Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

Навчально-науковий інститут деревообробних та комп'ютерних технологій і дизайну
(повне найменування інституту, назва факультету (відділення))

Кафедра інформаційних технологій
(повна назва кафедри (предметної, циклової комісії))

Пояснювальна записка

до дипломної роботи

другий (магістерський)
(рівень вищої освіти)

на тему: Розроблення веб-орієнтованої системи прогнозування розвитку вакансій ІТ-ринку в Україні.

Виконав: студент __6__ курсу групи КН-61м
спеціальності
122 "Комп'ютерні науки"
(шифр і назва напряму підготовки, спеціальності)

Дяків М. М.
(прізвище та ініціали)

Керівник Соколовський Я. І.
(прізвище та ініціали)

Рецензент Кособуцький П. С.
(прізвище та ініціали)

Львів – 2021

Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

ННІ деревообробних та комп'ютерних технологій і дизайну

Кафедра інформаційних технологій

Рівень вищої освіти другий (магістерський)

Спеціальність 122 "Комп'ютерні науки"
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Крошний І. М.
"____" _____2021 року

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Дяківу Мар'яну Мирославовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення веб-орієнтованої системи прогнозування розвитку вакансій ІТ-ринку в Україні.

керівник роботи *Соколовський Ярослав Іванович, доктор технічних наук*,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "31" грудня 2020 року № С-593

2. Термін подання студентом роботи

3. Вихідні дані до роботи Постановка задачі та її формалізація.

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
4.1 Стан проблемної області.
4.2 Інформаційне забезпечення.
4.3 Математичне забезпечення.
4.4 Розробка програмного забезпечення.
4.5 Висновок.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Знімки основних сторінок

6. Дата видачі завдання 18 грудня 2020р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1.	Огляд літературних джерел згідно із заданого завдання.	21.12.2021р.-11.01.2021р.	Виконано
2.	Аналіз об'єкту дослідження та стану проблемної області: 1. Аналіз досліджуваної проблеми. 2. Характеристика створюваної інформаційної системи. 3. Визначення вимог до системи.	12.01.2021р.-01.02.2021р.	Виконано
3.	Вибір методів і засобів створення веб-орієнтованої інформаційної системи, пошук математичних моделей прогнозування.	02.02.2021р.-24.05.2021р.	Виконано
4.	Програмна реалізація системи 1.Створення серверної частини. 2.Створення клієнтської частини.	25.05-2021р-20.09.2021р	Виконано
5.	Відлагодження роботи програми, заповнення бази даних.	21.09.2021р-22.11.2021р	Виконано
6.	Оформлення опису створеної програми	23.09.2021р-30.09.2021	Виконано
7.	Здача пояснювальної записки на перевірку та виправлення виявлених помилок	10.12.2021р	Виконано

Студент _____ **Дяків М.М.**
(підпис) (прізвище та ініціали)

Керівник роботи _____ **Соколовський Я.І.**
(підпис) (прізвище та ініціали)

Реферат

Дипломна робота містить 56 сторінок пояснювальної записки, 17 рисунків, 18 формул, 3 додатки, 9 джерел. 5 розділів, технічне завдання вступ, зміст, висновок.

Розглянуто математичні моделі прогнозування часових рядів.

Створено веб-орієнтовану систему прогнозування розвитку вакансій IT-ринку в Україні. Розроблено серверну частину, що збирає, аналізує та зберігає отримані дані, виконує прогнозування. Створено клієнтську частину із зручним інтерфейсом користувача. Досліджено розвиток ринку IT вакансій. Реалізовано алгоритм прогнозування для даного завдання.

Ключові слова: Java, Spring Boot, інформаційні технології, алгоритми прогнозування, AR, MA, ARIMA моделі.

Annotation

Master's thesis contains 56 pages of explanatory note, 17 figures, 18 formulas, 3 additions, 9 sources. 5 part, technical task, introduction, content, conclusion.

Mathematical modules of time series forecasting were considered.

A web-based system for forecasting the development of IT vacancies in Ukraine has been created. A server part has been developed that collects, analyzes and stores the received data, performs forecasting. A client part with a user-friendly interface has been created. The development of the IT vacancies is studied. The forecasting algorithm for this task is implemented.

Keywords: Java, Spring Boot, information technology, forecasting algorithms, AR, MA, ARIMA models.

ТЕХНІЧНЕ ЗАВДАННЯ.

Розробити програмне та алгоритмічне забезпечення для веб-орієнтованої системи прогнозування розвитку вакансій IT-ринку в Україні, а саме:

1. Реалізувати засоби збору інформації про вакансії із наявних джерел.
2. Розробити базу даних для зберігання інформації.
3. Розробити серверну частину програми призначену керування даними.
4. Серверна частина має надавати можливість фільтрування вакансій за різними критеріями, створювати статистичне представлення даних для їх відображення на клієнтській частині програми. Виконувати прогнозування за певними вибраними критеріями користувачем.
5. Вибрати алгоритм прогнозування та реалізувати його засобами мови програмування java.
6. Розробити клієнтську частину програми для взаємодії користувача та відображення даних отриманих із серверної частини.
7. Клієнтська частина має мати зрозумілий та простий у користуванні інтерфейс користувача.
8. Створити сторінку для зручного перегляду вакансій та із можливістю фільтрувати вакансії по певним критеріям.
9. Створити сторінки для відображення статистики вакансій на поточний момент часу та на певному відрізку. Відображення побудованого прогнозу.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ 8

ВСТУП 9

РОЗДІЛ 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ 10

1.1 Способи отримання інформації 10

1.2 Сервіси збору інформації 11

1.3 Прогнозування 12

1.4 Аналітичний огляд джерел 13

Висновок до розділу 14

РОЗДІЛ 2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ 15

2.1 Інтерфейс веб-орієнтованої системи 15

2.1 Мова програмування Java 17

2.3 Spring Framework 18

2.4 Реалізація процесу отримання інформації для системи 18

2.5 Фреймворк Angular 21

Висновок до розділу 22

РОЗДІЛ 3 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ 23

3.1 Основні методи аналізу часових рядів 23

3.1.1 Кореляційний аналіз 24

3.1.2 Регресійний аналіз 26

3.1.3 Модель “Ковзне середнє” 29

3.1.4 Моделі авторегресії і ковзного середнього ARIMA 31

Висновок до розділу 33

РОЗДІЛ 4. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 34

4.1 Створення баз даних 34

4.2 Створення серверної частини програми 36

4.3 Створення клієнтської частини програми 45

Висновок до розділу 52

РОЗДІЛ 5. РОЗРОБЛЕННЯ СТАРТАП-ПРОЄКТУ 53

5.1 Опис ідеї проєкту 53

5.2 Розроблення ринкової стратегії 54

5.3 Розроблення маркетингової програми 55

5.4 Вимоги до технічного та програмного забезпечення 55

Висновок до розділу 56

Висновок 57

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 58

Додаток А. Серверна частина 59

Додаток Б. Клієнтська частина 75

Додаток В. Діаграма класів програми 87

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

API - Application Programming Interface - Прикладний програмний інтерфейс.

IT – інформаційні технології.

DOM – Document Object Model – Об'єктна модель документа.

JSON – JavaScript Object Notation.

HTML – HyperText Markup Language - мова розмітки гіпертексту.

SQL – Structured query language — мова структурованих запитів.

AR - Autoregressive, авторегресійна модель.

ARIMA - autoregressive integrated moving average - авторегресійна інтегрована модель ковзнього середнього.

MA - moving average – модель ковзного середнього.

SMA - simple moving average – просте ковзне середнє.

WMA - weighted moving average – зважене ковзне середнє

SPA (Single Page Application) - односторінковий застосунок.

JVM (Java Virtual Machine). – віртуальна машина Java

ВСТУП

Розвиток IT ринку стає важливою складовою економіки в Україні. Щороку все більше людей починають працювати у цій галузі. Моделювання та прогнозування розвитку ринку вакансій дозволить людям зазирнути у майбутнє і краще зрозуміти котрі напрями та технології будуть затребувані.

На даний момент часу існує безліч сайтів із вакансіями для людей котрі хочуть почати працювати у IT сфері, проте вони не дозволяють бачити користувачам статистику зміни кількості вакансій по компаніям чи напрямкам. Також вони не будують прогнози зміни кількості вакансій на майбутнє.

Мета роботи – за допомогою засобів програмування створити веб-орієнтовану систему прогнозування розвитку ринку IT-вакансій Україні.

Об’єкт дослідження – процеси побудови та дослідження часових рядів.

Предмет дослідження – алгоритми прогнозування часових рядів.

Практичне значення полягає у тому, щоб зробити веб-орієнтовану систему котра буде об’єднювати дані про вакансії із декількох ресурсів та оброблятиме їх і приводитиме до спільної структури. Дозволятиме користувачу переглядати статистичні дані та прогнози розвитку ринку IT вакансій на майбутнє.

Новизна полягає у тому, що на даний момент не має подібних систем, котрі обробляють дані про вакансії із декількох джерел, їх зберігають, обробляють та зберігають статистику і будують прогноз.

Дана робота була представлена на всеукраїнській конференції "Комп'ютерне моделювання та інформаційні технології", що відбулася 19 жовтня 2021 року.

РОЗДІЛ 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

1.1 Методи отримання інформації

Збір певної інформації є рутинна та трудомістка робота, котра займає багато часу. Щоб отримати інформацію із сторонніх ресурсів можна використати парсинг або API.

Парсинг – це автоматичний процес збору неструктурованої інформації, обробка та перетворення її в структурований вигляд.

Парсер це програма, що призначена для обробки та перетворення вхідних даних в певний узагальнений формат.

Для збору інформації із сторонніх джерел використовують програми-парсери. Приклади використання: інтернет розсилки на пошту або мобільні номери перед тим отримавши електронні адреси і телефонні номера, аналіз цін на продукти у різних магазинах. Перевірка дипломних робіт на наявність плагіату містить додатково алгоритми аналізу документів на наявність схожої інформації у них.

Парсери не є нейронними мережами, що навчаються, тому вони лише порівнюють певний набір ключових фраз із тими, що вони знаходять на сайті.

При отриманні інтернет сторінки спочатку опрацьовується її DOM структура для отримання потрібних даних із сторінки. Парсер повинен вміти обходити DOM структуру сторінки та знаходити в ній потрібні секції або рядки інформації.

Значним недоліком у використанні парсерів є те, що вони не пристосовуються до зміни DOM структури сайту. Це призводить до того, що парсер потрібно переробляти. В наслідок цього розробники змушені змінювати алгоритми пошуку або створювати універсальний підхід для отримання даних та опрацювання і збереження у своєму продукті.

1.2 Сервіси збору інформації

Існує безліч універсальних сервісів для збору інформації із сайтів такі як:

Scrapinghub – це хмарний сервіс парсингу даних, котрий може вибирати і збирати потрібні дані. Scrapinghub використовує замінник проксі, обладнаний механізмами для обходу захисту від ботів та парсерів[7].

Marketparser – сервіс, що призначений для відслідковування цін у різних інтернет магазинах. Надає користувачу аналітику по зміні цін на заданий період часу та обрані продукти, що продаються у кількох інтернет магазинах.

Topvisor – платний сервіс що дозволяє відправляти користувачу дані із сайтів у вигляді CSV, PDF, HTML файлів, він також дозволяє відображати позиції відеороликів у платформі YouTube. Даний сервіс можна використовувати безкоштовно проте із лімітами на кількість отриманих даних.

Seoplane – надає 4 види отримання даних на вибір, глибина парсингу - 500 переходів у внутрішній структурі сторінок, зіставлення видимості ваших товарів із товарами конкурентів(для інтернет магазинів), програма надає звіти у форматі CSV, XLS, PDF, DOC. Сервіс надає гостьовий доступ у якому доступні 25 спроб запуску програми. Є можливість відстежити переходи відвідувачів. Надає фінансовий звіту.

PromoPult – цей парсер дозволяє отримувати метатеги та заголовки у інтернет сторінках.

Інструмент стане в нагоді при оптимізації сайтів. З його допомогою можна виявити:

- сторінки із порожніми метатегами;
- неінформативні заголовки чи заголовки з помилками;
- дублі метатегів.

Також парсер корисний під час аналізу SEO конкурентів. Ви можете проаналізувати під які ключові слова конкуренти оптимізують сторінки своїх сайтів, що прописують у title та description, як формують заголовки.

1.3 Прогнозування

Прогнозування — це метод, який використовує як і накопичений у минулому досвід, так і поточні здогадки щодо майбутнього для його визначення. Результатом правильного прогнозування буде картина майбутнього, яка може застосуватись для планування.

Головні різновиди прогнозів такі:

- економічні – передбачення загального стану економіки й обсягу продажу або виробництва конкретного продукту із можливістю врахування різноманітних економічних факторів.
- прогнози розвитку технології – передбачення доцільності використання та підтримки певних технологій.
- прогнози розвитку конкуренції.
- соціальне прогнозування – прогнозування змін стані суспільства.

Розрізняють два методи прогнозування: неформальні (прогноз будується на основі певного досвіду людини, яка будує прогноз), кількісні (прогноз будується на основі статистичних даних із наявною тенденцією змін).

У кількісному прогнозуванні вирізняють такі типові методи:

- Модель очікування споживача — це прогноз, що будується на результатах опитування клієнтів організації.
- Аналіз часових рядів - базується на дослідженні подій, які відбулися в минулому. Вони є основою для планування.
- Каузальне (причинно-наслідкове) моделювання – прогнозування того, що відбудеться у подібних ситуаціях. Досліджують статистичну залежності між фактором, що розглядається, й іншими змінними, що змінюються під дією цього фіктора. Ця залежність називається кореляцією. Чим кореляція рівномірна, тим більша надійність прогнозування даної моделі.

- Метод експертних оцінок — Прогноз будується на основі думок екпертів.

В процесі прогнозування досить часто використовується метод екстраполяції часових рядів. На основі вивчення динаміки зміни параметрів у попередніх періодах можна зробити висновки про значення показників у майбутньому. Обов'язковим елементом при цьому є побудова та аналіз ряду динаміки, який класифікує значення показників у часі та описує динаміку розвитку цих показників.

Теоретичною основою методів прогнозування є перш за все такі математичні дисципліни, як теорія імовірностей і математична статистика, математичне програмування.

1.4 Аналітичний огляд джерел

У книжці [2] у перших розділах описано основи роботи із даними. Описано основні можливості мови програмування Python для кращого ознайомлення із наукою про дані. В наступних розділах більш детально розглядаються способи візуалізації даних, робота із векторними і матричними даними. Подано основи математичної статистики і теорії імовірностей. Показано практичні способи отримання інформації із різних джерел за допомогою засобів мов програмування Java, Python та засобів API. Розкрито способи опрацювання лінійної, множинної, логічної регресій. Описано базові принципи роботи нейромереж та наведено приклади із навчанням мереж на основі природної мови.

Навчальний посібник [4] вміщає опис логіки статистичного моделювання, опис методу експертних оцінок та наводяться приклади. Описано правила для правильного формування інформаційної бази моделі, що буде досліджуватись. Наведено пояснення принципів прогнозування на основі часових рядів. Описано основні потрібні для цього формули та наведено приклади використання цих алгоритмів прогнозування на основі ковзніх середніх у економічній сфері з урахуванням сезонних змін.

У книжці [5] описано способи роботи із часовими рядами. Наведено приклади із аналізу рядів на основі AR та MA моделей. Наведено їх переваги і недоліки. Розкрито основи використання ARMA та ARIMA моделей і з якою метою моделі AR та MA використовуються разом. Продемонстровано приклади роботи із цими моделями у системі Matlab. Наведено практики побудови моделей ARCH та GARCH. Інструкції побудови цих моделей на основі часових рядів.

У книжці [1] наведено принципи роботи із фреймворком Spring. Його переваги над використанням Java EJB. Описано хороші практики роботи із шаблонами програмування для створення клієнт-серверних застосунків. Частина розділів присвячується для наведення прикладів роботи із різними складовими фреймворка, а саме: Hibernate, Core, WEB та ін.

Висновок до розділу

В даному розділі описано стан проблемної області, що включає в себе способи отримання інформації та сервіси які надають ці послуги, оскільки для наповнення бази даних даної системи використовуються дані із сторонніх джерел. Коротко описано способи та методи прогнозування та проаналізовано джерела, що містять опис математичних моделей прогнозування та опрацювання даних.

РОЗДІЛ 2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Інтерфейс веб-орієнтованої системи

Рис 2.1

Дана веб орієнтована система складається із двох частин: клієнтська та серверна частини.

Даний проєкт дозоляє користувачу виконувати пошук вакансій за різним критеріям. Також цей проєкт дозволяє оцінити користувачу популярність мов програмування на даний момент та тенденцію зміни кількості вакансій за певною мовою програмування. Отримати прогноз того чи іншого напряму в конкретному

місті або регіоні, чи по всій країні. Користувач може отримати список мов програмування, які є найбільш затребувані у компаніях, в яких є найбільша кількість відкритих вакансій. На рис 2.1 зображено варіанти дій користувача у системі.

Основними діями, котрі може здійснювати користувач, є перегляд вакансій, їх фільтрування за напрямом програмування, пошук вакансій у певному місті, перегляд детальної інформації про вакансії. Переглядати статистику по мовах програмування на даний момент часу, за попередні періоди часу та переглянути “Топ 10 компаній” за кількістю відкритих вакансій та які мови або напрями програмування є в них затребувані. Також статистику для зручності користувач може відфільтрувати за такими параметрами як місто, регіон або по усій країні.

Дана система працюватиме на основі клієнт-серверної архітектури Загальний принцип роботи зображсно на рис2.2

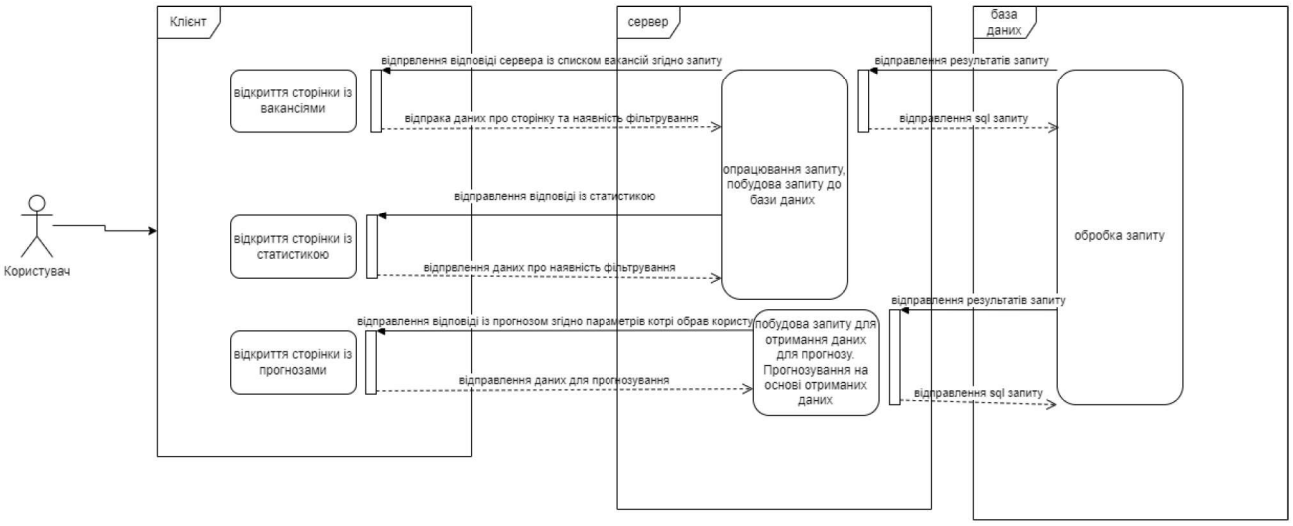

Рис 2.2

Як можна побачити, при діях користувача, клієнт відповідає за створення правильного запиту та відправлення його на сервер для отримання відповіді від сервера і відтворення відповіді на html сторінці. Сервер відповідає за правильність обробки запиту та отримання потрібних даних із бази даних і відправлення відповіді на клієнтську частину. Для створення серверної частини

було обрано мову програмування Java та фреймворк Spring. Для створення клієнтської частини було обрано мову програмування JavaScript та фреймворк Angular.

2.2 Мова програмування Java

На сьогоднішній момент мова Java є однією з найпоширеніших та найпопулярніших мов програмування.

Головною особливістю мови Java є те, що код спочатку перетворюється в спеціальний байт-код, незалежний від платформи. Далі цей байт-код виконується віртуальною машиною. Java відрізняється від основних інтерпретованих мов як Perl або PHP, код яких запускається інтерпретатором. Проте Java не є й чисто компільованою мовою, як C чи C++[7].

```
0000000 ca fe ba be 00 00 00 34 00 1d 0a 00 06 00 0f 09
0000010 00 10 00 11 08 00 12 0a 00 13 00 14 07 00 15 07
0000020 00 16 01 00 06 3c 69 6e 69 74 3e 01 00 03 28 29
0000030 56 01 00 04 43 6f 64 65 01 00 0f 4c 69 6e 65 4e
0000040 75 6d 62 65 72 54 61 62 6c 65 01 00 04 6d 61 69
0000050 6e 01 00 16 28 5b 4c 6a 61 76 61 2f 6c 61 6e 67
0000060 2f 53 74 72 69 6e 67 3b 29 56 01 00 0a 53 6f 75
0000070 72 63 65 46 69 6c 65 01 00 08 41 70 70 2e 6a 61
0000080 76 61 0c 00 07 00 08 07 00 17 0c 00 18 00 19 01
0000090 00 0c 48 65 6c 6c 6f 20 77 6f 72 6c 64 21 07 00
00000a0 1a 0c 00 1b 00 1c 01 00 09 68 65 6c 6c 6f 2f 41
00000b0 70 70 01 00 10 6a 61 76 61 2f 6c 61 6e 67 2f 4f
00000c0 62 6a 65 63 74 01 00 10 6a 61 76 61 2f 6c 61 6e
00000d0 67 2f 53 79 73 74 65 6d 01 00 03 6f 75 74 01 00
00000e0 15 4c 6a 61 76 61 2f 69 6f 2f 50 72 69 6e 74 53
00000f0 74 72 65 61 6d 3b 01 00 13 6a 61 76 61 2f 69 6f
0000100 2f 50 72 69 6e 74 53 74 72 65 61 6d 01 00 07 70
0000110 72 69 6e 74 6c 6e 01 00 15 28 4c 6a 61 76 61 2f
0000120 6c 61 6e 67 2f 53 74 72 69 6e 67 3b 29 56 00 21
0000130 00 05 00 06 00 00 00 00 00 02 00 01 00 07 00 08
0000140 00 01 00 09 00 00 00 1d 00 01 00 01 00 00 00 05
0000150 2a b7 00 01 b1 00 00 00 01 00 0a 00 00 00 06 00
0000160 01 00 00 00 03 00 09 00 0b 00 0c 00 01 00 09 00
0000170 00 00 25 00 02 00 01 00 00 00 09 b2 00 02 12 03
0000180 b6 00 04 b1 00 00 00 01 00 0a 00 00 00 0a 00 02
0000190 00 00 00 06 00 08 00 07 00 01 00 0d 00 00 00 02
00001a0 00 0e
00001a2
```

Рис 2.3 Байт код програми виводу напису ‘Hello world’ у консоль.

Подібна архітектура забезпечує кросплатформенність та апаратну переносимість програм на Java. Подібні програми можуть виконуватись на різних платформах без перекомпіляції.

2.3 Spring Framework

Spring Framework— це програмний каркас для мови програмування Java що підтримус іпверсію управліпня із відкритим програмпим кодом.

Spring Framework складається з багатьох модулів. Основні із них зображено на рисунку 2.4 [1].

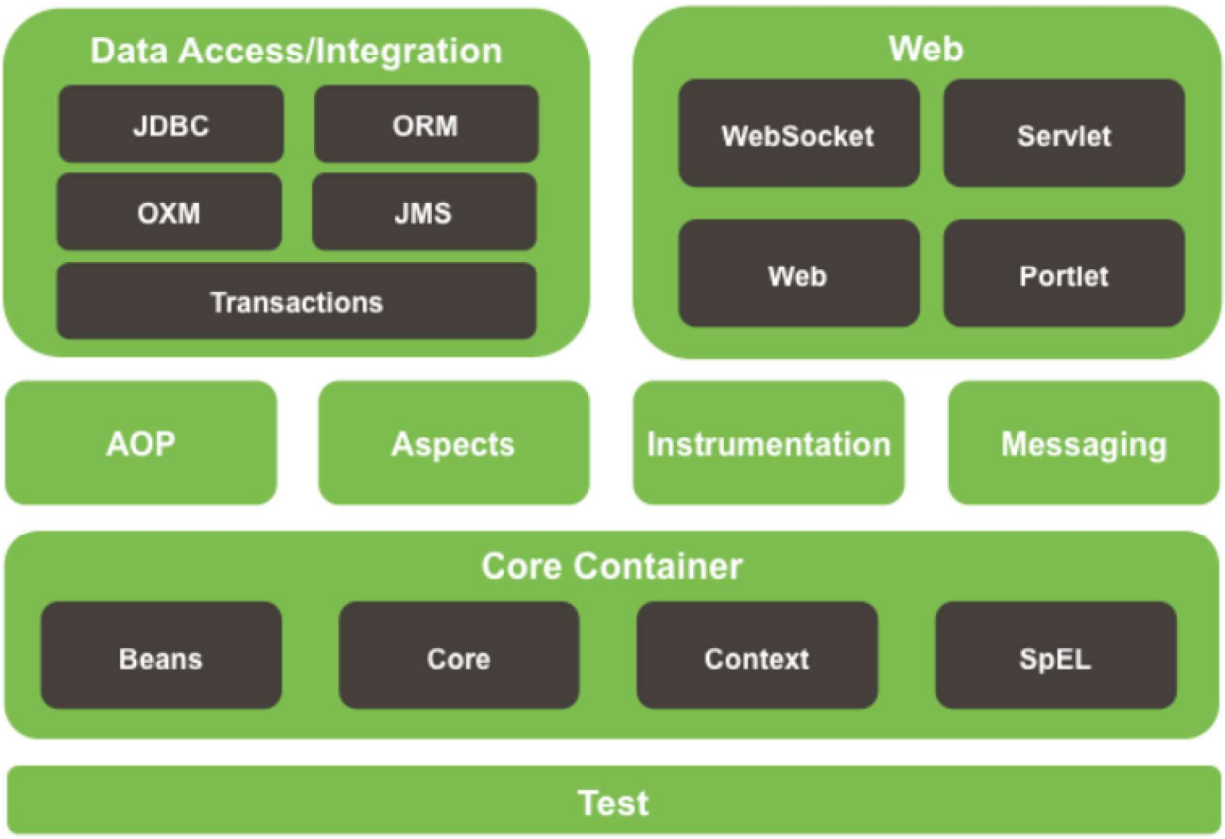

Рис 2.4 Модулі Spring Framework

Spring MVC – частина фреймворку для роботи із веб застосунками заснований на HTTP і сервлетах.

Spring Core Container – надає можливості конфігурації компонентів і керування життєвим циклом об'єктів.

Spring Security дозволяє побудувати та використання систем авторизації та аутентифікації.

Spring Boot – дозволяє легко створити автономні проєкти. Такі проєкти містять в собі вбудований сервер на якому можна запустити проєкт.

Spring Data Access -частина фреймворку, що містить механізми для роботи із різноманітними базами даних.

Spring Batch – пакетний фреймворк. Він забезпечує розробку надійних пакетних додатків. Дозволяє багаторазове використання додатків, що є важливими у процесі опрацювання великих обсягів записів [1].

2.4 Реалізація процесу отримання інформації для системи

Для можливості отримання інформації із сторонніх ресурсів обрано бібліотеку JSOUP.

JSOUP - це бібліотека Java , що дозволяє працювати із HTML сторінками інших сайтів. Jsoup аналізує HTML на рівні DOM, завдяки реалізації специфікації WHATWG HTML5.

Основні класи бібліотеки для роботи із DOM структурою[8]:

- org.jsoup.Jsoup – містить методи для з'єднання із веб сторінкою
- Document – призначений для отримання веб сторінки.
- Element – дозволяє взаємодіяти із окремими елементами сторінки, отримувати дані із частин сторінки у вигляді масиву об'єктів класу Element, або символьних значень.

Приклади основних методів класу Jsoup

static Connection connect (String url) – створює і повертає об'єкт класу Connection, що підключається до заданої адреси веб сторінки.

static String clean (String bodyHtml, Whitelist whitelist) – повертає безпечну сторінку HTML. Фільтрує вміст через білий (безпечний) список розширених тегів і атрибутів.

static Document parse (String html) – аналізує HTML код сторінки і повертає об'єкт класу Document.

Основні методи класу Document [8]:

Element head() – надає доступ в елемент head документа.

Document normalise() – нормалізує документ.

Element body() – надає доступ в елемент body документа.

Element createElement(String tagName) – дозволяє створити елемент за переданим у параметр ім’ям тегу.

String title() – повертає вміст заголовка документа.

static Document createShell(String baseUri) – створює нову пусту HTML сторінку для додавання елементів.

Основні методи класу Element [8]:

Elements getAllElements() – повертає клас Elements, котрий містить усі вкладені елементи у об’єкті, що викликав даний метод.

Elements getElementsByAttribute(String key) – поверне клас Elements, який містить елементи за вказаним значенням атрибута.

Element getElementById(String id) – повертає елемент за вказаним ідентифікатором.

Elements getElementsByClass(String className) – повертає об’єкт класу Elements, що містить елементи за переданим у параметр CSS класом на сторінці.

String text() – повертає стрічку із текстовим вмістом елементу.

Elements getElementsByTag(String tagName) – повертає об’єкт класу Elements, що містить елементи із вказаним тегом.

```
Document doc = Jsoup.connect("https://en.wikipedia.org/").get();
log(doc.title());
Elements newsHeadlines = doc.select("#mp-itn b a");
for (Element headline : newsHeadlines) {
  log("%s\n\t%s",
    headline.attr("title"), headline.absUrl("href"));
}
```

Рис 2.5 Зображення прикладу парсингу сторінки сайту Wikipedia.

2.5 Фреймворк Angular

Angular представляє собою фреймворк від компанії Google для створення SPA клієнтських програм. Він орієнтований на розробку односторінкових застосунків. Angular є спадкоємцем фреймворку AngularJS. Проте, Angular це не нова версія AngularJS, а цілковито інший фреймворк.

Angular надає такі можливості, як двостороннє зв'язування, що дозволяє динамічно змінювати дані в певному місці інтерфейсу користувача при зміні даних у моделі, шаблони, маршрутизація і так далі.

Однією з головних особливостей фреймворку Angular є те, що він використовує як основну мову програмування TypeScript. Також можна створювати програми на Angular за допомогою інших мов, таких, як: JavaScript або Dart.

Для компіляції програми на Angular використовується спеціально створений засів як Angular CLI. Angular CLI спрощує компіляцію і побудову каркасу проєкту. Angular CLI поширюється як пакет npm, тому щоб використати його потрібно встановити виконавши команду “npm install -g @angular/cli” у командному рядку[9].

Остання версія Angular – Angular 13 вийшла у листопаді 2021.

Angular дозволяє створювати складні і великі додатки. Фреймворк став більш гнучкішим у порівнянні із своїм попередником та більше використовується у enterprise-застосунках серез свою високу масштабованість.

Зміни в порівнянні із AngularJS [9]:

- Додано рішення Angular CLI, котре дозволяє створювати нові проєкти, і не тільки, використовуючи команду ng new [app name].
- Заміна контролерів на компоненти та директиви
- Використання концепції “ієрархія компонентів” замість концепції “області видимості”.
- Значна частина основного функціоналу перенесена у модулі, що дозволяє завантажувати і підключати лише потрібні модулі та дозволяє зменшити об’єм пам’яті що буде зайнята бібліотеками.
- Асинхронна компіляція шаблонів.
- Використовує мову Type Script.

Висновок до розділу

В даному розділі описано інтерфейс системи і її основні принципи роботи, можливості користувача та взаємозв’язки між клієнтською і серверною частино. Вказано матеріали які використовувались для створення системи такі, як: java, JSOUP, Angular. Наведено їх переваги та недоліки, та особливості використання.

РОЗДІЛ 3 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

Часовий ряд – це послідовність значень певного статистичного показника, впорядковану у хронологічному порядку. Використовують також терміни "ряд динаміки" або "динамічний ряд". Окремі спостереження часового ряду називають рівнем або елементом часового ряду. Кожний елемент ряду відповідає лише одному певному моменту часу. Рівні ряду можуть набувати як випадкових значень так і змінюватись за певною закономірністю. Порядок розташування рівнів не може змінюватися оскільки являється важливою характеристикою ряду.

Головними завданнями дослідження часових рядів є

- виокремлення та опис основних характерних ряду; вибір статистичної моделі, що найкращим чином характеризує часовий ряд.
- виокремлення закономірних складових ряду (тренду, сезонних та циклічних закономірностей).
- прогнозування майбутніх значень показників ряду за попередніми спостереженнями.
- виокремлення низько- та високочастотних значень.
- Дослідження випадкової складової часового ряду.
- дослідження отриманої математичної моделі і прогнозування майбутньої поведінки об’єкта.

3.1 Основні методи аналізу часових рядів

Часові ряди досліджуються з метою досягнення різних цілей. В одному випадку буває вдосталь отримати опис основних особливостей ряду, а в іншому часовому випадку потрібно не тільки передбачати можливі майбутні зміни значення часового ряду, а й керувати його поведінкою. Метод для проведення аналізу часового ряду вибирають з урахуванням цілей та природи формування його значень.

3.1.1 Кореляційний аналіз

Дослідження залежності між випадковими величинами називають кореляційним аналізом. Найпростіший випадок, коли потрібно дослідити дві вибірки, а в загальному випадку досліджують їх багатомірні комплекси.

Метою кореляційного аналізу — є виявлення існування істотної ознаки залежності одних змінних від інших.

Головні завдання кореляційного аналізу:

- перевірка значущості вибіркових коефіцієнтів або відношення.
- побудова довірчого інтервалу для коефіцієнтів кореляції.
- оцінка за вибірковими даними коефіцієнтів кореляції.
- оцінка близькості виявленого зв'язку до лінійного.

Щоб дослідити характер взаємозв’язку потрібно побудувати графічне зображення в системі координат результатів вимірів. Кожна пара значень залежної і незалежної величини буде відображатись однією точкою на площині. Побудоване зображення називатиметься діаграмою розсіювання або кореляційним полем

Діаграма розсіювання відображає статистичний зв’язок між результатами вимірів. Для оцінки форми, тісноти зв’язку і спрямованості використовують візуальний аналіз діаграми.

Форму зв’язку можна визначити по виду кореляційного поля: якщо через діаграму розсіювання можливо намалювати пряму лінію, яка не перетинає контури утвореної фігури, тоді форма зв'язку буде лінійною, якщо ні – нелінійною.

Із залежності між результатами вимірів визначають спрямованість взаємозв’язку. Якщо покращення одного показника спричиняє покращення другого – це пряма залежність. Якщо покращення першого показника спричиняє погіршення другого, тоді обернена залежність[6].

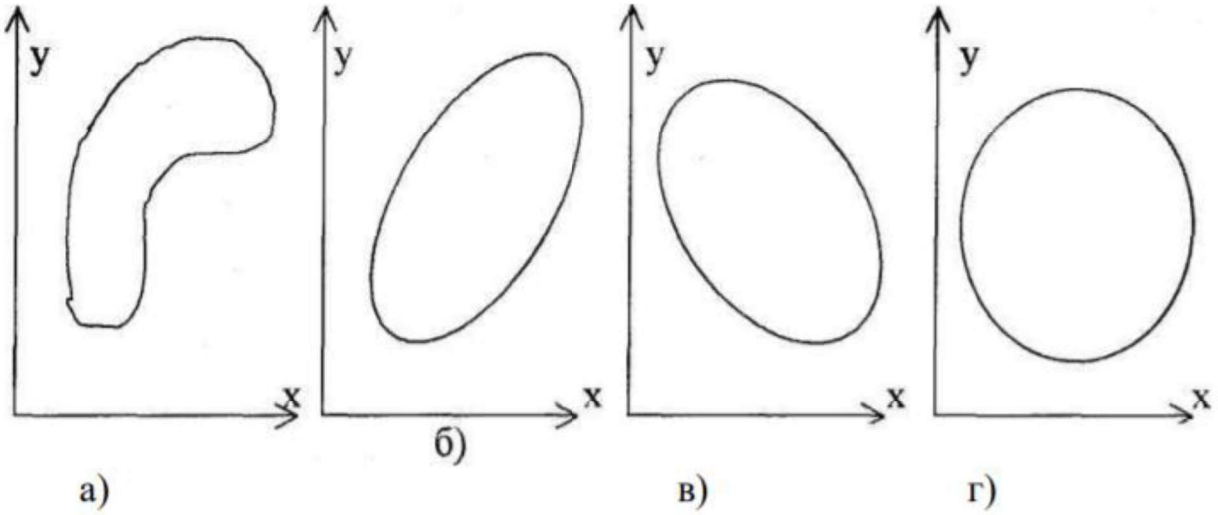

Рис 3.1 Графічне зображення взаємозв’язків

А) не лінійна форма.

Б) Лінійна форма, пряма залежність.

В) Лінійна форма, обернена залежність.

Г) Відсутність взаємозв’язку.

Про силу зв'язку роблять висновок за таким правилом: якщо коефіцієнт кореляції дорівнює 1, то зв'язок повний; якщо він становить 0,66-0,99, то зв'язок сильний; якщо він знаходиться на проміжку 0,33-0,66 - середній; якщо коефіцієнт кореляції менший за 0,33, то зв'язок слабкий.

Для оцінки статистичного взаємозв'язку при лінійному взаємозв’язку використовують коефіцієнт кореляції Браве-Пірсона, котрий обчислюють за формулою[6]:

$$r=\frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\left[n\sum_{i=1}^{n}x_i^{\,2}-\left(\sum_{i=1}^{n}x_i\right)^2\right]\left[n\sum_{i=1}^{n}y_i^{\,2}-\left(\sum_{i=1}^{n}y_i\right)^2\right]}}$$

(3.1)

Де x_i та y_i значення і-того результату виміру.

Про напрям зв'язку висновок роблять залежно від знаку при коефіцієнті кореляції: якщо він додатній, то кореляція пряма, а якщо від'ємний, то зворотна.

Якщо виміри виконані в шкалі порядку (ряд – впорядкований за зростанням або спаданням послідовності величин, що характеризують досліджувану властивість), за допомогою коефіцієнта кореляції Спірмена, котрий обчислюють за формулою (3.2) визначають взаємозв'язок величин шкали

$$r_s = 1 - \frac{6\sum d_i^2}{n(n^2-1)} \tag{3.2}$$

Де d різниця рангів даної пари показників x_i та y_i.

і – місце виміру в ряді (індекс показника в ряді).

n- число вимірів.

Коефіцієнт детермінації визначає частину проміжку зміни одного показника, який зумовлений зміною іншого показника. Обчислюється за формулою:

$$d = r^2 \tag{3.3}$$

Відсоткове значення даного коефіцієнту показує скільки випадків зміни коефіцієнту х призводять до зміни коефіцієнту у.

3.1.2 Регресійний аналіз

Регресійний аналіз – використовують як алгоритм прогнозування результату зміни певного параметра та його ідентифікації.

Регресійний аналіз виконують на основі створеного рівняння що описує певну регресію і визначають вплив незалежних змінних у зміну досліджуваної залежної величини.

Рівняння регресії відображає середнє значення зміни результативної ознаки ($У_x$) під дією факторних ознак (x_i).

Загальний вигляд рівняння регресії[6]:

$$Y_x = f(x_1, x_2, \ldots, x_n) \quad ,$$

(3.4)

x – незалежні змінні величини.

де $У_x$ –змінна величина;

Одним із найпростіших прикладів регресії є лінійна регресія. Проста лінійна регресія - це регресія, яка залежить від однієї незалежної змінної ознаки. Вона випливає з рівняння прямої:

$$Y = mx+c$$

(3.5)

Де, m = коефіцієнт регресії.

Y = цільова, залежна змінна.

c = постійна.

x = незалежна чи предикторна змінна.

Взаємозв’язок між зміною значень кількох залежних змінних під дією багатьох незалежних змінних визначається лінійною регресією з декількома змінними. З цього випливає нижче наведене рівняння прямої, де залежні змінні являються комбінацією всіх інших незалежних змінних:

$$Y = m1x1+m2x2+m3x3+\ldots+mnan+c$$

(3.6)

Де, x1, x2, x3… xn = незалежні змінні.

c = постійна

Y = цільова, залежна змінна.

m1, m2, m3… mn = Коефіцієнти регресії змінних.

Нелінійна регресія – регресія котра залежить від функції із кількома незалежними параметрами.

Види нелінійної регресії

- поліноміальна (означає наближення даних x_i та y_i поліномом k – ступеня.
- Гіперболічна $Y=a+(b/x)$ (3.7)
- Ступенева $Y = a*x^b$ (3.8)
- Експоненційна $Y = b_o*e^{b1x}$ (3.9)
- Показова $Y = a*b^x$ (3.10)

Множинна регресія – це регресія, корта відображає зв'язок між однією незалежною зміною та кількома залежними.

Рівняння має вигляд

$$y = a_0 + a_1x_1 + a_2x_2 + \ldots + a_mx_m,$$

(3.11)

Де y – значення залежної змінної.

x1, x2…xn- значення факторних ознак.

a0,a1…..an – коефіцієнти регресії.

Параметри рівняння регресії часто визначають за допомогою методу найменших квадратів.

3.1.3 Модель “Ковзне середнє”

Ковзне середнє (MA) – інструмент, що дозволяє згладжувати часові ряди, застосовується для зображення змін, цін на товари, кількість населення і так далі. Ковзне середнє – один з найстаріших і найбільш поширених засобів аналізу часових рядів. Він відображає середнє значення величини за обраний період часу.

Види ковзних середніх:

- просте ковзне середнє;
- зважене ковзне середнє;
- експоненційне ковзне середнє;
- синус-зважене ковзне середнє;
- ковзне середнє кінцевої точки;
- адаптивне ковзне середнє;
- трикутне ковзне середнє;

З яких найбільш поширеними є перші три.

Просте ковзне середнє (SMA) – є одними з найбільш часто використовуваних і не складних показників в технічному аналізі. Воно є середнім арифметичним від величин за обраний період. Також ковзне середнє називають лінією тренду.

Зважене ковзне середнє (WMA). - являється модифікацією SMA з використанням ваг коефіцієнтів, підібраними таким способом щоб надати останнім коефіцієнтам більшу вагу, а старішим – меншу вагу. Таким чином цей метод ліквідовує головний недолік SMA методу.

Зважене ковзне середнє визначається за формулою 3.12.

$$WMA = \frac{\sum_{i=1}^{n} P_i * W_i}{\sum_{i=1}^{n} W_i}$$

(3.12)

де: Wi – значення ваг для ціни і-періодів тому

Рі– значення ціни і-періодів тому, (і сьогодні = 1);[5].

Експоненційне ковзне середнє (ЕМА) зменшує помилку, надаючи більше значення ваги останнім значенням ряду у порівнянні з більш далекими значеннями. Цей метод дозволяє швидко реагувати на нові зміни значень в порівнянні з SMA. Вага, що надається останньому значенні, залежить від періоду ковзної середньої. Чим менший період ЕМА, тим значиміше значення ваги надаватиметься останньому значенню[6].

Формула виглядає наступним чином.

$$EMA = \frac{EMA_{i-1} * (n-1) + 2 * P_i}{n+1}$$

(3.13)

де ЕМА – експонентна ковзна середня;

n – період розрахунку;

Рі – значення ціни в і-му періоді;

ЕМАі−1 – значення ЕМА і-1 періоду.

В обчисленні ЕМА використовуються всі величини, за весь період її побудови. Вплив старих величин поступово зменшується з часом, проте його вплив на прогнозоване значення не зникатиме до кінця. Ефект впливу старих значень пропадає швидше для коротких рядів, в порівнянні з більш довгими.

3.1.4 Моделі авторегресії і ковзного середнього ARIMA

Найбільш розповсюдженим методом прогнозування часових рядів є побудова статистичних моделей. Найпростішою з цих моделей є модель AR(Autoregressive, авторегресійна). Модель AR(p) описується формулою 3.14[5].

$$y(t) = c + \sum_{i=1}^{p} a_i y(t-i) + \varepsilon(t) \quad (3.14)$$

де:

– $y(t)$ – значення часового ряду в момент t;

– c – константа;

– p – порядок моделі;

– ai – коефіцієнти моделі;

– $\varepsilon(t)$ – білий шум

Ця модель в якості параметрів для прогнозування використовує попередні значення часового ряду. Вона має деякі обмеження на використання, наприклад необхідність того, щоб часовий ряд був стаціонарним[3].

Стаціонарний ряд – це ряд, який має стале середнє значення, або це значення може змінюватись у певних мінімальних проміжках. Ряд вважається стаціонарним, коли середнє значення, дисперсія і значення варіації ряду дисперсії y_i є такі ж, як і для y_{i+1}. Якщо ці показники змінюються із часом то ряд нестаціонарний.

Продовженням авторегресійної моделі є модель ARMA (Autoregressive Moving Average model, авторегресійна модель з ковзним середнім). Модель ARMA(p, q) описується формулою 3.15[6].

$$y(t) = c + \sum_{i=1}^{p} a_i y(t-i) + \sum_{j=1}^{q} b_j \varepsilon(t-j) + \varepsilon(t)$$

(3.15)

де: $y(t)$ – значення часового ряду в момент t;

c – константа;

p – порядок моделі;

ai – коефіцієнти моделі;

$\varepsilon(t)$ – білий шум.

При моделюванні нестаціонарних процесів авторегресійна модель об'єднюється з іншими методами аналізу: ковзнім середнім, трендом, сезонною хвилею. Об'єднання різних моделей в одне ціле збільшує сферу застосування. Об'єднані моделі формуються на основі одних і тих же статистичних даних. Моделі такого класу називають об'єднаними (інтегрованими) моделями авторегресії.

Побудова ARIMA може використовуватись як допоміжний інструмент для обрахунку майбутніх значень окремих факторів. Дані фактори можуть мати вплив на зміну величини залежного показника, на час прогнозування за допомогою багатофакторних моделей із регресійними значеннями. Можна будувати різновиди ARIMA моделей. Таким різновидом є модель ARMAX. Вона може враховувати і додаткові параметри в різних формах досліджуваного показника разом із лаговими змінами [4].

Для ARIMA аналізу необхідно знати: одиницю виміру, хронологічні межі ряду, регулярність, частоту вимірів (періодичність).

За допомогою процедури Хенона–Рісанена визначають порядок AR та MA складових при умові, що процес не є процесом ковзнього середнього або авторегресійним.

При цьому оцінюють авторегресійну складову ARMA/ARIMA моделі використовуючи метод найменших квадратів. Щоб отримати найкращі значення лагів моделі потрібно знайти мінімальне значення критерію AIC для цих лагів.

Процедуру Хенона–Рісанена починають з послідовного оцінювання AR моделей різного порядку. Для кожної моделі виконують аналіз значення AIC критерію. Найкращим значенням обирають таке значення лагу, при якому буде обчислене найменше значення AIC критерію.

AIC критерій (Інформаційний критерій Акаіке) це оцінювач похибки позавибіркового передбачення і відносної якості статистичних моделей, для заданого набору даних[5].

Висновок до розділу

В даному розділі описано процеси аналізу часових рядів, а саме кореляційний, регресійний аналізи. Описано Модель ковзнього середнього, модель авторегресії і ковзнього середнього (ARIMA). Їх переваги та недоліки один над одним, способи та випадки застосування.

РОЗДІЛ 4. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Створення баз даних

Для збереження даних я використав базу даних PostrgreSQL. PostgreSQL не керується якоюсь однією компанією, її розробка здійснюється через співпрацю багатьох людей та компаній, які використовують цю СКБД, що дозволяє швидко впроваджувати у неї найкращі досягнення у сфері інформаційних технологій.

Створив базу даних зображену на рис 4.1

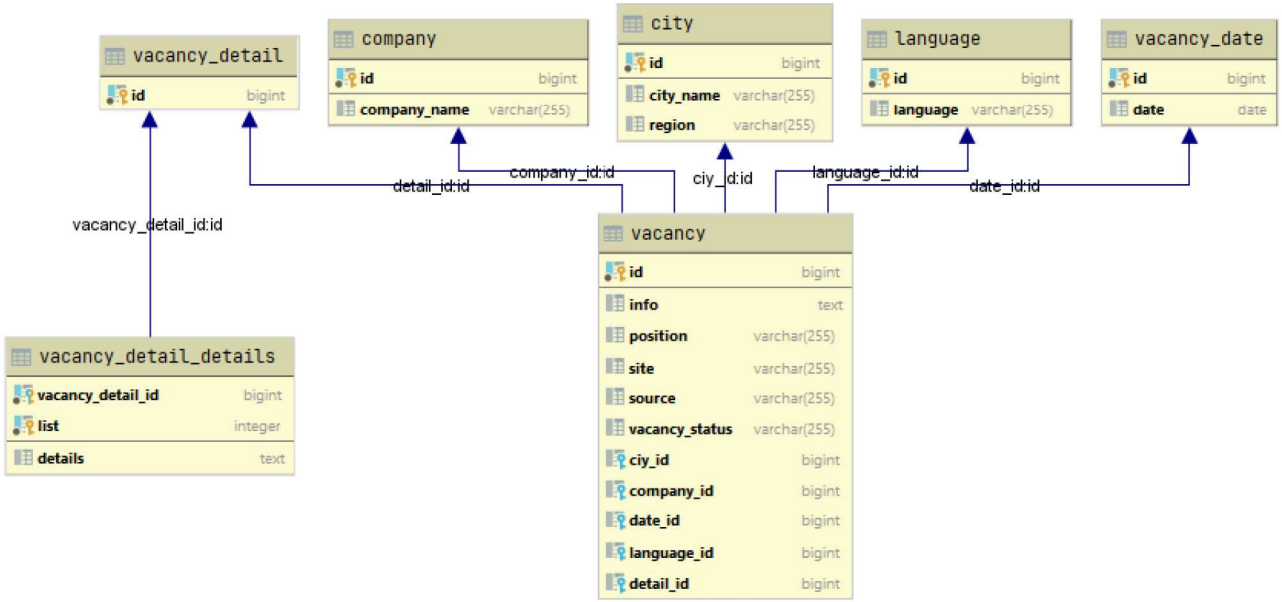

Рис 4.1

Основною таблицею є таблиця “vacancy”. Вона призначена для зберігання основної інформації про вакансію. Містить такі поля: посада, коротка інформація, назва сайту, посилання на вакансію, ID компанії, ID міста, ID дати, ID мови програмування, ID деталей вакансії, статус. Поле статус являється міткою для позначення не актуальних вакансій з метою їх не відображення користувачу. При отриманні нових вакансій усі попередні позначаються як “IRRELEVANT” та видаляється в них інформація про деталі вакансії, та опис з метою зменшення зайнятого простору у базі даних. Нові вакансії позначаються як “ACTUAL”.

Оскільки у різних вакансіях часто повторялись такі значення як: мова програмування, дата, місто, компанія. З метою нормалізації бази даних їх було винесено у окремі таблиці.

Таблиця “language” призначена для збереження мови або напряму програмування. Зв’язок із таблицю “vacancy” один до багатьох.

Таблиця “vacancy_date” містить дати зберігання вакансій із стороннійх джерел. Зв’язок із таблицю “vacancy” один до багатьох.

Таблиця “city” призначена для зберігання міста. Зв’язок із таблицю “vacancy” один до багатьох.

Таблиця “company” призначена для зберігання компанії. Зв’язок із таблицею “vacancy” один до багатьох.

Таблиця “vacancy_detail_details” призначена для зберігання детального опису вакансій. З метою кращого об’єднання різних вакансій із різних джерел зв’язок між цією таблицею та таблицею “vacancy” – багато до багатьох. Він реалізується за допомогою наявності таблиці “vacancy-detail”, котра поєднує записи із таблиць.

Для взаємодії програми і бази даних в конфігураційному файлі вказав такі властивості: тип драйвера, логін і пароль, шлях до бази даних. Вказав автоматично оновлювати структуру таблиці при зміні полів у класах, діалект SQL рис 4.2.

```
# Datasource
spring.datasource.url=jdbc:postgresql://localhost:5432/forecastIT
spring.datasource.username=username

spring.datasource.password=root

spring.datasource.driver-class-name=org.postgresql.Driver

# Hibernate

spring.jpa.hibernate.ddl-auto=update

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQL9Dialect

spring.jpa.show-sql=true
```

Рис 4.2

Для кращої взаємодії створив java класи, що будуть являти собою таблиці із бази даних. Це дозволить зручно керувати даними та зменшить кількість програмного коду, котрий перетворював отримані значення після виконання SQL запитів.

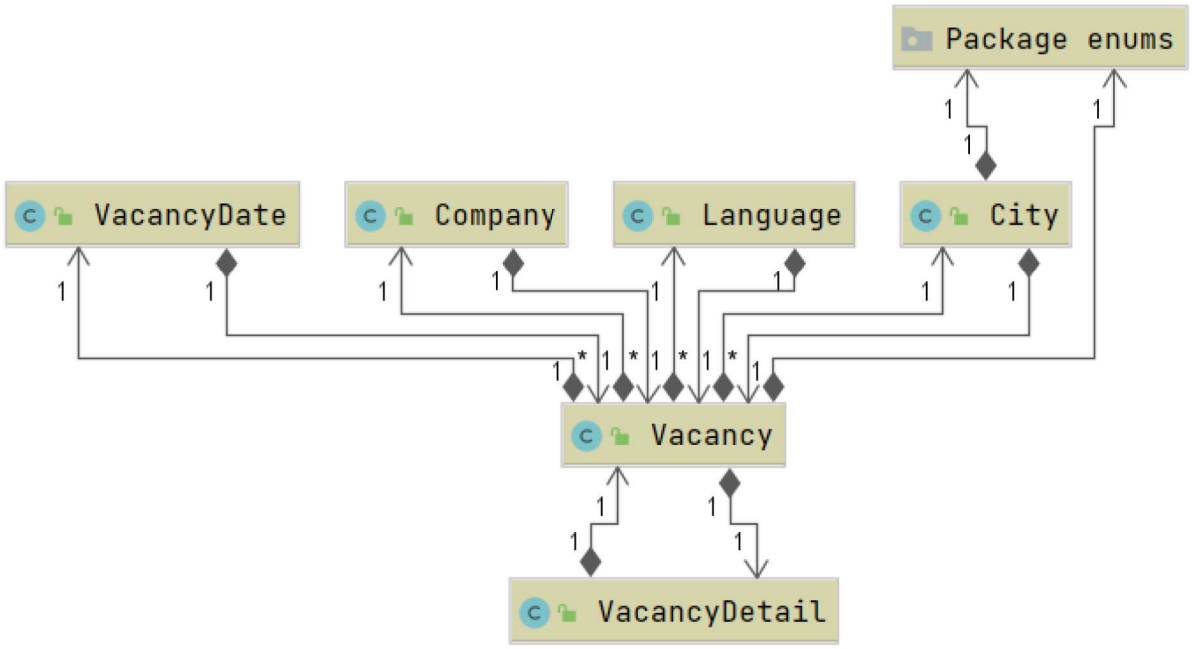

Рис 4.3 Діаграма класів представлень таблиць бази даних.

Як можна побачити на рисунку 4.3 взаємозв'язки класів повторяють взаємозв'язки таблиць у базі даних.

4.2 Створення серверної частини програми

Серверна частина програми призначена для отримання, збереження та опрацювання вакансій та прогнозування. Ця частина програми виконує одержані повідомлення із клієнтської частини і відправляє відповідь на клієнтську частину згідно із повідомлення.

Серверна частина побудована на основі MVC архітектури.

Загальна структура проєкту зображена на рис 4.4

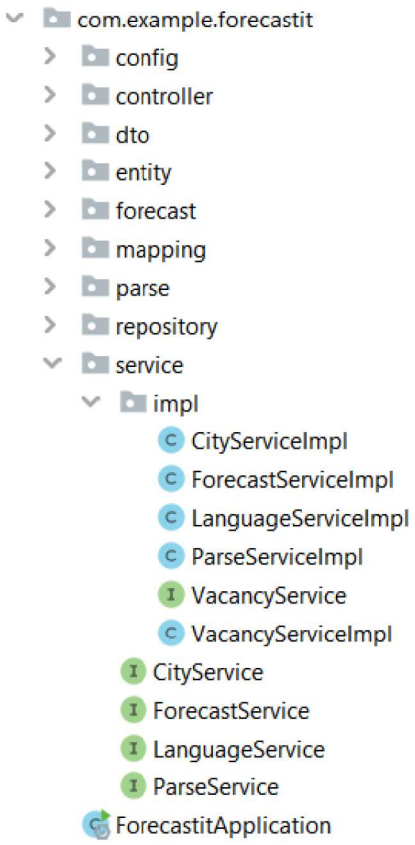

Рис 4.4

Весь програмний код, для зручності розробки і розуміння програми, поділений на пакети, котрі містять класи, що маюсь спільне призначення.

- Config – містить конфігураційні класи
- Controller – містить класи, для спілкування із клієнтською частиною.
- Dto – містить набір класів, що використовуються для передачі даних між серверною і клієнтською частинами.
- Entity – містить класи, які представляють собою таблиці у базі даних.
- Forecast – містить класи, які реалізують алгоритми прогнозування.
- Mapping – класи які перетворюють об’єкти класів із пакету dto в об’єкти класів із пакету entity і навпаки.
- Parse –класи цього пакету призначені для пошуку та отримання інформації із зовнішніх ресурсів.
- Repository – цей пакет мітить інтерфейси призначені для робити із базою даних.

- Service – цей пакет містить інтерфейси та класи, які містить в собі основну логіку для роботи програми.

Для об'єднання вакансії із різних сайтів в один загальний формат я описав власний клас Vacancy

Клас Vacancy об'єднює вакансії із різних сайтів під єдиний формат.

```
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private Long id;

@Column(columnDefinition = "text")
private String info;
private String position;

private String site
private String source;;

@Enumerated(value = EnumType.STRING)
@Column(name = "vacancy_status")
private VacancyStatus vacancyStatus;

@ManyToOne
@JoinColumn(name = "company_id")
private Company company;

@ManyToOne
private VacancyDate date;

@ManyToOne
@JoinColumn(name = "ciy_id")
private City city;

@OneToOne(cascade = CascadeType.ALL)
@JoinColumn(name = "detail_id")
private VacancyDetail vacancyDetail;

@ManyToOne
@JoinColumn(name = "language_id")
private Language language;
```

Анотації @OneToOne, @ManyToOne позначають типи зв'язків між таблицями у базі даних. @Id та @GeneratedValue позначає поле, що слугує ідентифікатором запису та спосіб автоматичного створення ідентифікатора[1].

Для отримання інформації із інших ресурсів я використав можливості бібліотеки Jsoup. Створив клас ParseData, котрий містить методи зчитування вакансій із сайтів.

Приклад методу парсингу вакансій із сайту dou.ua

```
public ArrayList<Vacancy> parseVacancyFromDou(String city, String language,
String url, VacancyDate date) throws IOException {
        Document pageDou = Jsoup.parse(new URL(url), 50000);
```

Даний метод приймає на вхід рядкові величини місто, мова/напрям програмування, посилання на конкретну сторінку. Дане посилання отримується на основі сканування початкової сторінки із вакансіями.

Даний приклад коду зображає отримання елементів із сторінки за допомогою назви класу елемента у DOM.

```
Elements elementsFromPage = pageDou.getElementsByClass("l-vacancy");
ArrayList<Vacancy> vacancyArrayList = new ArrayList<>();
```

При отриманні вакансії відбувається перевірка на наявність дублікату у базі даних та перевіряється компанія на її наявність у базі даних. Якщо назва компанії, яка визначена у вакансії уже записана, тоді не буде створюватись новий запис у таблиці Company, а буде отримано та підставлено ідентифікатор компанії із бази даних у таблиці Vacancy.

```
for (Element element:elements){
        Vacancy parsedVacancy = new Vacancy();
        parsedVacancy.setPosition(element.getElementsByClass("vt").text());
        parsedVacancy.setCity(cityRepository.getCityByNameCity(city));
        String companyName = e.getElementsByClass("company").text();
        Company company = companyRepository.
getCompanyByCompanyName(companyName);
        if (company==null){
            company = new Company();
            company.setComapnyName(companyName);
            company = companyRepository.save(company);
        }
        parsedVacancy.setCompany(company);
        String info = element.getElementsByClass("sh-
info").text().toString();
        System.out.println(info);
        parsedVacancy.setInfo(info);
        Elements elvt = e.getElementsByClass("vt");
        List<String> elhrefs = elvt.eachAttr("href");
        parsedVacancy.setSource(elhref.get(0));
        parsedVacancy.setDate(date);
        VacancyDetail vacancyDetail = parseDouDetail(elhrefs.get(0));
```

Метод, який зчитує детальний опис вакансії на сторінці.

```
parsedVacancy.setVacancyDetail(vacancyDetail);
parsedVacancy.setLanguage(languageRepository.getLanguageByLanguage(language));
        parsedVacancy.setSite("dou");
        vacancyArrayList.add(vacancy);
        parsedVacancy.setVacancyStatus(VacancyStatus.ACTUAL);
```

Збереження вакансії у базу даних

```
        vacancyRepository.save(parsedVacancy);
}
```

Аналогічно отримуються дані і з інших сайтів.

Перед викликом методу, котрий збирає дані із сторонніх сайтів спрацьовує метод, котрий змінює статус старим вакансіям, та видаляє про них частину інформації для оптимізації даних. Записи про вакансії потрібні для відображення статистики та прогнозування, тому все не видаляється. Приклад даного методу.

```
private void upadteStatusOldVacancy(){
    List<Vacancy> vacancies = vacancyRepository.getAllActual();
    for (Vacancy vacancy:vacancies ) {
        vacancy.setVacancyStatus(VacancyStatus.IRRELEVANT);
        vacancyRepository.deleteUnusedData(vacancy);
        vacancyRepository.save(vacancy);
    }
}
```

Для виклику методу, який зчитує дані із сторонніх ресурсів створив контролер.

```
@GetMapping("/addVacancy")
    public List<VacancyDto> writetestData(){
    parseService.parseData();
    return vacancyService.getAll();
}
```

Після завершення роботи методу parseData() викликається метод що повертає список всіх актуальних вакансій.

Для роботи із таблицею у базі даних для роботи із вакансіями створив інтерфейс VacancyRepository, що містить набір стандартних методів для роботи із базою даних і містить відповідні запити до бази даних такі, як видалення, збереження та отримання всіх записів чи по ідентифікатору.

```
@Repository
public interface VacancyRepository extends JpaRepository<Vacancy, Long> {
```

Анотація @Repository вказує фреймворку, що цей інтерфейс працюватиме із базою даних. Для отримання доступу до стандартних методів роботи із базою даних він успадковується від інтерфейсу JpaRepository, в який вказується тип таблиці із якою буде працювати даний інтерфейс та тип ідентифікатора[1].

Проте для отримання вакансій за певною конкретною ознакою потрібно написати власний метод та запит із яким метод буде працювати.

```
@Query("select v from Vacancy v where upper(v.language.language) like
upper(:language)   and v.vacancyStatus='ACTUAL'")
Page<Vacancy> getAllByLanguage(@Param("language") String language, Pageable
pageable);
```

Даний метод повертає список вакансій із статусом ACTUAL. Анотація @Query містить запит до бази даних. Анотація @Param дозволяє динамічно передавати значення із параметра методу у запит[1]. Також він повертає потрібну кількість елементів для однієї сторінки за допомогою інтерфейсу Pageable. Даний інтерфейс містить інформацію про номер сторінки та кількість елементів на ній. При повторному виклику даного метода в межах однієї сесії він поверне наступні вакансії у списку.

Схема алгоритму прогнозування Рис 4.5

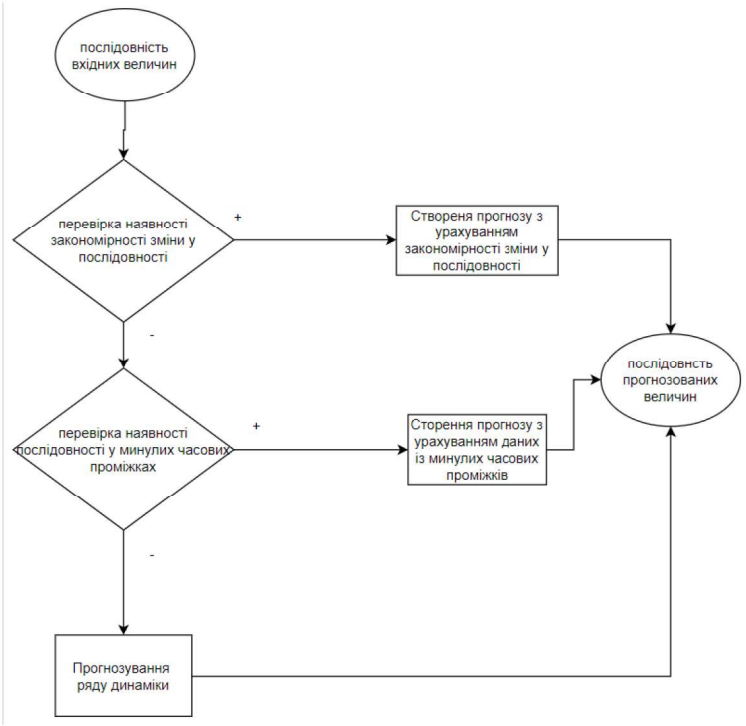

Рис 4.5

Прогноз будується декількома способами залежно від наявності у вхідній послідовності певних закономірностей, а саме: наявність однакового кроку зміни послідовності на певній кількості останніх вхідних величин, наявність послідовності останніх вхідних величин у попередніх часових проміжках не враховуючи проміжки із яких підмножина була взята. Зазвичай використовував п’ять останніх вхідних величин. Якщо ці умови не виконуються тоді прогноз створюється за допомогою вирівнювання часового ряду шляхом усереднення та

експонентного вирівнювання та модифікації коефіцієнтів, які будуть давати кращий результат.

Метод прогнозування, який відповідає наведеній діаграмі на рис 4.5

```
public List<Long> forecast(List<Long> data) {
    List<Long> forecastList = growthIncrease(data);
    if (forecastList != null) {
        return forecastList;
    }
    forecastList = getDataBySublist(data);
    if (forecastList!=null){
        return forecastList;
    }
    else {
        return forecastExp(data);
    }
}
```

Для зручності основні алгоритмічні компоненти виніс у окремі методи.

```
    public List<Long> forecastExp(List<Long> data) {
        double a = 0.2;
        List<Long> forecastList = new ArrayList<>();

        for (int i=0;i<3;i++){
            List<Double> vurivn = this.exponentaVurivn(data);
            double s = this.avgsqrtVidhulenia(data);
            double t = 1.64;
            double deltaY = t * s * Math.sqrt((a / (2.0 - a)));
            System.out.println("deltaY");
            System.out.println(deltaY);

            double ynp1 = vurivn.get(vurivn.size() - 1) + deltaY;
            System.out.println("forecast1");
            System.out.println(ynp1);
            forecastList.add(Math.round(ynp1));
            data.add(Math.round(ynp1));
        }

        return forecastList;
    }
```

В даному прикладі коду наведено спосіб прогнозування, якщо перші дві умови вище згадані не виконуються. В даному методі використовуються дві константи:

a – коефіцієнт згладжування вибирається на проміжку [0.1; 0.3].

t - коефіцієнт Стьюдента.

Даний метод являється реалізацією формул 4.1 та 4.2 та 4.3[6]

$$y_{n+1} = \tilde{y}_n \pm t^* s \sqrt{1 + \frac{\alpha}{2-\alpha}}.$$

(4.1)

$$\tilde{y}_{n+1} = \tilde{y}_n + \Delta\tilde{y} \ ,$$

(4.2)

$$\Delta\tilde{y} = \pm t^* s \sqrt{\frac{\alpha}{2-\alpha}},$$

(4.3)

Для можливості відправлення даних на клієнтську частину створив класи, які будуть описувати дані для передачі на клієнтську частину. Також створив сервіси, які оброблятимуть відповідні дані. Сервіси поділені на інтерфейси та їх реалізацію. Це зроблено для того, щоб можна було створити об'єкти Spring або біни. Створив контролери для взаємодії із клієнтською частиною.

Приклад класу із контролерами, що містять посилання для отримання прогнозованих даних.

```
@CrossOrigin(origins = "*", allowedHeaders = "*")
@RestController
public class ForecastController {

    @Autowired
    private ForecastService forecastService;
    @Autowired
    private LanguageService languageService;
```

Даний метод приймає два параметри у посиланні із яким він працює. В залежності від наявності цих параметрів і які вони мають значення буде сформовано відповідний прогноз за допомогою написаних сервісів прогнозування.

```
    @GetMapping("/forecast")
    public List<LanguageForecastDto> forecast(@RequestParam(required = false,
value = "languageId") Long langId,
                                              @RequestParam(required = false,
value = "cityId") Long cityId){
        if (cityId!=null){
```

```
            return forecastService.predictByCity(cityId);
        }else if (langId!=null){

            return forecastService.predictByLanguage(langId);
        }else {

            return forecastService.predictByCountry();
        }
    }
```

Для взаємодії класу контролера та класу, що містить реалізацію алгоритму прогнозування створив сервіс через який вони будуть зв'язуватись між собою.

```
@Service
public class ForecastServiceImpl  implements ForecastService {
```

Анотація @Service показує фреймворку, що даний клас є шаблоном для створення об'єкта Spring [7].

Створив метод, котрий буде отримувати потрібні дані для прогнозування із бази даних та викликатиме методи прогнозування і передаватиме на контролер прогноз.

```
@Override
    public List<LanguageForecastDto> predictByLanguage(Long languageId) {
        List<VacancyDate> dates = dateRepository.findAll();
        VacancyDate lastDate = dateRepository.getOne(dateRepository.count());
        Language language = languageRepository.getOne(languageId);
        System.out.println(language.toString());
        List<Long> counts = new ArrayList<>();
        for (VacancyDate date: dates){

counts.add(vacancyRepository.getAllVacancyByDateIDAndLanguageID(date.getId(),lan
guageId));
        }
        List<Long> forecastData =  forecast.forecast(counts);
        List<ForecastDto> forecastDtos = new ArrayList<>();
        int count = 1;
        for (Long l:forecastData){
            forecastDtos.add(ForecastDto.builder()
                .date(addWeek(lastDate.getDate(),count))
                .count(l)
                .build());
            count++;
        }

        List<LanguageForecastDto> languageForecastDtos = new ArrayList<>();
        languageForecastDtos.add(LanguageForecastDto.builder()
            .language(language.getLanguage())
            .forecastDtoList(forecastDtos)
            .build());
        return languageForecastDtos;
    }
```

Для перевірки точності роботи алгоритму прогнозування створив сервіс, що буде запускати алгоритм на перевірку. Даний метод отримує потрібну

інформацію для перевірки та відбирає три останні значення. Запускає алгоритм прогнозування (без урахування останніх записів) і порівнює їх із відібраними.

```
@Service
public class ForecastingAccuracyServiceImpl implements
ForecastingAccuracyService {
@Override
    public List<AccuracyPredictDto> accuracyPredictByLanguage(Long languageId) {

        List<VacancyDate> dates = dateRepository.findAll();
        VacancyDate lastDate = dateRepository.getOne(dateRepository.count());
        Language language = languageRepository.getOne(languageId);

        List<AccuracyPredictDto> accuracyPredictDtos = new ArrayList<>();
        List<Long> counts = new ArrayList<>();
        List<Long> actualCounts = new ArrayList<>();
        for (VacancyDate date : dates) {

counts.add(vacancyRepository.getAllVacancyByDateIDAndLanguageID(date.getId(),
language.getId()));
        }
        actualCounts.addAll(counts.subList(counts.size() - 4, counts.size() -
1));
        for (int i = 1; i <= 3; i++) {
            counts.remove(counts.size() - 1);
        }
        List<Long> forecastData = forecast.forecast(counts);
        List<AccuracyPredictCountDto> accuracyPredictCountDtos = new
ArrayList<>();

        int count = 3;
        int expectedIndex = 0;

        for (Long l : forecastData) {
            Long actual = actualCounts.get(expectedIndex);
            Long diference = Math.abs(actual - l);
            double persDif = 100 - (diference * 100) / actual;
            accuracyPredictCountDtos.add(AccuracyPredictCountDto.builder()
                .date(minusWeek(lastDate.getDate(), count))
                .actual(actual)
                .expected(l)
                .accuracy(persDif)
                .build());
            expectedIndex++;
        }
        accuracyPredictDtos.add(AccuracyPredictDto.builder()
            .languageName(language.getLanguage())
            .predictCount(accuracyPredictCountDtos)
            .build());
        return accuracyPredictDtos;
    }
```

4.3 Створення клієнтської частини програми

клієнтську частину програми я створив, використавши мову програмування TypeScript та фреймворк Angular.

Створив новий проєкт. Основними елементами проєкту створеного на Angular є сервіси та компоненти. Сервіси містять основну логіку та працюють із

API. Компоненти являють собою набір файлів котрі мають своє призначення. Файли із розширеннями html являють собою частину інтернет сторінки. CSS файли із стилями. Вони можуть бути застосовані лише до html файлу, котрий знаходиться в одній і тій же компоненті. Файл із розширеннями ts містить TypeScript код для наповнення сторінки даними[9].

Створення нових компонент або сервісів потрібно виконувати за допомогою відповідних команд у консолі.

Приклад команди створення компоненти ng generate component vacancy-list або її скорочений варіант ng g c vacancy-list[9].

У файлі app.module прописав маршрутизацію для окремих компонент

```
onst appRoutes: Routes = [

{
  path: 'vacancy-list',
  component: VacancyListComponent
},
{
path: 'vacancy/:id',
component: VacancyComponent
},
{
  path: 'vacancy-statistics', component: StatisticsComponent
},
  {
    path: 'vacancy-forecast', component: ForecastComponentComponent
  },
  {
    path: 'home', component: HomeComponent
  },
  {
    path: 'accuracy', component: ForecastAcurracyComponent
  },
{path: '/', redirectTo: '/home', pathMatch: 'full'},
];
```

Створив сервіс для отримання даних про вакансії із серверної частини.

```
export class VacancyServiceService {
 public API = '//localhost:8080/vacancy/'

  constructor(private http:HttpClient) {
   }
   getAllVacancy():Observable<any>{
     return this.http.get(this.API+"all");
   }
   getAllVacancyByID(id:String){
     return this.http.get(this.API+id);
   }
```

В даному прикладі присутні методи для отримання всіх вакансій та однієї за її ідентифікатором.

Аналогічно створив сервіси для отримання статистики та прогнозування.

Приклад методу для отримання даних прогнозу.

```
getPredictionByAllLanguageAndCityID(id): Observable<any>{
 return  this.http.get(this.API + 'forecast?languageId=&cityId=' + id);
}
```

Створив компоненту forecast-component та додав змінну для зберігання даних, які будуть отримані із серверної частини.

```
forecasts: Array<any>;
```

Приклад методу, який викликає сервіс та отримує дані і їх записує у відповідну змінну.

```
getPredictionbyLanguageID(id){
  this.forecastService.getPredictionByLanguage(id).subscribe(data => {
    this.forecasts = data;
  });
}
```

На html сторінці цього компонента створив таблицю для відображення даних.

```
<div class="col-sm-9">
  <table class="table table-striped" >
    <tr class="table-header">
      <td>Мова/напрям</td>
      <td>Дата</td>
      <td>Прогноз</td>
      <td>Дата</td>
      <td>Прогноз</td>
      <td>Дата</td>
      <td>Прогноз</td>
    </tr>
    <ng-container *ngFor = " let f of forecasts" >
      <tr class="table-row">
          <td>{{f.language}}</td>
        <ng-container *ngFor="let dto of f.forecastDtoList">
          <td> {{dto.date}}</td>
          <td> {{dto.count}}</td>
        </ng-container>
      </tr>
    </ng-container>

  </table>
</div>
```

В даній таблиці відображатимуться дані прогнозу на 3 одинакові періоди часу які є незмінними.

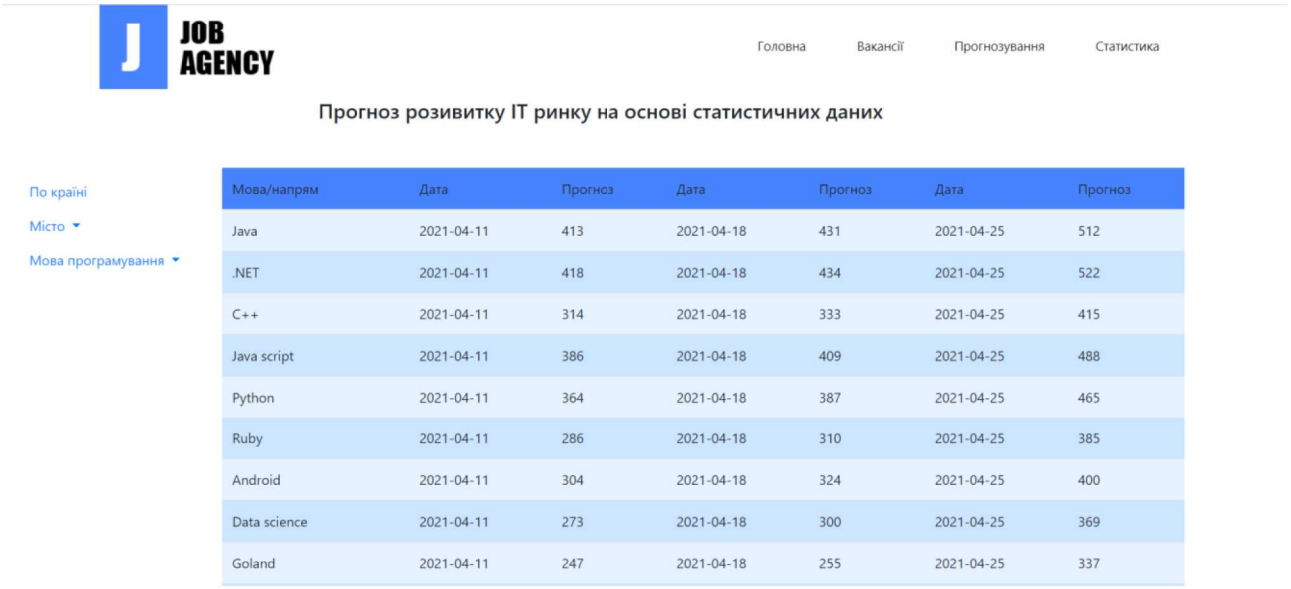

Рис 4.6 Сторінка відображення прогнозування

Як зображено на рис 4.6 користувач має змогу обирати категорії по яких будуть створюватись прогнози, а саме: може обрати напрям програмування, або місто чи отримати дані для всіх напрямів у містах України, які збережені у базі даних.

Для відображення всіх вакансій створив нову компоненту Vacancy-List. Створив метод для отримання списку вакансій на сторінку із урахуванням застосування фільтра та паджинації .

```
getVacancyByPage(){

this.vacancyService.getAllVacancyByPage(this.currentPage,this.city,this.technolo
gy).subscribe(data=>{
      this.vacancy = data.page;
      this.pageabledto = data;
      this.totalElem = data.totalElements;
      this.totalPage = data.totalPages;
      this.totalPage = this.totalPage-1;
      this.currentPage = data.number;
      this.getPagesNumber(data.totalPages
          })
```

Для відображення створив html сторінку Рис 4.7

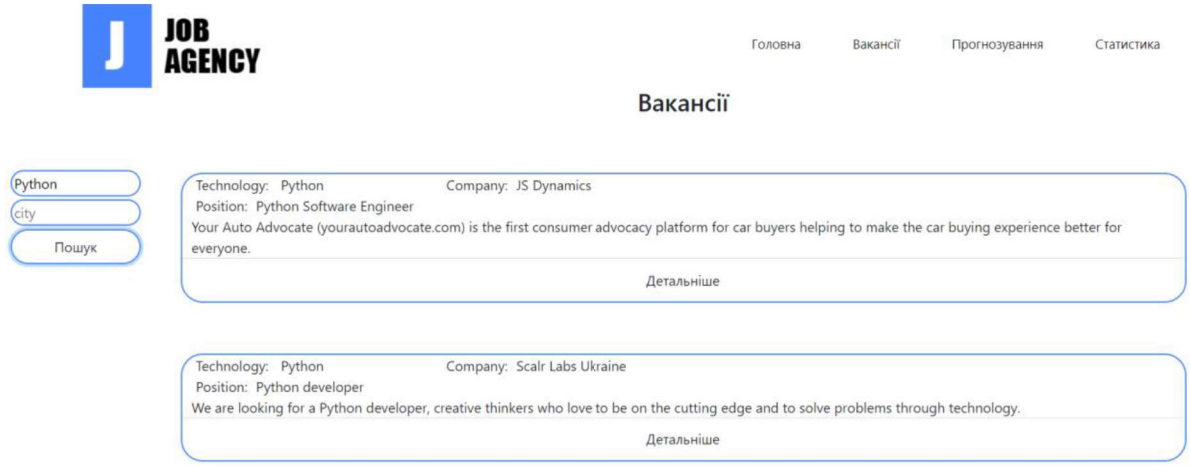

Рис 4.7

При натисканні на кнопку детальніше відбувається перехід на сторінку для відображення опису вибраної вакансії. Приклад методу, що отримує дані із сервісу по одній вакансії, передаючи в сервіс унікальний ідентифікатор.

```
ngOnInit(): void {
 this.subscriber = this.route.params.subscribe(p=>{
  const identifier = p['id'];
  if(id){
   this.vacancyService.getAllVacancyByID(identifier).subscribe(data=>{
    this.vacancy = data;
    })
  }
 });
```

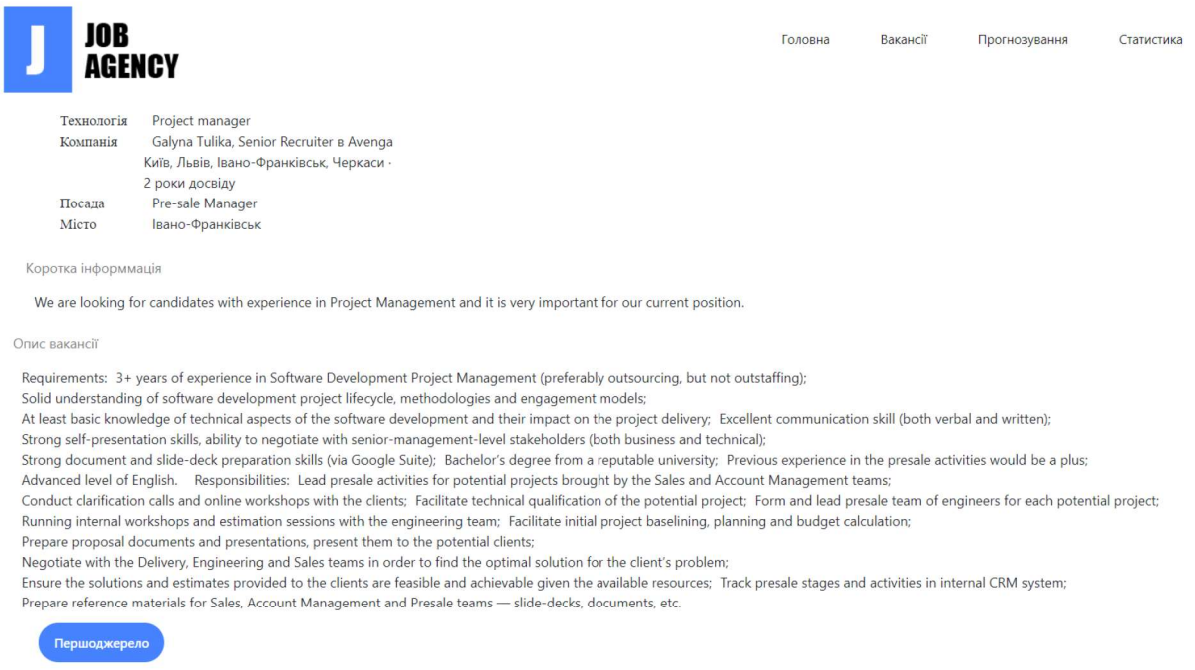

Рис 4.8. Приклад сторінки із описом вакансії

Створив нову компоненту призначену для перегляду статистики. В даній системі користувач може побачити кількість за декілька тижнів, поточний момент часу та побачити “Топ 10 компаній” із найбільшою кількістю вакансій та які напрями є в них затребуваними. Це дозволяє зрозуміти в якому напрямі технологій розвивається та чи інша компанія. Також користувач на кожній із вкладок може виконати фільтрування по місту чи мові програмування.

Приклад отримання статистики за поточний період часу та за декілька останніх тижнів

```
getLanguageCountCurrent() {
  this.statiscticsService.getAllLanguageCount().subscribe(data => {
    this.languageCountCurrent = data;
  });
}

getLanguageCountByDate() {
  this.statiscticsService.getAllLanguageCountByDate().subscribe(data => {
    this.languageCountCurrentByDate = data;
  });
  this.statiscticsService.getAllLanguages().subscribe(data => {
    this.languagesList = data;
  });
}
```

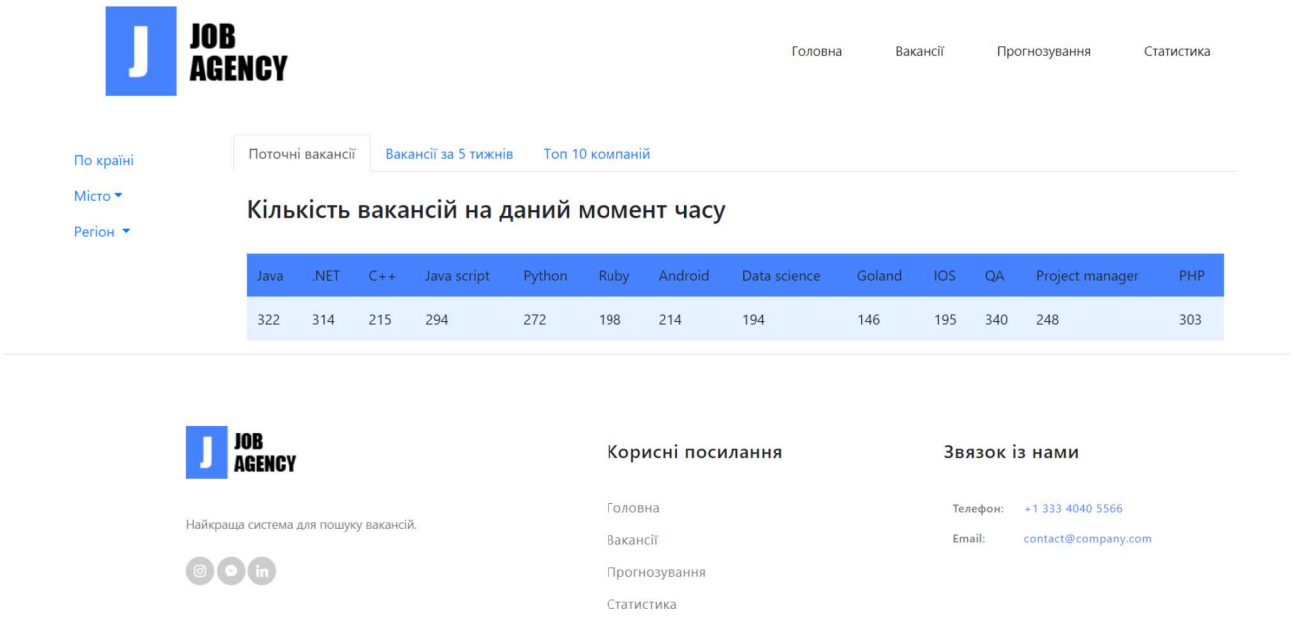

Рис 4.9 Сторінка статистики за поточний період часу.

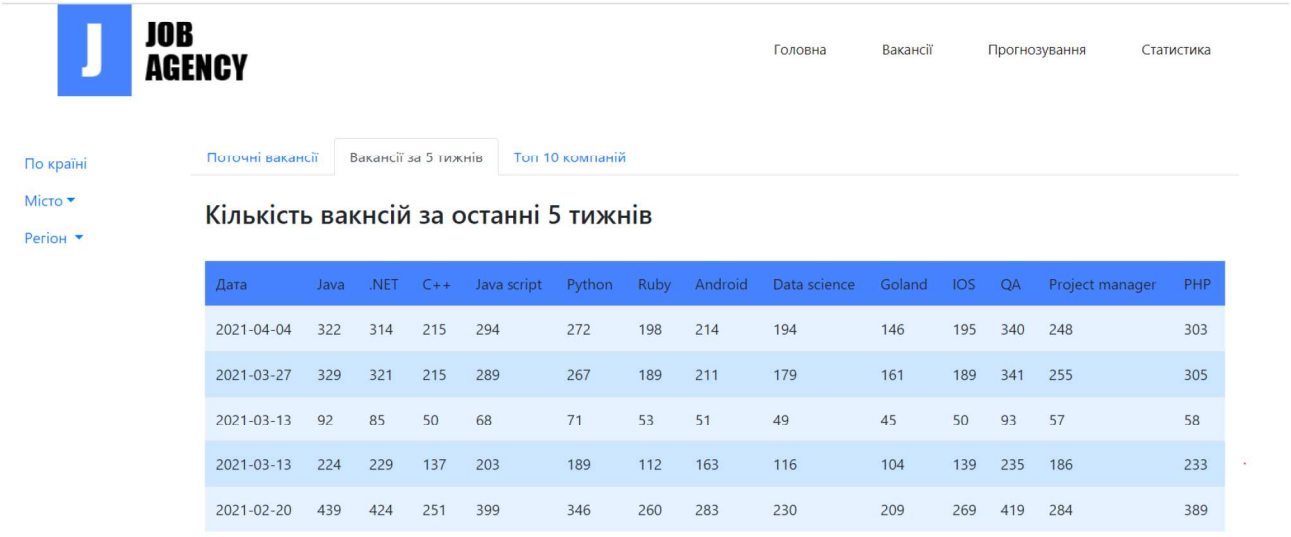

Рис 4.10 Сторінка статистики за останні 5 тижнів.

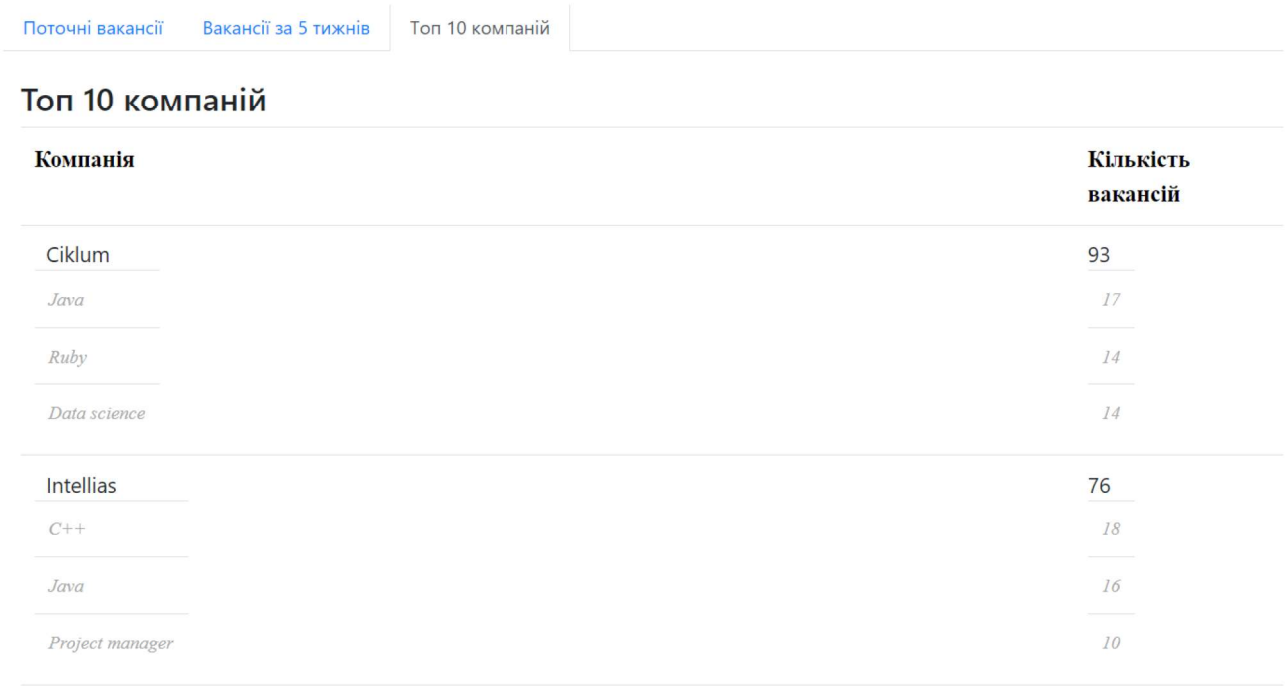

Рис 4.11 Сторінка “Top 10 компаній”

Для відображення точності прогнозування створив аналогічно до прогнозування компоненти та сервіси для доступу до API. Приклад оцінки прогнозування Рис 4.12

Точність прогнозуваня

Мова/ напрям	Дата	Поточне значення	Прогноз	Точність %	Дата	Поточне значення	Прогноз	Точність %	Дата	Поточне значення	Прогноз	Точність %
C++	2021-07-16	350	399	86%	2021-07-16	335	413	77%	2021-07-16	341	455	67%
Java script	2021-07-16	415	497	81%	2021-07-16	379	499	69%	2021-07-16	432	564	70%
Python	2021-07-16	418	470	88%	2021-07-16	415	490	82%	2021-07-16	412	536	70%

Рис4.12

Приклад сервісу для отримання даних із серверної частини

```
export class ForecastingAccuracyService {
  public API = '//localhost:8080/';

  constructor(private http: HttpClient) { }
  getPredictionAccuracyByAllLanguageAndCityID(id): Observable<any>{
    return  this.http.get(this.API + 'accuracy?languageId=&cityId=' + id);
  }

  getPredictionAccuracyByLanguage(id): Observable<any>{
    return this.http.get(this.API + 'accuracy?cityId=&languageId=' + id);
  }

  getPredictionAccuracyByCountry(): Observable<any>{
    return this.http.get(this.API + 'accuracy?cityId=&languageId=');
  }
```

Висновок до розділу

В даному розділі описано процес створення бази даних, серверної та клієнтської частин. Наведено достатню кількість прикладів програмного коду у частині опису серверної частини. В описі клієнтської частини наведено приклади зображення html сторінок та приклади програмного коду для взаємодії із серверною частиною. Наведено діаграми зв'язків між таблицями у базі даних. Для опису роботи алгоритму прогнозування, надано основні формули та частини програмного коду.

РОЗДІЛ 5. РОЗРОБЛЕННЯ СТАРТАП-ПРОЄКТУ

5.1 Опис ідеї проєкту

Основна ідея проєкту полягає у створені системи, котра зможе об'єднювати вакансії із різних джерел. Зберігати статистику вакансій для розуміння як розвивається ринок IT вакансій і які спеціальності є найбільш затребуваними. На основі статистичних даних будувати прогноз розвитку ринку IT вакансій.

Основне завдання даної системи – це економія часу при пошуку вакансій для людей, котрі визначились із напрямом своєї роботи. Кращого розуміння в який напрям рухаються українські IT компанії для людей, котрі тільки починають обирати свій фах.

Кожна людина при завершенні навчання стикається із проблемою пошуку роботи. Часто вчасно побачене оголошення може надати хорошого старту у розвитку професійної кар'єри. Але для цього потрібно проводити багато часу на таких сайтах як dou.ua, djini.com та сайтах великих компаній з метою пошуку хорошої вакансії.

Основні переваги для користувача є:

- Економія часу при пошуку вакансії
- Перегляд статистичних даних.
- Наявність кількох джерел отримання вакансій.
- Переспрямування на першоджерело вакансії для можливості користувачу відправити своє резюме.

На даний момент є декілька прямих конкурентів даної системи це Dou.ua та Djini.com

Відмінності даних систем

Таблиця 5.1

Параметр	Розроблена система	Dou.ua	Djini
Наявність статистичних даних	так	ні	ні
Побудова прогнозування	Так	ні	ні
Можливість відгукнутись на вакансію	Так(переспрямування на оригінальну вакансію)	так	так
Наявність форуму для обговорення	ні	так	ні
Зручність користування	так	так	так

Основну аудиторі. користувачів складають люди віком від 17 років. Більшість становлять випускники університетів та коледжів, які шукають роботу в IT сфері. Також це продукт буде корисний в процесі вибору технологій котрі будуть затребувані у майбутньому для студентів 1-2 курсів та випускників шкіл.

5.2 Розроблення ринкової стратегії

Правильно розроблена ринкова стратегія дозволить у майбутньому отримати прибуток із даної системи. В даному випадку ринкова стратегія буде тісно пов’язана із маркетинговою стратегією. Для того, щоб виходити на більший обсяг зацікавлення і залучення нових користувачів потрібно правильно і поступово переходити із одного кроку маркетингової стратегії на інші.

При залучені стабільної кількості користувачів наступним кроком буде співпраця із рекрутерами. Це допоможе зробити зручний інтерфейс для створення власних вакансій і надання можливості рекрутерам спілкуватись через дану систему. Ці дії дозволять залучити більше нових користувачів.

Наступним кроком буде співпраця із великими компаніями для публікування їх вакансій у наших системах. Більшість великих компаній такі, як: Epam, Softserve, DataArt часто не публікують свої вакансії на сторонніх ресурсах, а лише на власних сайтах у спеціалізованих розділах. Досягнення цієї мети буде великим кроком у розвитку системи.

5.3 Розроблення маркетингової програми

Правильно розроблена маркетингова програма дозволить швидко залучати нових користувачів і підтримувати їх стабільну кількість.

Основною аудиторією являються люди котрі шукають роботу у сфері IT. На даний момент часу кількість таких людей досить велика, проте знайшовши роботу вони на деякий час перестають користуватись подібними сайтами. Для цього потрібно підтримувати постійний потік нових користувачів.

Першочерговими завданнями є налагодження зв'язків між вищими навчальними закладами, для залучення студентів випускних курсів, котрі шукають роботу та студентів перших курсів, котрі тільки починають визначатись у якій IT сфері будуть себе реалізовувати, та знову вони в майбутньому будуть користуватись даною системо для пошуку роботи.

Створення реклами на спеціалізованих форумах для IT спеціалістів. Розміщення відгуків про роботу даної системи.

5.4 Вимоги до технічного та програмного забезпечення

Оскільки дана система буде працювати на хмарних середовищах вимоги до серверної частини мають бути такі:

База даних PostgreSQL версії 12.5 і вище.

Розмір бази даних 1 Гб, із можливістю за необхідності його збільшити.

Можливість розгортання контейнерів із підтримкою мови програмування java версії 8.

Фізичне розташування серверів у східній частині Європи. Для зменшення затримки між сигналом із клієнтської частини до серверної.

Для роботи клієнтської частини потрібне швидкісне з'єднання із мережею інтернет, щоб зменшити час відправки повідомлень між серверною і клієнтською частиною.

Підтримка JavaScript та TypeScript.

Основною вимогою до програмного забезпечення є стабільна робота системи та зменшення залежності від людського фактору.

Висновок до розділу

В даному розділі описано основну ідею створення даної системи, проаналізовано переваги системи над конкурентами, а також наведено відмінності між створеною системою та конкурентами. Описано ринкову та маркетингову стратегію. Наведено вимоги до програмного та технічного забезпечення.

Висновок

В ході дипломного проектування було досліджено способи отримання інформації у мережі Інтернет. Проаналізовано існуючі засоби для аналізу веб-сторінок. Описано декілька прикладів та наведено переваги та недоліки.

Було створено веб-орієнтовану систему прогнозування розвитку ринку IT-вакансій в Україні. Вона дозволяє користувачеві отримати вакансії із різних джерел одночасно, виконати фільтрування вакансій за такими критеріями як: місто та напрям. Користувач може переглянути статистику вакансій за напрямами на поточний момент часу та за декілька минулих тижнів. Також є можливість побачити “Топ 10 компаній” за кількістю вакансій та які напрями в них затребувані. Побачити прогноз розвитку того чи іншого напрямку по всій країні чи окремому місті або регіоні та за певним напрямом.

Згідно із прогнозом найбільший приріст буде у таких мовах програмування C++, JavaScript та Python.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Craig Walls Spring Boot in Action Manning Publications Co, 2016р., 248с
2. Джоєл Грас Data Science. Нака про дані з нуля, БХВ-Петербург, 2017р. 336с., переклад Логунов А
3. Spyros Makridakis, Steven Wheelwright, Rob Hyndman Manual of Forecasting: Methods and Applications 1997.
4. Єріна А. М. Статистичне моделювання та прогнозування: Навч. посібник. — К.: КНЕУ, 2001 р.
5. І. Г. Лук'яненко, В. М. Жук Аналіз часових рядів. Побудова ARIMA, ARCH/GARCH моделей з використанням пакета E.Views 6.0, Частина 1, Київ 2003 р.
6. Джон Д. Келлехер, Брайан Мак-Нейми, Аоифе д'Арсі Основи машинного навчання для аналітичного прогнозування: алгоритми, робочі приклади та тематичні дослідження, Київ 2010 р
7. Benjamin J Evans, James Gough, and Chris Newland Optimizing Java, 2019р. 450с.
8. https://jsoup.org/apidocs/.
9. https://material.angular.io/guides

Додаток А. Серверна частина

Клас ForecastitApplication

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

    @Scheduled(cron ="0 0 * * 6" )
    public void parse(){
        ParseServiceImpl  parseService = new ParseServiceImpl();
        parseService.parseData();
    }
}
```

Клас Vacancy

```
package com.example.forecastit.entity;

import com.example.forecastit.entity.enums.Region;
import com.example.forecastit.entity.enums.VacancyStatus;
import java.util.Date;
import javax.persistence.*;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

@Getter
@Setter
@NoArgsConstructor
@Entity
@Table(name = "vacancy")
public class Vacancy {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String position;
    @Column(columnDefinition = "text")
    private String info;
    private String source;
    private String site;

    @Enumerated(value = EnumType.STRING)
    @Column(name = "vacancy_status")
    private VacancyStatus vacancyStatus;

    @ManyToOne
    private VacancyDate date;

    @ManyToOne
    @JoinColumn(name = "company_id")
    private Company company;

    @ManyToOne
    @JoinColumn(name = "ciy_id")
    private City city;

    @ManyToOne
    @JoinColumn(name = "language_id")
    private Language language;
    @OneToOne(cascade = CascadeType.ALL)
    @JoinColumn(name = "detail_id")
```

```
    private VacancyDetail vacancyDetail;
}
```

Клас VacancyDetail

```
@Getter
@Setter
@NoArgsConstructor
@Entity
@Table(name = "vacancy_detail")
public class VacancyDetail {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "details",columnDefinition = "TEXT")
    @ElementCollection
    @OrderColumn(name = "list")
    private List<String> details = new ArrayList<>();

    @OneToOne(mappedBy = "vacancyDetail",cascade = CascadeType.ALL)
    private Vacancy vacancy;

    public void addDetails(String str){
        details.add(str);
    }
    public void addDetails(ArrayList<String> strings){
        for (String s :strings){
            details.add(s);

        }
    }
}
```

Клас Forecast

```
@Component
public class Forecast {

    public List<Long> forecast(List<Long> data) {
        List<Long> forecastList = growthIncrease(data);
        if (forecastList != null) {
            return forecastList;
        }
        forecastList = getDataBySublist(data);
        if (forecastList!=null){
            return forecastList;
        }
        else {
            return forecastExp(data);
        }
    }

    public List<Long> forecastExp(List<Long> data) {
        double a = 0.2;
        List<Long> forecastList = new ArrayList<>();

        for (int i=0;i<3;i++){
            List<Double> vurivn = this.exponentaVurivn(data);
            double s = this.avgsqrtVidhulenia(data);
            double t = 1.64;// коеф Стюденса при a0.1 і безкінечній свободі
            double deltaY = t * s * Math.sqrt((a / (2.0 - a)));


            double ynp1 = vurivn.get(vurivn.size() - 1) + deltaY;
            forecastList.add(Math.round(ynp1));
```

```
            data.add(Math.round(ynp1));
        }

        return forecastList;
    }

    public List<Double> exponentaVurivn(List<Long> data){
        //Експонентне вирівнювання

        double y1 = (1.0/5.0)*(3*data.get(0)+2*data.get(1)+data.get(2)-
data.get(3));
        double a =0.2; //  параметр згладжування Значення
        List<Double> normExp = new ArrayList<>();
        normExp.add(y1);
        for (int i=1;i<data.size();i++){
            double yi = a*data.get(i)+(1.0-a)*data.get(i-1);
            normExp.add(i,yi);
        }

        return normExp;
    }

    private Double avgsqrtVidhulenia(List<Long> data){
        // середньо квадратичне відхилення
        double sum  = 0;
        for (int i=0;i<data.size();i++){
            sum = sum+ data.get(i);
        }
        Double yavg =(1.0/data.size())* sum; //Середній рівень інтервального
ряду

        double s = 0;
        double sumS =0;
        for (int i=0;i<data.size();i++){
            sumS = sumS+Math.pow((data.get(i)-yavg),2);
        }

        s = Math.sqrt(sumS/(data.size()-1));
                return s;
    }

    private List<Long> growthIncrease(List<Long> allList){
        List<Long> subListLocal = allList.subList(allList.size()-
5,allList.size());
        Long growth =  subListLocal.get(0)-subListLocal.get(1);
        for (int i =1;i<subListLocal.size()-1;i++){
            if (growth!=(subListLocal.get(i)-subListLocal.get(i+1))||growth==0){
                return null;
            }
        }
        List<Long> predictionList = new ArrayList<>();
        growth = Math.abs(growth);
        predictionList.add(subListLocal.get(subListLocal.size()-1)+growth);
        for(int i=0;i<2;i++){
            predictionList.add(predictionList.get(i)+growth);
        }
        return predictionList;
    }

    private List<Long> getDataBySublist(List<Long> allList){
        List<Long> sublist = allList.subList(allList.size()-5,allList.size());
        List<Long> predictionList;

        int index =  Collections.indexOfSubList(allList,sublist);
        if (index!=-1){
```

```
            if ((index+5)<(allList.size()-3)){
                System.out.println(allList.size());
                System.out.println("<");
                predictionList = allList.subList(index+5,index+8);
                return predictionList;
            }else {
                return null;
            }
        }else {
            return null;
        }
    }
}
```

Клас ForecastServiceImpl

```
@Service
public class ForecastServiceImpl  implements ForecastService {

    @Autowired
    private VacancyDateRepository dateRepository;
    @Autowired
    private LanguageRepository languageRepository;
    @Autowired
    private VacancyRepository vacancyRepository;
    @Autowired
    private CityRepository cityRepository;
    @Autowired
    private Forecast forecast;

    @Override
    public List<LanguageForecastDto> predictByLanguageAndCity(Long languageId,
Long cityId) {
        return null;
    }

    @Override
    public List<LanguageForecastDto> predictByLanguage(Long languageId) {
        List<VacancyDate> dates = dateRepository.findAll();
        VacancyDate lastDate = dateRepository.getOne(dateRepository.count());
        Language language = languageRepository.getOne(languageId);
        System.out.println(language.toString());
        List<Long> counts = new ArrayList<>();
        for (VacancyDate date: dates){

counts.add(vacancyRepository.getAllVacancyByDateIDAndLanguageID(date.getId(),lan
guageId));
        }
        List<Long> forecastData =  forecast.forecast(counts);
        List<ForecastDto> forecastDtos = new ArrayList<>();
        // кількість тижнів
        int count = 1;
        for (Long l:forecastData){
            forecastDtos.add(ForecastDto.builder()
                .date(addWeek(lastDate.getDate(),count))
                .count(l)
                .build());
            count++;
        }

        List<LanguageForecastDto> languageForecastDtos = new ArrayList<>();
        languageForecastDtos.add(LanguageForecastDto.builder()
            .language(language.getLanguage())
            .forecastDtoList(forecastDtos)
            .build());
```

```
        return languageForecastDtos;
    }

    @Override
    public List<LanguageForecastDto> predictByCity(Long cityId) {
        List<VacancyDate> dates = dateRepository.findAll();
        VacancyDate lastDate = dateRepository.getOne(dateRepository.count());
        List<Language> languageList = languageRepository.findAll();
        List<Long> counts = new ArrayList<>();
        List<LanguageForecastDto> languageForecastDtos = new ArrayList<>();
        for (Language language:languageList){
            for (VacancyDate date: dates){

counts.add(vacancyRepository.getAllVacancyByDateIDAndLanguageIDAndCityID(date.ge
tId(),language.getId(),cityId));
            }
            List<Long> forecastData =  forecast.forecast(counts);
            List<ForecastDto> forecastDtos = new ArrayList<>();
            int count = 1;
            for (Long l:forecastData){
                forecastDtos.add(ForecastDto.builder()
                    .date(addWeek(lastDate.getDate(),count))
                    .count(l)
                    .build());
                count++;
            }

            languageForecastDtos.add(LanguageForecastDto.builder()
                .language(language.getLanguage())
                .forecastDtoList(forecastDtos)
                .build());
        }

        return languageForecastDtos;
    }

    @Override
    public List<LanguageForecastDto> predictByCountry() {
        List<VacancyDate> dates = dateRepository.findAll();
        VacancyDate lastDate = dateRepository.getOne(dateRepository.count());
        List<Language> languageList = languageRepository.findAll();

        List<LanguageForecastDto> languageForecastDtos = new ArrayList<>();
        for (Language language:languageList){
            List<Long> counts = new ArrayList<>();
            for (VacancyDate date: dates){

counts.add(vacancyRepository.getAllVacancyByDateIDAndLanguageID(date.getId(),lan
guage.getId()));
            }
            List<Long> forecastData =  forecast.forecast(counts);
            List<ForecastDto> forecastDtos = new ArrayList<>();
            // кількість тижнів
            int count = 1;
            for (Long l:forecastData){
                forecastDtos.add(ForecastDto.builder()
                    .date(addWeek(lastDate.getDate(),count))
                    .count(l)
                    .build());
                count++;
            }

            languageForecastDtos.add(LanguageForecastDto.builder()
                .language(language.getLanguage())
                .forecastDtoList(forecastDtos)
```

```
                .build());
        }
        return languageForecastDtos;
    }

    private String addWeek(Date date,int count){
        Calendar calendar = Calendar.getInstance();
        calendar.setTime(date);
        calendar.add(Calendar.WEEK_OF_YEAR,count);
        DateFormat simpleDateFormat = new SimpleDateFormat("yyyy-MM-dd");

        return simpleDateFormat.format(calendar.getTime());
    }

}
```

Клас ForecastController

```
@CrossOrigin(origins = "*", allowedHeaders = "*")
@RestController
public class ForecastController {

    @Autowired
    private ForecastService forecastService;
    @Autowired
    private LanguageService languageService;

    @GetMapping("/forecast")
    public List<LanguageForecastDto> forecast(@RequestParam(required = false,
value = "languageId") Long langId,
                                              @RequestParam(required = false,
value = "cityId") Long cityId)

        if (cityId!=null){
            return forecastService.predictByCity(cityId);
        }else if (langId!=null){

            return forecastService.predictByLanguage(langId);
        }else {
                        return forecastService.predictByCountry();
        }
    }

    @GetMapping("/forecast/language")
    public List<LanguageDto> getAllLanguageDto(){
        return languageService.getAllDto();
    }
}
```

Клас ForecastingAccuracyServiceImpl

```
@Service
public class ForecastingAccuracyServiceImpl implements
ForecastingAccuracyService {

    @Autowired
    private VacancyDateRepository dateRepository;
    @Autowired
    private LanguageRepository languageRepository;
    @Autowired
    private VacancyRepository vacancyRepository;
    @Autowired
    private CityRepository cityRepository;
    @Autowired
    private Forecast forecast;
```

```
    @Override
    public List<AccuracyPredictDto> accuracyPredictByLanguageAndCity(Long
languageId, Long cityId) {
        return null;
    }

    @Override
    public List<AccuracyPredictDto> accuracyPredictByLanguage(Long languageId) {

        List<VacancyDate> dates = dateRepository.findAll();
        VacancyDate lastDate = dateRepository.getOne(dateRepository.count());
        Language language = languageRepository.getOne(languageId);

        List<AccuracyPredictDto> accuracyPredictDtos = new ArrayList<>();
        List<Long> counts = new ArrayList<>();
        List<Long> actualCounts = new ArrayList<>();//3 очікуваних елемента
        for (VacancyDate date : dates) {

counts.add(vacancyRepository.getAllVacancyByDateIDAndLanguageID(date.getId(),
language.getId()));
        }
        actualCounts.addAll(counts.subList(counts.size() - 4, counts.size() -
1));
        for (int i = 1; i <= 3; i++) {
            counts.remove(counts.size() - 1);
        }

        List<Long> forecastData = forecast.forecast(counts);
        List<AccuracyPredictCountDto> accuracyPredictCountDtos = new
ArrayList<>();

        int count = 3;
        int expectedIndex = 0;

        for (Long l : forecastData) {
            Long actual = actualCounts.get(expectedIndex);
            Long diference = Math.abs(actual - l);
            double persDif = 100 - (diference * 100) / actual;
            accuracyPredictCountDtos.add(AccuracyPredictCountDto.builder()
                .date(minusWeek(lastDate.getDate(), count))
                .actual(actual)
                .expected(l)
                .accuracy(persDif)
                .build());
            expectedIndex++;
        }
        accuracyPredictDtos.add(AccuracyPredictDto.builder()
            .languageName(language.getLanguage())
            .predictCount(accuracyPredictCountDtos)
            .build());
        return accuracyPredictDtos;
    }

    @Override
    public List<AccuracyPredictDto> accuracyPredictByCity(Long cityId) {

        List<VacancyDate> dates = dateRepository.findAll();
        VacancyDate lastDate = dateRepository.getOne(dateRepository.count());

        List<Language> languageList = languageRepository.findAll();
        List<AccuracyPredictDto> accuracyPredictDtos = new ArrayList<>();
        for (Language language : languageList) {
            List<Long> counts = new ArrayList<>();
            List<Long> actualCounts = new ArrayList<>();//3 очікуваних елемента
```

```
            for (VacancyDate date : dates) {
                counts.add(vacancyRepository
                    .getAllVacancyByDateIDAndLanguageIDAndCityID(date.getId(),
language.getId(), cityId));
            }
            actualCounts.addAll(counts.subList(counts.size() - 4, counts.size()
- 1));
            for (int i = 1; i <= 3; i++) {
                counts.remove(counts.size() - 1);
            }

            List<Long> forecastData = forecast.forecast(counts);
            List<AccuracyPredictCountDto> accuracyPredictCountDtos = new
ArrayList<>();

            int count = 3;
            int expectedIndex = 0;

            for (Long l : forecastData) {
                Long actual = actualCounts.get(expectedIndex);
                Long diference = Math.abs(actual - l);
                double persDif = 100 - (diference * 100) / actual;
                accuracyPredictCountDtos.add(AccuracyPredictCountDto.builder()
                    .date(minusWeek(lastDate.getDate(), count))
                    .actual(actual)
                    .expected(l)
                    .accuracy(persDif)
                    .build());
                expectedIndex++;
            }
            accuracyPredictDtos.add(AccuracyPredictDto.builder()
                .languageName(language.getLanguage())
                .predictCount(accuracyPredictCountDtos)
                .build());
        }

        return accuracyPredictDtos;

    }

    @Override
    public List<AccuracyPredictDto> accuracyPredictByCountry() {
        List<VacancyDate> dates = dateRepository.findAll();
        VacancyDate lastDate = dateRepository.getOne(dateRepository.count());
        List<Language> languageList = languageRepository.findAll();
        List<AccuracyPredictDto> accuracyPredictDtos = new ArrayList<>();
        for (Language language : languageList) {
            List<Long> counts = new ArrayList<>();
            List<Long> actualCounts = new ArrayList<>();//3 очікуваних елемента
            for (VacancyDate date : dates) {

counts.add(vacancyRepository.getAllVacancyByDateIDAndLanguageID(date.getId(),
language.getId()));
            }
            actualCounts.addAll(counts.subList(counts.size() - 4, counts.size()
- 1));
            for (int i = 1; i <= 3; i++) {
                counts.remove(counts.size() - 1);
            }
            List<Long> forecastData = forecast.forecast(counts);
            List<AccuracyPredictCountDto> accuracyPredictCountDtos = new
ArrayList<>();

            int count = 3;
            int expectedIndex = 0;
```

```
            for (Long l : forecastData) {
                Long actual = actualCounts.get(expectedIndex);
                Long diference = Math.abs(actual - l);
                double persDif = 100 - (diference * 100) / actual;
                accuracyPredictCountDtos.add(AccuracyPredictCountDto.builder()
                    .date(minusWeek(lastDate.getDate(), count))
                    .actual(actual)
                    .expected(l)
                    .accuracy(persDif)
                    .build());
                expectedIndex++;
            }
            accuracyPredictDtos.add(AccuracyPredictDto.builder()
                .languageName(language.getLanguage())
                .predictCount(accuracyPredictCountDtos)
                .build());
        }
        return accuracyPredictDtos
    }
    private String minusWeek(Date date, int count) {
        Calendar calendar = Calendar.getInstance();
        calendar.setTime(date);
        calendar.add(Calendar.WEEK_OF_YEAR, -1 * count);
        DateFormat simpleDateFormat = new SimpleDateFormat("yyyy-MM-dd");

        return simpleDateFormat.format(calendar.getTime());
    }
}
```

Клас AccuracyController

```
@CrossOrigin(origins = "*", allowedHeaders = "*")
@RestController
public class AccuracyController {
    @Autowired
    private ForecastingAccuracyService forecastingAccuracyService;

    @GetMapping("/accuracy")
    private List<AccuracyPredictDto> getAccuracy(@RequestParam(required = false,
value = "languageId") Long langId,
                                                 @RequestParam(required = false,
value = "cityId") Long cityId) {
        if (cityId != null) {
            return forecastingAccuracyService.accuracyPredictByCity(cityId);
        } else if (langId != null) {
            //конкретна мова по місту
            return forecastingAccuracyService.accuracyPredictByLanguage(langId);
        } else {
            return forecastingAccuracyService.accuracyPredictByCountry();
        }
    }
}
```

Інтерфейс VacancyRepository

```
@Repository
public interface VacancyRepository extends JpaRepository<Vacancy, Long> {
    @Query("SELECT  v FROM  Vacancy v where v.vacancyStatus='ACTUAL'")
    List<Vacancy> getAllActual();

    @Query("SELECT  v FROM  Vacancy v where v.vacancyStatus='ACTUAL'")
    Page<Vacancy> getAllActualPageable(Pageable pageable);

    @Query("select v from Vacancy v where upper(v.language.language) like
upper(:language)   and v.vacancyStatus='ACTUAL'")
```

```
    Page<Vacancy> getAllByLanguage(@Param("language") String language, Pageable
pageable);
    @Query("select v from Vacancy v where upper(v.city.nameCity) like
upper(:city) and v.vacancyStatus='ACTUAL' ")
    Page<Vacancy> getAllByCity(@Param("city") String city, Pageable pageable);
    @Query("select v from Vacancy v where upper(v.city.nameCity) like
upper(:city) and upper(v.language.language) like upper(:language) and
v.vacancyStatus='ACTUAL' ")
    Page<Vacancy> getAllByLanguageAndCity(@Param("city") String city,
@Param("language") String language, Pageable pageable);
    @Query("select count (v.id) from Vacancy v where v.language.id = :langId and
v.vacancyStatus='ACTUAL'")
    Long getCountByLanguageID( @Param("langId") Long id);

    @Query("select count (v.id) from Vacancy v where v.language.id = :langId and
v.city.id=:cityId and v.vacancyStatus='ACTUAL'")
    Long getCountByLanguageIDAndCityID( @Param("langId") Long
id,@Param("cityId") Long cityId);

    @Query("select count (v.id) from Vacancy v where v.language.id = :langId and
v.city.region =:region and v.vacancyStatus='ACTUAL'")
    Long getCountByLanguageIDAndRegion( @Param("langId") Long
id,@Param("region") Region region);


    @Query("select count(v.id) from Vacancy v where v.date.id=:dateId and
v.language.id=:langId")
    Long getAllVacancyByDateIDAndLanguageID(@Param("dateId") Long dateID,
@Param("langId") Long langId );

    @Query("select count(v.id) from Vacancy v where v.date.id=:dateId and
v.language.id=:langId and v.city.id=:cityID")
    Long getAllVacancyByDateIDAndLanguageIDAndCityID(@Param("dateId") Long
dateID, @Param("langId") Long langId,@Param("cityID") Long cityID );

    @Query("select count(v.id) from Vacancy v where v.date.id=:dateId and
v.language.id=:langId and v.city.region=:region")
    Long getAllVacancyByDateIDAndLanguageIDAndRegion(@Param("dateId") Long
dateID, @Param("langId") Long langId,@Param("region") Region region );

    // top 10
    @Query("select  count(v.id) from Vacancy  v where v.company.id = :companyID
and v.vacancyStatus = 'ACTUAL'")
    Long getCountByCompanyID(@Param("companyID") Long companyID);

    @Query("select  count(v.id) from Vacancy  v where v.company.id = :companyID
and v.city.id=:cityID and  v.vacancyStatus = 'ACTUAL'")
    Long getCountByCompanyIDAndCityID(@Param("companyID") Long
companyID,@Param("cityID") Long cityID);

    @Query("select  count(v.id) from Vacancy  v where v.company.id = :companyID
and v.city.region = :region and v.vacancyStatus = 'ACTUAL'")
    Long getCountByCompanyIDAndRegion(@Param("companyID") Long
companyID,@Param("region") Region region);

    @Query("select  count(v.id) from Vacancy v where v.company.id = :companyId
and v.language.id = :languageId and v.vacancyStatus = 'ACTUAL'")
    Long getCountByCompanyAndLanguage(@Param("companyId") Long
companyId,@Param("languageId") Long languageID);

    @Query("select  count(v.id) from Vacancy v where v.company.id = :companyId
and v.language.id = :languageId and v.city.id = :cityId and v.vacancyStatus =
'ACTUAL'")
    Long getCountByCompanyAndLanguageAndCity(@Param("companyId") Long
companyId,@Param("languageId") Long languageID, @Param("cityId") Long cityId);
```

```
    @Query("select  count(v.id) from Vacancy v where v.company.id = :companyId
and v.language.id = :languageId and v.city.region = :region and v.vacancyStatus
= 'ACTUAL'")
    Long getCountByCompanyAndLanguageAndRegion(@Param("companyId") Long
companyId,@Param("languageId") Long languageID, @Param("region") Region region);

    @Query("select count(v.id) from Vacancy v where v.vacancyStatus = 'ACTUAL'")
    Long getCountActualVacancy();
}
```

Інтерфейс LanguageRepository

```
@Repository
public interface LanguageRepository extends JpaRepository<Language, Long> {
    @Query("select  l from Language  l where l.language like :lang")
    Language getLanguageByLanguage(@Param("lang") String lang);

    @Query("select distinct v.language from Vacancy v where v.company.id =
:companyId")
    List<Language> getLanguageByCompany(@Param("companyId") Long id);

    @Query("select distinct v.language from Vacancy v where v.company.id =
:companyId and v.city.id = :cityId and v.vacancyStatus='ACTUAL'")
    List<Language> getLanguageByCompanyAndCity(@Param("companyId") Long
id,@Param("cityId") Long cityId);

    @Query("select distinct v.language from Vacancy v where v.company.id =
:companyId and v.city.region = :region and v.vacancyStatus='ACTUAL'")
    List<Language> getLanguageByCompanyAndRegion(@Param("companyId") Long id,
@Param("region")Region region);

}
```

Інтерфейс CompanyRepository

```
@Repository
public interface CompanyRepository extends JpaRepository<Company, Long> {
    @Query("select c from Company c where c.comapnyName like :name")
    Company getCompanyByCompanyName(@Param("name") String nameCompany);

    @Query("select distinct v.company from Vacancy v where v.vacancyStatus =
'ACTUAL'")
    List<Company> findAllActual();

    @Query("select distinct v.company from Vacancy v where v.vacancyStatus
='ACTUAL' and v.city.id = :cityID")
    List<Company> getCompanyByCityId(@Param("cityID") Long cityId);
    @Query("select distinct v.company from Vacancy v where v.vacancyStatus =
'ACTUAL' and v.city.region = :region")
    List<Company> getCompanyByRegion(@Param("region")Region region);
}
```

Клас ParseData

```
@Autowired
    private CityRepository cityRepository;
    @Autowired
    private CompanyRepository companyRepository;
    @Autowired
    private LanguageRepository languageRepository;
    @Autowired
    private VacancyRepository vacancyRepository;

    public ArrayList<Vacancy> parseDou(String city, String language, String url,
VacancyDate date) throws IOException {
        ArrayList<Vacancy> vacancyArrayList = new ArrayList<>();
```

```
        try {
             Document page = Jsoup.parse(new URL(url), 50000);
             Elements elements = page.getElementsByClass("l-vacancy");
             for (Element e : elements) {
                 Vacancy vacancy = new Vacancy();
                 vacancy.setPosition(e.getElementsByClass("vt").text());
                 vacancy.setCity(cityRepository.getCityByNameCity(city));
                 String companyName = e.getElementsByClass("company").text();
                 Company company =
companyRepository.getCompanyByCompanyName(companyName);
                 if (company == null) {
                     company = new Company();
                     company.setComapnyName(companyName);
                     company = companyRepository.save(company);
                 }
                 vacancy.setCompany(company);
                 String info = e.getElementsByClass("sh-
info").text().toString();
                 System.out.println(info);
                 vacancy.setInfo(info);
                 Elements elAs = e.getElementsByClass("vt");
                 List<String> elhref = elAs.eachAttr("href");

                 vacancy.setSource(elhref.get(0));
                 vacancy.setDate(date);
                 VacancyDetail vacancyDetail = parseDouDetail(elhref.get(0));
                 vacancy.setVacancyDetail(vacancyDetail);
vacancy.setLanguage(languageRepository.getLanguageByLanguage(language));

                 vacancy.setSite("dou");
                 vacancyArrayList.add(vacancy);
                 vacancy.setVacancyStatus(VacancyStatus.ACTUAL);
                 vacancyRepository.save(vacancy);
             }
         }catch (Exception e ){
             System.out.println(e);
         }
        return vacancyArrayList;

    }
    private VacancyDetail parseDouDetail(String url) throws IOException{
        Document page = Jsoup.parse(new URL(url),50000);
        VacancyDetail vacancyDetails = new VacancyDetail();
        Elements elementsDate = page.getElementsByClass("date");
        Elements elnames = page.getElementsByClass("g-h3");
        Elements elDetails = page.getElementsByClass("text b-typo vacancy-
section");

        ArrayList<String> detailsList = new ArrayList<>();
        for (int i=0;i<elnames.size()-1;i++){
            String det = elnames.get(i).text();
             detailsList.add(det);
            Elements elementsP = elDetails.get(i).getElementsByTag("p");
            detailsList.add(elDetails.get(i).text().toString());
        }
        vacancyDetails.addDetails(detailsList);
        return vacancyDetails;
    }
    public ArrayList<Vacancy> parseDjinni(String city,String language,String
url,VacancyDate date) throws IOException{

            ArrayList<Vacancy> vacancyArrayList = new ArrayList<>();
        try {
            Document page = Jsoup.parse(new URL(url), 500000);
```

```
            Elements elements = page.getElementsByClass("list-jobs__item");
            for (Element e : elements) {
                Vacancy vacancy = new Vacancy();

vacancy.setPosition(e.getElementsByClass("profile").get(0).text());
                System.out.println(vacancy.getPosition());
                vacancy.setInfo(e.getElementsByClass("list-
jobs__description").get(0).text());
                System.out.println(vacancy.getInfo());
                vacancy.setCity(cityRepository.getCityByNameCity(city));
                String companyName = e.getElementsByClass("list-
jobs__details").get(0).text();
                System.out.println("-------------------------------");
                System.out.println(companyName);
                Company company =
companyRepository.getCompanyByCompanyName(companyName);
                if (company == null) {
                    String selectedName = this.checkCompanyName(companyName);
                    company =
companyRepository.getCompanyByCompanyName(selectedName);
                    if (company == null) {
                        company = new Company();
                        company.setComapnyName(companyName);
                        company = companyRepository.save(company);
                    }
                }
                vacancy.setCompany(company);
                System.out.println(vacancy.getCompany());
                String details = "https://djinni.co" +
e.getElementsByClass("profile").attr("href");
                System.out.println(details);
                vacancy.setSource(details);
                vacancy.setVacancyDetail(getDjinniDetails(details));

vacancy.setLanguage(languageRepository.getLanguageByLanguage(language));
                vacancy.setSite("gjinni");
                vacancyArrayList.add(vacancy);
                vacancy.setVacancyStatus(VacancyStatus.ACTUAL);
                vacancy.setDate(date);
                vacancyRepository.save(vacancy);


            }
        }catch (Exception e ){
            System.out.println(e.getMessage());
        }

        return vacancyArrayList;
    }
    public  VacancyDetail getDjinniDetails(String url) throws IOException{
        Document page = Jsoup.parse(new URL(url),500000);
        VacancyDetail details = new VacancyDetail();
        try {

            String testProfilrSection = ""+ page.getElementsByClass("profile-
page-section").get(1);

            ArrayList<String> stringArrayList  =
this.splitDjini(testProfilrSection);

            details.addDetails(stringArrayList);

        }catch (IndexOutOfBoundsException e){
            System.out.println(e.getMessage());
            return details;
```

```
        }

        return details;
    }

    public ArrayList<String> splitDjini(String string){
        String[] strArray  = string.split("<br>");
        ArrayList<String> newStringlist = new ArrayList<>();
        for (int i=0;i<strArray.length;i++){

            if(strArray[i].contains("</b>")){
                String[] temp = strArray[i].split("</b>");
                for (int j=0;j<temp.length;j++){
                    newStringlist.add(temp[i]);
                }

            }else {
                newStringlist.add(strArray[i]);
            }
        }

        ArrayList<String> listDetails= new ArrayList<>();
        for (String s :newStringlist){
            String deletestr = "<div class=\"profile-page-section\">";

            if (s.contains(deletestr)){
                s= s.replace(deletestr,"");
                listDetails.add(s);
            }else if(s.contains("</div>")){
                s= s.replace("</div>","");
                listDetails.add(s);
            }else {
                listDetails.add(s);
            }

        }

        return listDetails;
    }

    private String checkCompanyName(String name){
        List<Company> companies = companyRepository.findAll();
        for (Company c:companies){
            String companyNameDB = c.getComapnyName();
            if (name.contains(companyNameDB)){
                return companyNameDB;
            }
        }
        if (name.contains(" at ")) {
            int index1 = name.indexOf(" at ");
            String name2 = name.substring(index1+4);
            return name2.substring(0,name2.indexOf(" "));
        }else if (name.contains(" в ")){
            int index1 = name.indexOf(" в ");
            String name2 = name.substring(index1+3);
            return name2.substring(0,name2.indexOf(" "));
        }else {
            return name;
        }
    }
}
```

Клас StatisticsController

```
@CrossOrigin(origins = "*", allowedHeaders = "*")
@RestController
@RequestMapping("/statisctics")
public class StatisticsController {
    @Autowired
    private CityService cityService;
    @Autowired
    private VacancyService vacancyService;
    @Autowired
    private LanguageService languageService;

    @GetMapping("/cities")
    public List<CityVO> getAllCity(){
        return cityService.getAll();
    }
    @GetMapping("/count")
    public List<LanguageCount> getAllLanguageCount(@RequestParam( required =
false, value = "cityID") Long cityID,
                                                   @RequestParam(required =
false, value = "region") String region){
        if (cityID!=null){
            return vacancyService.getLanguageCount(cityID);
        }else if (region!=null){
            return vacancyService.getLanguageCount(Region.valueOf(region));
        }else {
            // по Україні
            return vacancyService.getLanguageCount();
        }
    }
    @GetMapping("/count-five")
    public List<LanguagesCountByDate> getAllLanguageCountByDate(@RequestParam(
required = false, value = "cityID") Long cityID,
@RequestParam(required = false, value = "region") String region){

        if (cityID!=null){
            return vacancyService.getLanguageCountByDate(cityID);
        }else if (region!=null){
            return
vacancyService.getLanguageCountByDate(Region.valueOf(region));
        }else {
            return vacancyService.getLanguageCountByDate();
        }
    }
    @GetMapping("/languages")
    public List<LanguageVO> getAllLanguage(){
        return languageService.getAll();
    }

    @GetMapping("/top-ten")
    public List<CompanyLanguageCount> topTen(
        @RequestParam( required = false,value = "cityID") Long cityId,
        @RequestParam(required = false, value = "region") String region ) {
        if (cityId != null) {
            return vacancyService.getTopTenCompany(cityId);
        } else if (region != null) {
            return vacancyService.getTopTenCompany(Region.valueOf(region));
        } else {
            return vacancyService.getTopTenCompany();
        }
    }
```

Клас VacancyController

```
@CrossOrigin(origins = "*", allowedHeaders = "*")
@RestController
```

```
@RequestMapping("/vacancy")
public class VacancyController {
    @Autowired
    private VacancyService vacancyService;

    @GetMapping("/all")
    public PageableAdvancedDto<VacancyDto> getByLang(@RequestParam("language")
String lang,
                                                     @RequestParam("city")
String city,
                                                     Pageable pageable){
        if (!city.isEmpty()&&!lang.isEmpty()){
            System.out.println("ALLLL");
            System.out.println("city"+city+"language"+lang);
            PageableAdvancedDto<VacancyDto> dto =
vacancyService.getAllVacancyByLanguageAndCity(lang,city,pageable);
            return dto;
        }else if(!city.isEmpty()){

            PageableAdvancedDto<VacancyDto> dto
=vacancyService.getAllVacancyByCity(city,pageable);
            return dto;
        }
        else if(!lang.isEmpty()){
            PageableAdvancedDto<VacancyDto> dto =
vacancyService.getAllVacancyByLanguage(lang,pageable);

            return dto;
        }
        PageableAdvancedDto<VacancyDto> dto
=vacancyService.getAllVacancyByPage(pageable);
        return dto;
    }

    @GetMapping("/{id}")
    public VacancyVO getVacancy(@PathVariable("id") Long id){
        return vacancyService.getOne(id);
    }

    @GetMapping("/get-random")
    public List<VacancyDto> getRandom(){
        return this.vacancyService.getRandomVacancy();
    }
}
```

Додаток Б. Клієнтська частина

Клас VacancyServiceService

```
export class VacancyServiceService {
 public API = '//localhost:8080/vacancy/'

  constructor(private http:HttpClient) {
   }
   getAllVacancy():Observable<any>{
     return this.http.get(this.API+"all");
   }
   getAllVacancyByID(id:String){
     return this.http.get(this.API+id);
   }
   getAllWithParams(params):Observable<any>{
    return this.http.get(this.API+"all",{params});
   }
   getAllVacancyByPage(page,city,language):Observable<any>{
    return this.http.get(this.API+"all/?page=" +
page+"&language="+language+"&city="+city);
   }
   getSixRandomVacancy(): Observable<any>{
   return this.http.get(this.API+"/get-random");
   }
```

Клас StatisticsService

```
export class StatisticsService {
 public API = '//localhost:8080/statisctics/'

  constructor(private http:HttpClient) {
   }
   getAllCity():Observable<any>{
      return this.http.get(this.API+"cities");
   }
   getAllLanguageCount():Observable<any>{
     return this.http.get(this.API+"count");
   }

   getAllLanguageCountAndCityID(id):Observable<any>{
     return this.http.get(this.API+"count?cityID="+id);
   }

   getAllLanguageCountAndRegion(region):Observable<any>{
     return this.http.get(this.API+"count?region="+region);
   }


   getAllLanguageCountByDate():Observable<any>{
     return this.http.get(this.API+"count-five");
   }

   getAllLanguageCountByDateAndCityID(id):Observable<any>{
     return this.http.get(this.API+"count-five?cityID="+id);
   }

   getAllLanguageCountByDateAndRegion(region):Observable<any>{
     return this.http.get(this.API+"count-five?region="+region);
   }

   getAllLanguages():Observable<any>{
     return this.http.get(this.API+"languages");
   }
```

```
   getTopAll():Observable<any>{
   return this.http.get(this.API+"top-ten");
   }
  getTopByCityID(id):Observable<any>{
    return this.http.get(this.API+"top-ten?cityID="+id);
  }
  getTopByRegion(region):Observable<any>{
    return this.http.get(this.API+"top-ten?region="+region);
  }
}
```

Клас ForecastingAccuracyService

```
export class ForecastingAccuracyService {
  public API = '//localhost:8080/';

  constructor(private http: HttpClient) { }
  getPredictionAccuracyByAllLanguageAndCityID(id): Observable<any>{
    return  this.http.get(this.API + 'accuracy?languageId=&cityId=' + id);
  }

  getPredictionAccuracyByLanguage(id): Observable<any>{
    return this.http.get(this.API + 'accuracy?cityId=&languageId=' + id);
  }

  getPredictionAccuracyByCountry(): Observable<any>{
    return this.http.get(this.API + 'accuracy?cityId=&languageId=');
  }
}
```

Компонснта ForccastComponcntComponcnt

```
export class ForecastComponentComponent implements OnInit {

  cities: Array<any>;
  languages: Array<any>;
  forecasts: Array<any>;

  constructor(private  statisticsService: StatisticsService, private
forecastService: ForecastService) { }

  ngOnInit(): void {
    this.statisticsService.getAllCity().subscribe(data => {
      this.cities = data;
    });
    this.forecastService.getAllLanguageDto().subscribe(data => {
      this.languages = data;
    });
    this.getPredictionByCountry();
  }
  getPredictionbyCityID(id){
this.forecastService.getPredictionByAllLanguageAndCityID(id).subscribe(data => {
        this.forecasts = data;
      });
  }
  getPredictionbyLanguageID(id){
    console.log(id);
    this.forecastService.getPredictionByLanguage(id).subscribe(data => {
      this.forecasts = data;
    });
  }
  getPredictionByCountry(){
    // усі мови по країні
    this.forecastService.getPredictionbyCountry().subscribe(data => {
      this.forecasts = data;
```

```
    });
  }

}
```

Сторінка Forecast

```
<!--<p>forecast-component works!</p>-->
<div class="row" >
  <div class="col-sm-2" ></div>
  <div class="col-sm-9" >
    <div style="margin-left: 10%;margin-top: 1%" >
      <h4 > Прогноз розивитку IT ринку на основі статистичних даних</h4>
    </div>
  </div>
  <div class="col-sm-1" ></div>
</div>
<div class="row" style="margin-top: 3%;">
  <div class="col-sm-2" style="padding-left: 2%;" >
    <nav class="navbar">
      <!-- Links -->
      <ul class="navbar-nav">
        <li class="nav-item">
          <a class="nav-link" (click)="getPredictionByCountry()">По країні</a>
        </li>
        <li class="nav-item dropdown">
          <a class="nav-link dropdown-toggle" href="#" id="navbardrop" data-
toggle="dropdown"> Місто
          </a>
          <div class="dropdown-menu">
            <div *ngFor ="let c of cities">
              <a class="dropdown-item"
(click)="getPredictionbyCityID(c.id)">{{c.cityName}}</a>
<!--              <a class="dropdown-item" (click) =
"getStatisticsbyCityID(c.id)" >{{c.cityName}}</a>-->
            </div>
          </div>
        <li class="nav-item">
          <a class="nav-link dropdown-toggle" href="#" id="navbardrop" data-
toggle="dropdown"> Мова програмування
          </a>
          <div class="dropdown-menu">
            <div *ngFor="let l of languages">
              <a (click)="getPredictionbyLanguageID(l.id)">{{l.language}}</a>
            </div>
          </div>
        </li>

      </ul>

    </nav>
  </div>
  <div class="col-sm-9">
    <table class="table table-striped" >
      <tr class="table-header">
        <td>Мова/напрям</td>
        <td>Дата</td>
        <td>Прогноз</td>
        <td>Дата</td>
        <td>Прогноз</td>
        <td>Дата</td>
        <td>Прогноз</td>
      </tr>
      <ng-container *ngFor = " let f of forecasts" >
        <tr class="table-row">
```

```
            <td>{{f.language}}</td>
            <ng-container *ngFor="let dto of f.forecastDtoList">
              <td> {{dto.date}}</td>
              <td> {{dto.count}}</td>
            </ng-container>
          </tr>
        </ng-container>

      </table>
    </div>
    <div class="col-sm-1"></div>
</div>
```

Компонента ForecastAcurracyComponent

```
export class ForecastAcurracyComponent implements OnInit {

  cities: Array<any>;
  languages: Array<any>;
  accuracy: Array<any>;

  constructor(private  statisticsService: StatisticsService, private
forecastService: ForecastService, private forecastingAccuracyService:
ForecastingAccuracyService) {
  }

  ngOnInit(): void {
    this.statisticsService.getAllCity().subscribe(data => {
      this.cities = data;
    });
    this.forecastService.getAllLanguageDto().subscribe(data => {
      this.languages = data;
    });
    this.getPredictionAccuracyByCountry();
  }

  // tslint:disable-next-line:typedef
  getPredictionAccuracyByCityID(id) {

this.forecastingAccuracyService.getPredictionAccuracyByAllLanguageAndCityID(id).
subscribe(data => {
      this.accuracy = data;
    });
  }

  // tslint:disable-next-line:typedef
  getPredictionByAccuracyLanguageID(id) {
    console.log(id);

this.forecastingAccuracyService.getPredictionAccuracyByLanguage(id).subscribe(da
ta=>{
      this.accuracy = data;
    });
  }

  // tslint:disable-next-line:typedef
  getPredictionAccuracyByCountry() {

this.forecastingAccuracyService.getPredictionAccuracyByCountry().subscribe(data
=> {
      this.accuracy = data;
    });
  }

}
```

Сторінка ForecastAcuuracy

```
<div class="row" >
  <div class="col-sm-2" ></div>
  <div class="col-sm-9" >
    <div style="margin-left: 10%;margin-top: 1%" >
      <h4 > Точність прогнозуваня </h4>
    </div>
  </div>
  <div class="col-sm-1" ></div>
</div>
<div class="row" style="margin-top: 3%;">
  <div class="col-sm-2" style="padding-left: 2%;" >
    <nav class="navbar">
      <!-- Links -->
      <ul class="navbar-nav">
        <li class="nav-item">
          <a class="nav-link" (click)="getPredictionAccuracyByCountry()">По
країні</a>
        </li>
        <li class="nav-item dropdown">
          <a class="nav-link dropdown-toggle" href="#" id="navbardrop" data-
toggle="dropdown"> Місто
          </a>
          <div class="dropdown-menu">
            <div *ngFor ="let c of cities">
              <a class="dropdown-item"
(click)="getPredictionAccuracyByCityID(c.id)">{{c.cityName}}</a>
            </div>
          </div>
        <li class="nav-item">
          <a class="nav-link dropdown-toggle" href="#" id="navbardrop" data-
toggle="dropdown"> Мова програмування
          </a>
          <div class="dropdown-menu">
            <div *ngFor="let l of languages">
              <a
(click)="getPredictionByAccuracyLanguageID(l.id)">{{l.language}}</a>
            </div>
          </div>
        </li>

      </ul>

    </nav>
  </div>
  <div class="col-sm-9">
    <table class="table table-striped" >
      <tr class="table-header">
        <td>Мова/напрям</td>
        <td>Дата</td>
        <td>Поточне значення</td>
        <td>Прогноз</td>
        <td>Точність %</td>
        <td>Дата</td>
        <td>Поточне значення</td>
        <td>Прогноз</td>
        <td>Точність %</td>
        <td>Дата</td>
        <td>Поточне значення</td>
        <td>Прогноз</td>
        <td>Точність %</td>
      </tr>
```

```
      <ng-container *ngFor = " let f of accuracy" >
        <tr class="table-row">
          <td>{{f.languageName}}</td>
          <ng-container *ngFor="let dto of f.predictCount">
            <td> {{dto.date}}</td>
            <td> {{dto.actual}}</td>
            <td>{{dto.expected}}</td>
            <td>{{dto.accuracy}}%</td>
          </ng-container>
        </tr>
      </ng-container>

    </table>
  </div>
  <div class="col-sm-1"></div>
</div>
```

Компонента StatisticsComponent

```
export class StatisticsComponent implements OnInit {

  EAST = "EAST";
  WEST = "WEST";
  NORTH = "NORTH";
  SOUTH = "SOUTH";
  CENTER = "CENTER";
  REMOTE = "REMOTE";


  cities: Array<any>;
  languageCountCurrent: Array<any>;
  languageCountCurrentByDate: Array<any>;
  languagesList: Array<any>;
  topTenList: Array<any>;

  constructor(private statiscticsService: StatisticsService) {
  }

  ngOnInit(): void {
    this.statiscticsService.getAllCity().subscribe(data => {
      this.cities = data;
    });
    // загрузка поточних даних по країні
    this.getLanguageCountCurrent();
    //дані за 5 тижнів
    this.getLanguageCountByDate();
    // топ по країні
    this.getTopTenByCountry();
  }

  getStatisticsbyCityID(id) {
    console.log("COTYID" + id);
    this.statiscticsService.getAllLanguageCountAndCityID(id).subscribe(data => {
      this.languageCountCurrent = data;
    });


this.statiscticsService.getAllLanguageCountByDateAndCityID(id).subscribe(data =>
{
      this.languageCountCurrentByDate = data;
    });
    this.statiscticsService.getTopByCityID(id).subscribe(data => {
      this.topTenList = data;
    });
```

```
    // знову зчитуємо мову
    this.statiscticsService.getAllLanguages().subscribe(data => {
      this.languagesList = data;
    });
  }

  getLanguageCountCurrent() {
    this.statiscticsService.getAllLanguageCount().subscribe(data => {
      this.languageCountCurrent = data;
    });
  }

  getLanguageCountByDate() {
    this.statiscticsService.getAllLanguageCountByDate().subscribe(data => {
      this.languageCountCurrentByDate = data;
    });
    this.statiscticsService.getAllLanguages().subscribe(data => {
      this.languagesList = data;
    });
  }

  getByRegion(region) {
    console.log(region);
    this.statiscticsService.getAllLanguageCountAndRegion(region).subscribe(data
=> {
      this.languageCountCurrent = data;
    });

this.statiscticsService.getAllLanguageCountByDateAndRegion(region).subscribe(dat
a => {
      this.languageCountCurrentByDate = data
    });
    this.statiscticsService.getAllLanguages().subscribe(data => {
      this.languagesList = data;
    });
    this.statiscticsService.getTopByRegion(region).subscribe(data => {
      this.topTenList = data;
    });
  }

  getTopTenByCountry() {
    console.log("country");
    this.statiscticsService.getTopAll().subscribe(data => {
      this.topTenList = data;
    });
  }


}
```

Компонента StatisticsComponent

```
<div class="row" style="margin-top: 3%;" xmlns="http://www.w3.org/1999/html">
 <div class="col-sm-2" style="padding-left: 5%;">
  <!--  Навігаційна панель  -->
  <!-- A vertical navbar -->
  <nav class="navbar">

    <!-- Links -->
    <ul class="navbar-nav">
     <li class="nav-item">
          <a class="nav-link" (click)="getLanguageCountCurrent()">По країні</a>
      </li>
      <li class="nav-item dropdown">
          <a class="nav-link dropdown-toggle" href="#" id="navbardrop" data-
```

```
toggle="dropdown">Місто</a>
          <div class="dropdown-menu">
           <div *ngFor ="let c of cities">
            <a class="dropdown-item" (click) = "getStatisticsbyCityID(c.id)"
>{{c.cityName}}</a>
           </div>
          </div>
       <li class="nav-item">
         <a class="nav-link dropdown-toggle" href="#" id="navbardrop" data-
toggle="dropdown"> Регіон
          </a>
          <div class="dropdown-menu">
            <a class="dropdown-item" (click) = "getByRegion(WEST)" >Західний</a>
            <a class="dropdown-item" (click) = "getByRegion(EAST)" >Східний</a>
            <a class="dropdown-item" (click) = "getByRegion(CENTER)"
>Центральний</a>
            <a class="dropdown-item" (click) = "getByRegion(NORTH)"
>Північний</a>
            <a class="dropdown-item" (click) = "getByRegion(SOUTH)"
>Південний</a>
            <a class="dropdown-item" (click) = "getByRegion(REMOTE)"
>Віддалено</a>
          </div>
       </li>

    </ul>

  </nav>
 </div>
 <div class="col-sm-9">
   <div id = "allVacContent" >
    <ul class="nav nav-tabs" role="tablist">
        <li class="nav-item">
          <a class="nav-link active" data-toggle="tab" href="#home">Поточні
вакансії</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" data-toggle="tab" href="#menu1">Вакансії за 5
тижнів</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" data-toggle="tab" href="#menu2">Топ 10
компаній</a>
        </li>
       </ul>
       <!-- Tab panes -->
       <div class="tab-content">
         <div id="home" class="container tab-pane active"><br>
           <h3>Кількість вакансій на даний момент часу</h3>
           <div style="margin-top: 3%" >
            <table class="table table-stripped">
             <tr class="table-header">
              <td *ngFor = "let lang of languageCountCurrent">
               {{lang.language}}
              </td>
             </tr>
             <tr class="table-row">
              <td *ngFor = "let lang of languageCountCurrent">
               {{lang.count}}
              </td>
             </tr>
            </table>
           </div>

        </div>
```

```
        <div id="menu1" class="container tab-pane fade"><br>
          <h3>Кількість вакнсій за останні 5 тижнів</h3>
          <div style="margin-top: 3%" >
           <table class="table table-stripped">
            <tr class="table-header">
             <td>Дата</td>
             <td *ngFor = "let lang of languagesList">
              {{lang.language}}
             </td>
            </tr>
            <tr *ngFor = "let count of languageCountCurrentByDate" class="table-
row">
             <td>{{count.date}}</td>
             <td *ngFor = "let lan of count.list">
              {{lan.count}}
             </td>
            </tr>
           </table>
          </div>
        </div>
        <div id="menu2" class="container tab-pane fade"><br>
          <h3>Топ 10 компаній </h3>
              <table class="table table-striped">
                <tr>
                  <th class="text-style"> Компанія</th>
                  <th class="text-style">Кількість вакансій       </th>
                </tr>
                <tr *ngFor="let s of topTenList ">
                  <td class="text-company"> <span>{{s.companyName}}</span>
                    <table>
                      <tr *ngFor="let lang of s.languageCounts" >
                        <td class="lang-style">{{lang.language}}</td>
                      </tr>
                    </table>
                  </td>
                  <td class="text-company"> {{s.totalVacancyCount}}
                    <table>
                      <tr *ngFor="let lang of s.languageCounts" >
                        <td class="lang-style">{{lang.count}}</td>
                      </tr>
                    </table>
                  </td>
                </tr>
              </table>

        </div>
      </div>
    </div>
  </div>
  <div class="col-sm-1" >

  </div>
</div>
```

Компонента VacancyListComponent

```
export class VacancyListComponent implements OnInit {
public technology='';
public city='';

pageabledto:any;
vacancy:Array<any>;

totalElem:any;
totalPage:any;
```

```
pages:Array<number>;
currentPage:any;

firstDis:boolean;
nextDss:boolean;
prevDis:boolean;
lastDis:boolean;



  constructor(private vacancyService:VacancyServiceService) { }

  ngOnInit(): void {
     this.getVacancyByPage();
     this.currentPage = 0;

     this.firstDis=true;
     this.nextDss=false;
     this.prevDis=false;
     this.lastDis=false;
  }

getVacancyByPage(){

this.vacancyService.getAllVacancyByPage(this.currentPage,this.city,this.technolo
gy).subscribe(data=>{
       this.vacancy = data.page;
       this.pageabledto = data;
       this.totalElem = data.totalElements; // кількість елементів на сторінці
       this.totalPage = data.totalPages;// кількість сторінок
       this.totalPage = this.totalPage-1;
       this.currentPage = data.number;
       this.getPagesNumber(data.totalPages);//
     })
}


getPagesNumber(countPage){
  this.pages =[];
  for(let i=0;i<countPage; i++){
       this.pages.push(i);
  }
}
search(){

this.vacancyService.getAllVacancyByPage(0,this.city,this.technology).subscribe(d
ata=>{
       this.vacancy = data.page;
       this.pageabledto = data;
       this.totalElem = data.totalElements; // кількість елементів на сторінці
       this.totalPage = data.totalPages;// кількість сторінок
       this.totalPage = this.totalPage-1;
       this.currentPage = data.number;
       this.getPagesNumber(data.totalPages);// номери сторінок

     });
}

firstPageFunk(){
 event.preventDefault();
  this.currentPage = 0;
  this.getVacancyByPage();
}
nextPageFunk(){
```

```
    event.preventDefault();
    this.currentPage = this.currentPage+1;
    this.getVacancyByPage();
}
prevPageFunk(){
  event.preventDefault();
  this.currentPage = this.currentPage-1;
  this.getVacancyByPage();
}
lastPageFunk(){
  event.preventDefault();
  this.currentPage = this.totalPage;
  this.getVacancyByPage();

}

}
```

Сторінка Home

```
<link rel="icon" type="image/x-icon" href="favicon.ico">
<link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.2.0/css/all.css"
      integrity="sha384-
hWVjflwFxL6sNzntih27bfxkr27PmbbK/iSvJ+a4+0owXq79v+lsFkW54bOGbiDQ"
crossorigin="anonymous"/>
<link rel="stylesheet"
href="https://pro.fontawesome.com/releases/v5.2.0/css/all.css"/>

<div class="row">
  <div class="row" style="width: 100%;" >
    <div class="col-sm-1"></div>
    <div class="col-md-10">
      <a [routerLink]="['/home']"><div class="logo">
        <img src="../assets/images/logo.png" alt="Logo">
      </div></a>
      <nav id="primary-nav" class="dropdown cf" style="margin-top: -8%">
        <ul class="dropdown menu">
          <li><a [routerLink]="['/home']">Головна</a></li>
          <li><a [routerLink]="['/vacancy-list']">Вакансії</a></li>
          <li><a [routerLink]="['/vacancy-forecast']">Прогнозування</a></li>
          <li><a [routerLink] ="['/vacancy-statistics']"
href="#">Статистика</a></li>
          <li><a [routerLink] ="['/accuracy']">Точність прогнозування</a></li>
        </ul>
      </nav><!-- / #primary-nav -->
    </div>
    <div class="col-sm-1"></div>
  </div>
</div>
<div class="row"  >
  <div class="col-sm-12"  >

    <router-outlet></router-outlet>
  </div>
</div>
<footer>
  <div class="container">
    <div class="row">
      <div class="col-md-5">
        <div class="about-veno">
          <div class="logo">
            <img src="assets/images/footer_logo.png" alt="Venue Logo">
          </div>
```

```
            <p>Найкраща система для пошуку вакансій.</p>
            <ul class="social-icons">
              <li>
                <a href="#"><i class="fab fa-instagram"></i></a>
                <a href="#"><i class="fab fa-facebook-messenger"></i></a>
                <a href="#"><i class="fab fa-linkedin-in"></i></a>
              </li>
            </ul>
          </div>
        </div>
        <div class="col-md-4">
          <div class="useful-links">
            <div class="footer-heading">
              <h4>Корисні посилання</h4>
            </div>
            <div class="row">
              <div class="col-md-6">
                <ul>
                  <li><a [routerLink]="['/home']">Головна</a></li>
                  <li><a [routerLink]="['/vacancy-list']">Вакансії</a></li>
                  <li><a [routerLink]="['/vacancy-
forecast']">Прогнозування</a></li>
                  <li><a [routerLink] ="['/vacancy-statistics']"
href="#">Статистика</a></li>

                </ul>
              </div>
            </div>
          </div>
        </div>
        <div class="col-md-3">
          <div class="contact-info">
            <div class="footer-heading">
              <h4>Звязок із нами</h4>
            </div>
            <ul>
              <li><span>Телефон:</span><a href="#">+1 333 4040 5566</a></li>
              <li><span>Email:</span><a href="#">contact@company.com</a></li>
            </ul>
          </div>
        </div>
      </div>
    </div>
  </div>
</footer>
```

Додаток В. Діаграма класів програми