Site – Режим доступу: http://www.asp.net/mvc/overview/getting-started/getting-started-with-ef-using-mvc/migrations-and-deployment-with-the-entity-framework-in-an-asp-net-mvc-application – 10.11.2019 р. – Загол. З екрану.

18. JavaScript / Портал wikipedia.org − Режим доступу: https://ru.wikipedia.org/wiki/JavaScript – 12.11.2019 р. Загол. З екрану.

19. Jquery / Портал wikipedia.org − Режим доступу: https://ru.wikipedia.org/wiki/JQuery – 30.10.2019 р. Загол. З екрану.

20. CSS / Портал wikipedia.org − Режим доступу: https://ru.wikipedia.org/wiki/CSS – 16.11.2019 р. Загол. З екрану.

21. Модульное тестирование в приложениях ASP.NET / MDSN – Режим доступу: https://msdn.microsoft.com/ru-ru/library/gg416510(v=vs.98).aspx – 19.11.19 р.–Загол. З екрану.

22. Кириченко. І.В., Шубін І.Ю. Концепція пошукової системи агентного типу в системах електронного навчання/ І.В. Кириченко, І.Ю. Шубін // Збірник «Системи обробки інформації». – Х.: Харківський університет повітряних сил ім. Івана Кожедуба, Харків, 2018 № 2