Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2023 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

за другим (магістерським) рівнем вищої освіти

на тему

“ Дослідження та програмна реалізація застосування технологій семантичного веб для вирішення задач бізнес-аналітики”

Виконав здобувач вищої освіти
ІІ курсу, групи КІ22М-1
ОПП «Комп'ютерна інженерія»
спеціальності 123 «Комп'ютерна інженерія»
_____ Бек О.О
«____» _____ 2023р.

Керівник проекту
кандидат технічних наук, доцент
_____ Босько В.В.
«____» _____ 2023 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет
Факультет *Механіко-технологічний*
Кафедра *Кібербезпеки та програмного забезпечення*
Освітній ступінь *магістр*
Галузь знань *12 " Інформаційні технології"*
Спеціальність *123 " Комп'ютерна інженерія"*
Освітньо-професійна (освітньо-наукова) програма *" Комп'ютерна інженерія "*

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____Олексій СМІРНОВ
"____" _____20___року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Беку Олександру Олеговичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація застосування технологій семантичного веб для вирішення задач бізнес аналітики*

2. Керівник роботи *Босько Віктор Васильович, канд. техн. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 34-13 від 04.08.23

3. Строк подання роботи до захисту *20.12.2023 р.*

4. Мета та завдання випускної кваліфікаційної роботи: *Метою розробки є програмне дослідження та програмна реалізація застосування семантичного веб в системах BI*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.
2.Перегляд аналогічних існуючих систем.
3. Опис і обґрунтування проектних рішень.
4. Етапи програмування системи.
5. Впровадження системи в промислову експлуатацію.
6. Наукова новизна
7. Економічна ефективність розробленої програми.
8. Заходи з охорони праці та техніки безпеки.
9. Висновки.

6. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Наукова новизна	*1 аркуш*
Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*
Діаграма процесів	*1 аркуш*
Показники економічної ефективності	*1 аркуш*

6. Консультанти по роботі, із зазначенням розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Савеленко Г.В., к.т.н., доцент	09.11.2023 р.	17.11.2023 р.
Охорона праці	Оришака О.В., к.т.н., доцент	03.11.2023 р.	21.11.2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	12.10.2023 р.	
2.	Постановка задачі, оформлення ТЗ	18.10.2023 р.	
3.	Розробка моделі компонента	23.10.2023 р.	
4.	Розробка структур даних	25.10.2023 р.	
5.	Розробка алгоритмів зв’язку та відображення	32.10.2023 р.	
6.	Програмування алгоритмів	11.11.2023 р.	
7.	Розрахунок економічної ефективності	13.11.2023 р.	
8.	Розрахунки з охорони праці та техніки безпеки	16.11.2023 р.	
9.	Оформлення ПЗ	18.11.2023 р.	
10.	Попередній захист роботи	04.12.2023 р	

Дата видачі завдання
«__»_____20 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
«__»_____20 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Бек О.О. Дослідження та програмна реалізація застосування технологій семантичного веб для вирішення задач бізнес-аналітики. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній магістерській роботі розроблено програмне забезпечення, яке призначено для реалізації BI системи з технологією семантичного веб.

Метою розробки є дослідження та програмна реалізація BI системи з застосуванням семантичного веб для покращення аналізу даних і прийняття бізнес рішень.

Об'єктом дослідження є процес BI системи з семантичним вебом.

Предметом дослідження є методи реалізації BI-систем.

Методи дослідження базуються на методах теорії кодування, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація веб-застосунку засобами технології семантичного вебу та мов програмування.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися у браузері Chrome, Firefox, Safari.

Програму розроблено в середовищі Python 3.10 та JS.

Ключові слова: комп'ютерна інженерія, веб-сайт, BI, семантичний веб.

ABSTRACT

Bek O.O. Research and software implementation of the application of semantic web technologies for solving business analytics problems. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this master's work, software was developed, which is intended for the implementation of a BI system based on semantic web technology.

The goal of the development is the research and software implementation of a BI system using the semantic web to improve data analysis and business decision-making.

The object of the study is the BI process of the Semantic Web system.

The subject of the study is the implementation methods of BI systems.

Research methods are based on coding theory methods, mathematical statistics methods, and software development methods.

The result of the work is the software implementation of a web application using semantic web technology and programming languages.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

A convenient user interface has been developed. Instructions for working with software tools are provided.

The program can be used in Chrome, Firefox, and Safari browsers.

The program was developed in the Python environment + JS.

Keywords: computer engineering, website, BI, semantic web.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, СИМВОЛІВ ТА СПЕЦІАЛЬНИХ ТЕРМІНІВ

JSON - JavaScript Object Notation

MIME - Multipurpose Internet Mail Extensions

REST - Representational State Transfer

MVC - Model-View-Controller – архітектурний шаблон

BI - Business Intelligence - бізнес-аналітика

HTML - Hypertext Markup Language

ОС - операційна система

XML - Extensible Markup Language

CMS - Content Management System

ORM - Object-relational mapping

Sass - Syntactically Awesome Stylesheets

DOM - Document Object Model

EJS - Embedded JavaScript templating

RDF - Resource Description Framework

СУБД - система управління базами даних

БД – база даних

KPI - Key Performance Indicator – показник досягнення успіху

URL – Uniform Resource Locator – стандартизована адреса ресурсу

DTO – Data Transfer Object – об’єкт передачі даних

OLAP - Online Analytical Processing - технологія обробки та аналізу даних

SW - Семантичний Веб (Semantic Web)

ВСТУП

Актуальність теми. Сучасний бізнес може існувати без аналітики. Це можливість оцінити успіхи компанії, порахувати доходи та витрати, зрозуміти слабкі місця та відкрити нові горизонти для бізнесу, а отже й отримати більше прибутку. Система автоматизації BI дозволяє збирати всі необхідні дані в автоматичному режимі та структурувати їх. Цей інструмент виводить аналітику підприємства нового рівня.

Семантичний веб - це інноваційний підхід до організації та розуміння інформації в Інтернеті, що має потенціал революціонізувати сферу бізнес-аналітики. У сучасному світі, де обсяги даних непереборні, важливо мати ефективні інструменти для збору, обробки та аналізу інформації. Семантичний веб надає змогу структурувати дані, надавати їм смисловий контекст і використовувати цю інформацію для прийняття обґрунтованих бізнес-рішень.

Семантичний веб відкриває перед нами безмежні можливості у плані збору та аналізу даних, і його реалізація у бізнес-аналітиці може допомогти підвищити ефективність бізнес-процесів та приймати кращі рішення на основі об'єктивної інформації. Також дозволяє надавати смислові теги та анотації даним, що робить їх зрозумілими для комп'ютерів. Це допомагає автоматично розпізнавати та обробляти дані.

В цілому, семантичний веб спрямований на те, щоб зробити Інтернет більш зрозумілим як для машин, так і для людей, і відкрити нові можливості у сфері обробки та аналізу інформації.

Ціль BI – перетворення даних у знання, а знань – у бізнес-дії для отримання певної користі.

Мета й завдання дослідження. Метою роботи є застосування технологій семантичного веб для вирішення задач BI. В роботі розглянуто загальну архітектуру платформ для бізнес-аналізу, розроблено нову архітектуру зважаючи

на необхідність інтеграції з BI, розроблено прототип семантичної системи для задач бізнес-аналітики.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– проаналізувати перспективи інтеграції BI з сучасними технологіями;

– проаналізувати можливості сучасних BI-платформ;

– дослідження ринку BI платформ та з'ясування, які засоби вже імплементовано для спрощення сценаріїв використання;

– проектування архітектури програмного рішення;

– створення прототипу використовуючи результати попереднього аналізу;

– проведення тестування на основі реального набору даних.

– програмна реалізація застосунку для бізнес аналітики.

Об'єктом дослідження є системи аналізу бізнес-даних.

Предметом дослідження є бізнес аналіз та семантичні технології в сучасних IT та способи їх взаємодії.

Методи дослідження базуються на методах теорії кодування, методах побудови архітектури програмного забезпечення, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– розроблено концепцію побудови системи яка об'єднує технології семантичного вебу та задач бізнес аналізу;

– сформульовано основні засоби інтеграцій технологій SW;

– система може стати альтернативою сучасним BI-системам.

Практична цінність отриманих результатів полягає в створенні прототипу додатку для інтеграції технології семантичного веб в BI застосунки.

Достовірність наукових результатів підтверджена теоретичними викладенням та отриманими даними під час тестування розробленої системи.

Таким чином, виходячи з вищеперерахованого, дослідження та програмної реалізації нової архітектури для інтеграції з BI є актуальною задачею а дана робота демонструє переваги застосування нових технологій для бізнеса в якості оптимізації бізнес-процесів.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1. Призначення системи

Сьогодні Business Intelligence (BI) сформувалося в самостійний напрям індустрії інформаційних технологій (IT). BI є споживачем та сферою застосування різноманітних нових технологій. За даними TAdviser, ринок BI-систем країн СНД (ліцензії та послуги) в 2018 році подолав позначку в 1404 USD, що на 15% більше, ніж роком раніше. Темп зростання ринку сповільнився (35% в 2022 році), але все ще залишився досить значним, випереджаючим темпи зростання ринку в цілому. Це свідчить про великий інтерес до цього напряму та його потенційний вплив на інші напрями IT.

BI системами у загальному сенсі називають корпоративний софт для внутрішньої аналітики компанії. Розшифровується як Business Intelligence. Це набір технологій, що дозволяє збирати та аналізувати дані, а також надавати їх у зручній формі.

Компанія завжди використовує кілька джерел отримання даних. Але обробка вручну не дозволяє ефективно структурувати інформацію. Помилки співробітників, повільна швидкість, незрозуміла структура створюють проблеми для бізнесу та не дозволяють повністю оцінити ситуацію. BI система – це інструмент автоматичного збору даних із різних джерел, об'єднання їх у єдину систему та подання у зручному форматі.

Наприклад, у вас заявки йдуть по телефону, з сайту, з месенджерів та з мобільного додатка. Як порахувати кількість клієнтів? Як визначити середній чек? Як зрозуміти, які товари покупців більше цікавлять? Чи можна оцінити цільову аудиторію і зробити прогноз продажу наступного місяця? Під час обробки даних вручну це займе багато часу. А деякі функції зробити просто неможливо.

Ціль BI – перетворення даних у знання, а знань – у бізнес-дії для отримання певної користі. Тому доцільно проаналізувати перспективи інтеграції BI з сучасними технологіями керування розподіленими знаннями, а саме – з Semantic Web.

На жаль, промислові застосунки не завжди вчасно впроваджують наукові досягнення: в сучасних комерційних BI-застосунках технології та стандарти Semantic Web практично не використовуються, а замість цього здійснюються спроби заново розробити засоби подання знань та метаописів.

Принцип дії для користувачів досить просто: софт ставлять на комп'ютери співробітників. Далі до системи підключаються джерела отримання даних, вся інформація потрапляє до єдиного сховища та обробляється. Після цього формуються готові звіти та надаються у зручній формі.

Ключова перевага системи BI – для роботи з нею не потрібно мати спеціалізованих навичок або програміст. Вона проста в обслуговуванні, легко налаштовується та має зрозумілий інтерфейс. Кожен співробітник без проблем розбереться у її налаштуваннях. Особливо якщо програма розробляється індивідуально під окремий бізнес.

Система формує панель для роботи – дашборд. на ній розташовуються всі дані, сформовані на кшталт інформації. Дашборд можна легко налаштувати на власний розсуд. Елементи інтерактивні, надаються графіки, а інформація оновлюється у реальному часі.

Незважаючи на простоту системи BI для користувачів вона має досить складну архітектуру. Платформа обробляє безліч процесів, збирається інформація із різних джерел, система інтегрується з безліччю сервісів. У рішення BI входять такі компоненти:

Система інтеграції даних ETL. Вона дозволяє витягувати дані з різних джерел, очищати від дублікатів та іншої непотрібної інформації та поміщає до єдиного сховища.

Аналітичне сховище даних. Спеціальна база даних, куди входить вся інформація. Далі вона структурується за заданими параметрами та проводить аналітику. По суті всі логічні процеси відбуваються саме тут.

Інструменти Data Mining. Це засоби, в яких обробляються дані та проводиться аналітика, що дозволяє виявляти тренди та залежності. Наприклад, визначення середнього чека, цільової аудиторії, статистики продажів за певні проміжки часу, семантичного аналізу, що дозволяє визначити якість реклами тощо.

Інструменти формування візуального відображення даних. Те, що бачить користувач. Візуальні звіти системи BI. При цьому користувач може вибрати формат відображення даних, перелік показників, які будуть відображатися. Можна використовувати спеціальні фільтри для сортування даних.

1.2. Область застосування

Навіщо впроваджувати BI системи у бізнесі. Перше, що варто зрозуміти: система BI підлаштовується персонально під потреби бізнесу. Індивідуальна розробка дозволяє створити аналітичний інструмент, який збиратиме абсолютно різні дані. Скажімо, якщо вам навіщось необхідно знати, якого кольору очі ваших клієнтів, систему можна налаштувати на збирання цієї інформації.

У кожній компанії утворюється велика кількість даних. Ручна їхня обробка неефективна. Вона займає багато часу, схильна до ризику помилок людського фактора, деякі дані просто губляться або упускаються. Будь-який бізнес вважає прибуток, збитки, кількість продажів, трафік клієнтів, джерело покупців та інші дані. Система BI вирішує безліч завдань бізнесу, серед яких:

- структурування інформації в єдиній системі з багатьох джерел, що дозволяє визначити їх ефективність і збирати всі дані в одному місці;
- аналіз великої кількості інформації за короткий час;

- моделювання подальшого розвитку компанії, прогнозування продажів, створення нових бізнес-рішень;

- визначення слабких сторін бізнесу та можливість зрозуміти, як їх усунути;

- формування звітностей як для внутрішнього використання в компанії, так і для зовнішніх структур, на кшталт податкової, відділу закупівель, логістичних компаній або партнерів;

- систематизація інформації, з якою легко працювати, наприклад, приходять нові співробітники та швидко вникають у систему роботи підприємства, тому що всі звітності представлені у зручній та зрозумілій формі;

- формування бази знань діяльності компанії, яка дозволить покращити якість роботи за рахунок передачі даних надалі серед співробітників.

Головна мета, яку виконує система BI – надання можливості приймати рішення щодо подальшого розвитку компанії на підставі зібраних даних та з урахуванням докладної аналітики.

Технологія Semantic Web (семантичний веб) має широкий спектр застосувань у різних сферах життя та галузях. Ось деякі з найважливіших областей застосування цієї технології:

Бізнес і економіка:

– Бізнес-аналітика: Використання семантичного вебу для аналізу бізнес-даних та виявлення зв'язків між різними факторами для прийняття більш обґрунтованих стратегічних рішень.

– Управління знаннями: Ведення та обмін знань в організації за допомогою семантичних технологій для підвищення продуктивності та інноваційності.

Медицина та наука

– Медичні дані: Застосування семантичного вебу для інтеграції та аналізу медичних даних, обміну інформацією між медичними системами та поліпшення діагностики та лікування.

– Дослідження та наукові статті: Використання семантичних технологій для покращення пошуку та взаємодії зі статтями та науковими даними.

Інтернет та соціальні мережі

– Рекомендації та пошук: Використання семантичного вебу для покращення систем рекомендацій та пошуку на інтернет-платформах.

– Семантичні соціальні мережі: Розробка соціальних мереж, які дозволяють користувачам виражати та спільно використовувати свої інтереси та знання.

Освіта

– Інтелектуальні системи навчання: Використання семантичного вебу для створення інтелектуальних систем навчання, які пропонують користувачам індивідуалізовані матеріали та завдання.

– Електронні бібліотеки та освітні ресурси: Зручний доступ до електронних бібліотек та ресурсів, завдяки семантичній анотації та інтероперабельності даних.

Культурна спадщина

– Семантична анотація музейних колекцій: Збереження та доступ до культурних об'єктів та музейних колекцій з використанням семантичної анотації.

– Цифрова гуманітаристика: Використання семантичного вебу для аналізу та дослідження літератури, мистецтва та інших аспектів культурної спадщини.

Геоінформаційні системи (ГІС)

– Семантичні ГІС: Застосування семантичного вебу для інтеграції та аналізу геоданих та геопросторової інформації.

Інтернет речей (IoT)

– Управління та аналітика IoT-даних: Використання семантичного вебу для збору, обробки та аналізу даних з IoT-пристроїв, що допомагає вдосконалити управління та прийняття рішень в реальному часі.

Фінанси та банкінг

– Фінансова аналітика: Використання семантичних технологій для аналізу фінансових даних, виявлення ризиків та можливостей.

Проаналізувавши вищенаведені задачі, можна зробити висновок, що BI системи є ланкою, яка слугує для певної інтеграції функцій та технологій усіх відділів підприємства. До того ж об'єднує дані з будь-яких «точок» організації, баз даних не залежно від їх формату та каналу надання. Інформаційний потік може також виходити і за межі організаційних кордонів, обчислювальних платформ та спеціалізованих інструментів. Загальний принцип роботи систем аналізу бізнес даних наведений на рисунку 2.1.

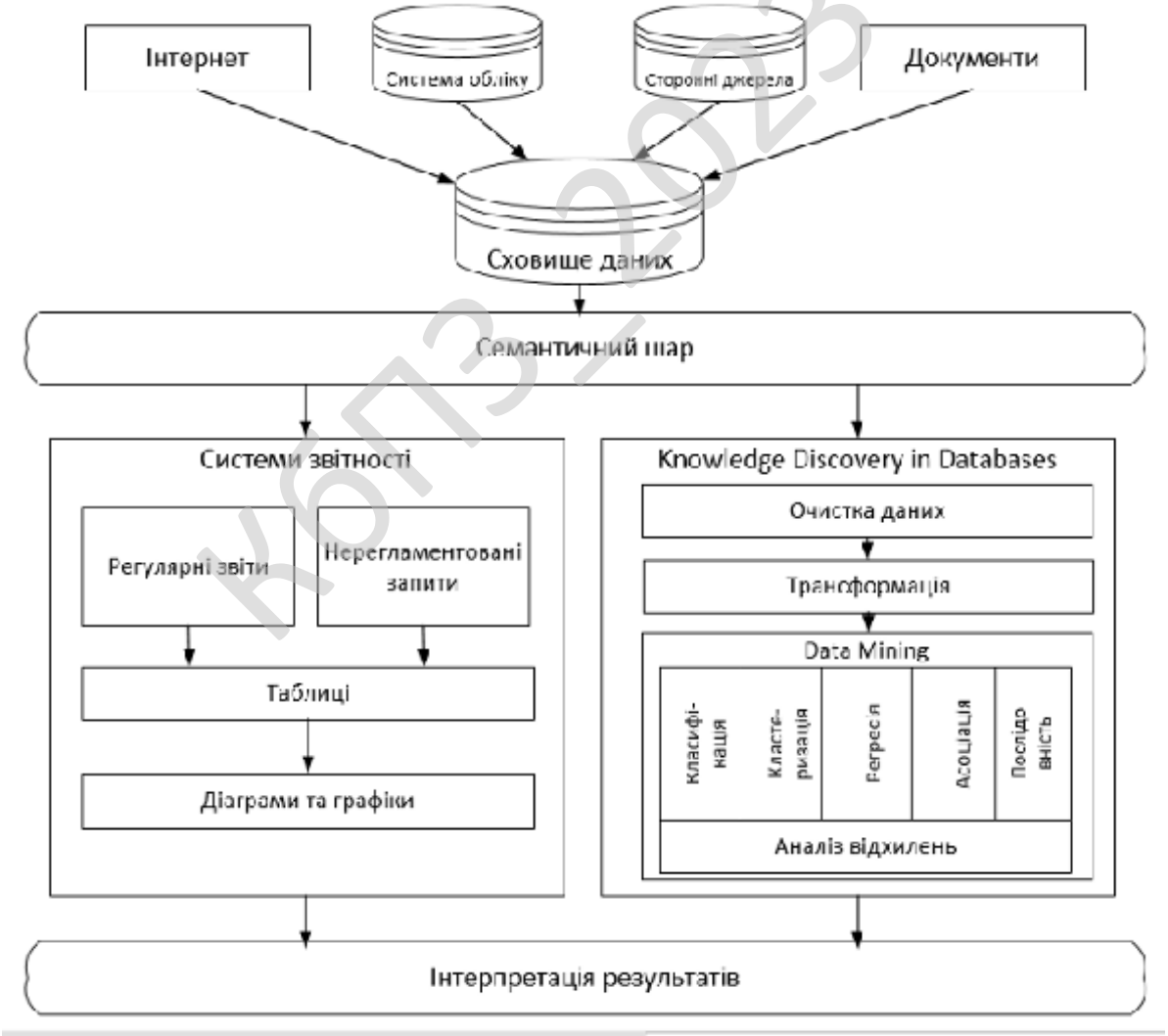

Рисунок 1.1 – Загальний принцип роботи систем аналізу бізнес-даних

Основний функціонал BI систем включає:

– збір даних з різних джерел, їх структуризація та зберігання в єдиній цілісній системі;

– аналіз великих об’ємів даних для формування і підтвердження гіпотез або розробки бізнес-рішень, спираючись на аналітику;

– моделювання можливих рішень для оцінки їх впливу на кінцеві показники діяльності і прогнозування подальшого розвитку на основі поточних даних;

– формування оперативної і стратегічної звітності, в тому числі сповіщення про відхилення показників від допустимих норм;

– збереження і систематизація знань з ціллю майбутньої передачі новим працівникам, для передавання досвіду і стабільного підвищення якості роботи.

Не дивлячись на такі масштабні можливості систем бізнес-аналізу, впровадження їх в роботу підприємств не гарантує миттєвого підвищення ефективності. Адже тут важливу роль грає людський фактор – професійність і обізнаність людей, які відповідальні за роботу з системою (зазвичай бізнесменеджери й аналітики), та як вони тлумачать надані результати. Адже інтерфейс, інтерпретація результатів та в цілому взаємодія з системою є досить складними для пересічного користувача та інтуїтивно не до кінця зрозумілими. Тому коли постала така проблема нестачі спеціалістів в області даних, дослідники та розробники почали шукати можливі шляхи до спрощення взаємодії та звернули увагу на технології обробки природних мов (NLP).

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1. Огляд існуючих систем

Семантичний веб (Semantic Web) - це концепція та набір технологій, спрямованих на поліпшення можливостей Світової павутини, роблячи веб-контент більш зрозумілим для комп'ютерів. Ідея за Семантичним вебом полягає в тому, щоб забезпечити можливість спільного використання, зв'язку та обробки даних таким чином, щоб комп'ютери могли розуміти та виконувати більш складні завдання.

Semantic Web являє собою мережу інформаційних вузлів, які пов'язані один з одним таким чином, щоб наявна інформація могла легко оброблятися комп'ютером. Його можна розглядати як ефективний спосіб представлення даних у Всесвітній павутині, або як глобально пов'язану базу даних. Даний проект пропонує реалізацію повної системи з автоматизованого створення та зберігання семантичного ядра контенту, наданого у Всесвітній павутині.

Проект Semantic Web - це спроба зібрати всі сталі ідеї і зробити так, щоб вони змогли працювати разом всередині мережі Інтернет.

Для досягнення цієї мети використовуються стандарти, які розроблені не тільки консорціумом W3C, а й іншими організаціями.

Мета проекту - дозволити взаємодіяти цим стандартам між собою, всередині децентралізованої системи, без втручання людини.

Аспекти семантичного вебу

RDF (Resource Description Framework): RDF є основною технологією Семантичного вебу. Він надає структурований спосіб представлення даних та їх зв'язків. RDF використовує триплети (суб'єкт-предикат-об'єкт) для вираження висловлювань про ресурси.

OWL (Web Ontology Language): OWL - це родина мов для представлення онтологій на Семантичному вебі. Онтології описують концепції, відносини, інформаційну модель і семантику даних.

SPARQL (SPARQL Protocol and RDF Query Language): SPARQL - це мова запитів для даних RDF. Вона дозволяє виконувати складні запити до даних, подібно до того, як SQL використовується для реляційних баз даних.

Семантична анотація. Дані на Семантичному вебі можуть бути анотовані семантичною інформацією, яка допомагає розуміти їх значення та контекст.

Лінкед-дані (Linked Data): Лінкед-дані - це підходи до розміщення та спільного використання даних відповідно до принципів Семантичного вебу. Дані розміщуються в мережі і зв'язуються за допомогою URI, створюючи величезний глобальний граф знань.

Онтології: Онтології на Семантичному вебі використовуються для опису семантики даних та створення формальних моделей знань.

Семантичні пошукові системи: Семантичні пошукові системи намагаються зрозуміти смисловий контекст запитів та допомагають користувачам знаходити більш точні та релевантні результати пошуку.

Існують різні системи та інструменти, які використовують технології Семантичного вебу.

DBpedia. DBpedia - це проект, який перетворює дані з Вікіпедії в машинно-читабельний формат RDF, створюючи величезний набір лінкед-даних.

Schema.org. Схема.org - це спільна ініціатива від Google, Microsoft, Yahoo та інших, яка надає специфікації мікроданих для веб-сторінок, щоб полегшити розуміння контексту та інтеграцію з пошуковими системами.

Protege - це інструмент для створення та редагування онтологій на Семантичному вебі.

TopBraid Composer - Це інтегрована розробка і моделювання інструментів для Семантичного вебу та онтологій.

Stardog - це гібридна Семантична база даних, яка поєднує RDF з реляційною моделлю даних.

Системи бізнес-аналітики(BI)

На даний час існує безліч систем бізнес-аналітики, які надають різноманітні функціональні можливості для обробки даних та прийняття обґрунтованих бізнес-рішень.

Tableau: Tableau - це одна з найпопулярніших систем візуалізації даних, яка дозволяє аналізувати та відображати дані у вигляді інтерактивних графіків та таблиць. Вона підтримує з'єднання з різними джерелами даних та має декілька версій для різних потреб користувачів.

Power BI: Power BI - інструмент для аналізу та візуалізації даних від Microsoft. Він має потужні можливості для підключення до різних джерел даних, створення звітів та панелей інструментів та автоматизації завдань аналітики.

QlikView і Qlik Sense: Обидві ці системи надають можливості для аналізу та візуалізації даних. QlikView - це попередня версія, а Qlik Sense - більш сучасна система, яка надає інтуїтивний інтерфейс для створення звітів та дашбордів.

IBM Cognos: Це інтегрована платформа бізнес-аналітики, яка дозволяє створювати, розповсюджувати та аналізувати звіти та дані в організації.

SAS Business Intelligence: SAS BI - це рішення бізнес-аналітики, яке надає інструменти для аналізу даних, статистики, моделювання та візуалізації.

MicroStrategy: MicroStrategy - це платформа бізнес-аналітики з рядом інструментів для візуалізації даних, створення звітів і панелей інструментів, а також інтеграції з різними джерелами даних.

Sisense: Sisense - це система для об'єднання даних з різних джерел та створення динамічних звітів та дашбордів.

Domo: Domo - це хмарна платформа бізнес-аналітики, яка надає інструменти для агрегації та візуалізації даних, а також спрощує спільну роботу та прийняття рішень в команді.

Looker: Looker - це платформа бізнес-аналітики, яка спеціалізується на аналізі даних в реальному часі та надає інструменти для розробки звітів та дашбордів.

Google Data Studio: - це безкоштовний інструмент для створення звітів та візуалізації даних, який інтегрується з іншими продуктами Google і різними джерелами даних.

Ці системи бізнес-аналітики мають різні особливості, вартість та функціональні можливості, і вибір конкретного рішення залежить від потреб вашої організації та галузі. Також важливо враховувати тенденції та нові розробки у світі бізнес-аналітики для вибору найбільш підходящого рішення для вашого бізнесу.

З точки зору архітектури Semantic Web можна розглядати, як три яруси (рисунок 2.1): базис, який складається з унікальної глобальної ідентифікації ресурсу, метаданих для декларування фактів про ресурси, і спільної мови для вираження метаданих і знань, що реалізовані за допомогою онтологій, для загальнодоступного розуміння і загального словника метаданих, і правил для додавання нових метаданих та знань; базовий сервіс, наприклад, логічний висновок і запити до метаданих, і онтологія, роз'яснення таких висновків, управління довірою, агенти, пошукові системи, онтології; сервіси додатків, наприклад сервіс агентства подорожей.

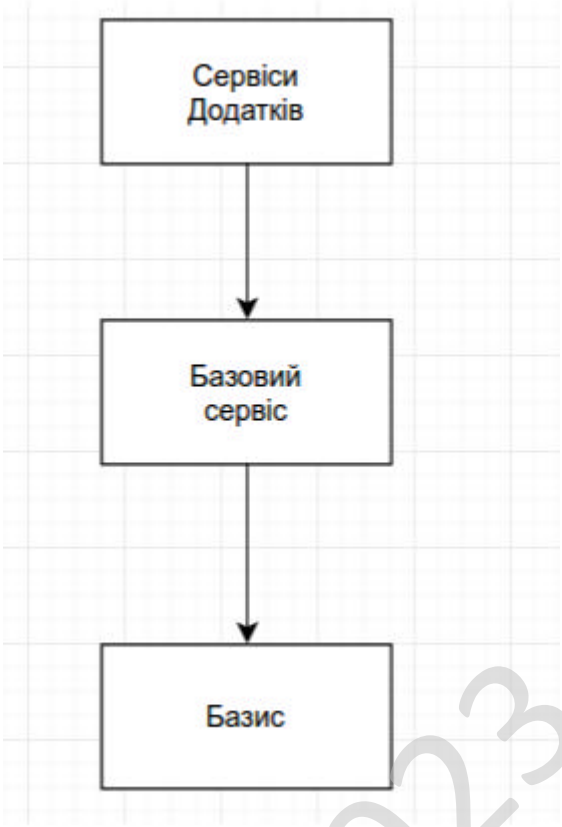

Рисунок 2.1. Три яруси мережі Semantic Web

Базова модель Semantic Web (пиріг Тіма) показана на рисунку 2.2

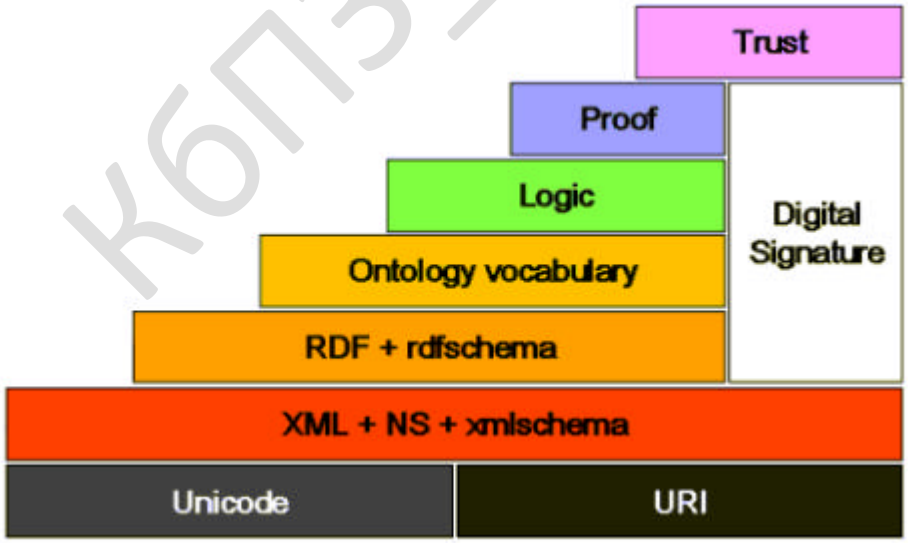

Рисунок 2.2. Базова модель Semantic Web

Порівняння застосунків Business Intelligence

Відповідно до досліджень провідних ринкових аналітиків, таких як Forrester і Gartner, розробники програмного забезпечення BI можуть бути впорядковані за допомогою “магічного квадрата”.

Лідери – постачальники з широким функціоналом можливостей платформ BI (Cognos, Business Objects, Microsoft, Oracle, MicroStrategy, SAS); претенденти на лідерство (SAP, Information Builders) пропонують платформи із широкою функціональністю, але в них є обмеження в частині програмного чи оточення предметних областей або каналів збуту; провидці (QlikTech, Tibko Spotfire) мають гарне представлення про платформу BI, відрізняються відкритістю і гнучкістю архітектури і пропонують добре розвитого функціонала у своїх цільових областях, однак, діапазон пропонованих функцій не досить широкий; нишеві гравці (Actuate, arcplan, Board International, Panorama Software) – компанії, що досягли успіху тільки у конкретній області.

Розглянемо більш детально характеристики компаній-виробників BI-додатків і їхньої продукції. SAP Business Objects пропонує широкий набір апробованих закінчених рішень BI, включаючи продукти з інтеграції та якості даних і аналіз тексту.

Всі продукти SAP Business Objects переводяться на загальну платформу для підвищення безпеки і поліпшення адміністрування, інтеграція різних продуктів, забезпечення більш зручного обміну метаданими і переходу від одного продукту до іншого.

IBM Cognos [7] – комплекс інтегрованих програмних продуктів для керування ефективністю діяльності підприємства й управлінського обліку (Corporate Performance Management, CPM). Це одне з найсучасніших і масштабованих середовищ BI. Cognos базується на єдиній платформі J2EE і надає єдиний інтерфейс для більшості продуктів.

SAS – постачальник повного комплексу рішень BI та аналітичних інструментів [8]. SAS забезпечує інтеграцію даних, аналіз тексту, інструментарій

для статистичного аналізу і проактивне моделювання, а також маркетинговий аналіз і керування ризиками.

Oracle реалізувала свої ідеї щодо BI у Enterprise Edition Plus [9], що містить сервер аналітики з механізмом ROLAP (Relational OLAP) і технологією інтеграції корпоративних даних (EII). MicroStrategy – платформа, яка охоплює весь комплекс можливостей BI, з власним механізмом ROLAP, що ефективно оптимізує різні моделі даних. Долі цих платформ у загальній кількості зображено на рисунку 2.3.

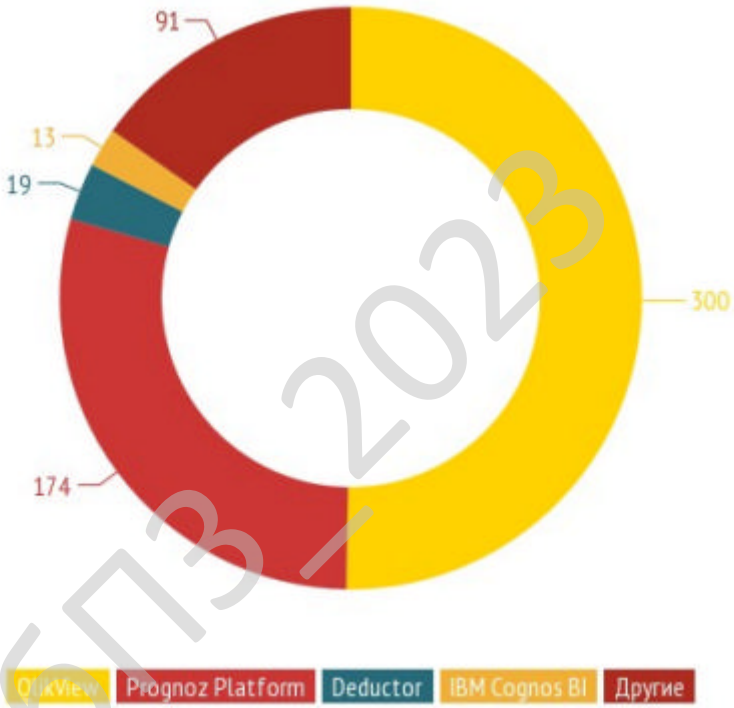

Рисунок 2.3 Долі BI-платформ в загальній кількості проектів

Більш наглядне порівняння можливостей сучасних BI-платформ наведено в таблиці 2.1.

Таблиця 2.1 - Порівняльна таблиця засобів бізнес-аналітики

Назва платформи	Засоби розробки	Сповіщення	Колективна робота	Звіти	Dashboard (індикатори)	Власна мова запитів	Веб та мобільні додатки	OLAP	Візуальний аналіз	Аналіз через пошук	Прогноз та DataMining
Birst	V	V	V	V	V	V	V	V	V	X	V
IBM Cognos	V	V	V	V	V	V	V	V	V	V	V
IBM SPSS Decision	V	+/-	V	X	X	X	X	X	X	X	V
Jaspersoft	V	+/-	V	V	V	V	V	V	V	X	X
Microstrategy Express	V	+/-	V	V	V	V	V	V	V	X	X
QlikView	V	V	V	V	V	V	V	V	V	V	+/-
SAP Lumira	X	X	+/-	X	X	X	V	X	V	X	X
SAS	X	V	V	V	V	V	V	V	V	X	V
Tableau	+/-	V	V	V	V	V	V	V	V	X	+/-
Tibco Personal	X	X	X	V	V	V	V	V	V	X	V

2.2 Обґрунтування вибору методів розробки

XML і RDF - сучасні Internet-стандарти, які служать для забезпечення семантичної інтероперабельності в Web. При цьому XML піднімає питання, пов'язані тільки зі структурою документів. RDF більше пристосований для забезпечення семантичної інтероперабельності, оскільки пропонує модель даних, яку можна розширити таким чином, щоб вона охоплювала більш досконалі методики подання онтології.

Загальна схема опису ресурсів RDF

Схема опису ресурсів у форматі RDF (Resource Description Framework) базується на використанні триплетів, що складаються з суб'єкта, предиката та об'єкта. Ця модель дозволяє представляти дані у вигляді графа знань.

Суб'єкт (Subject): Суб'єкт - це об'єкт або ресурс, який описується у триплеті. Це може бути будь-що, від осіб та місць до концепцій та подій. Суб'єкт представляється за допомогою URI (Uniform Resource Identifier), ідентифікатора, який унікально ідентифікує ресурс.

Предикат (Predicate): Предикат - це відношення або властивість, яка пов'язує суб'єкт з об'єктом. Предикат також представляється за допомогою URI і вказує на природу відношення між суб'єктом та об'єктом.

Об'єкт (Object): Об'єкт - це значення або ресурс, з яким суб'єкт пов'язаний через предикат. Об'єкт також може бути іншим суб'єктом, що розширює можливості моделі RDF для створення складних зв'язків.

Наприклад, ось триплет, який описує факт, що "John Doe є автором книги з назвою 'Semantic Web'":

– суб'єкт (Subject): John Doe (URI або інше ідентифікаційне значення);

– предикат (Predicate): є автором (URI або інше ідентифікаційне значення);

– об'єкт (Object): 'Semantic Web' (літерал, опис назви книги).

Такі триплети можуть бути об'єднані для створення більш складних графів знань, де різні ресурси пов'язані між собою за допомогою різних предикатів і властивостей.

RDF дозволяє представляти дані з різних джерел і об'єднувати їх для створення семантично багатих графів даних.

Метадані (metadata) - це дані, які надають інформацію про інші дані. Метадані допомагають зрозуміти, описати та керувати іншими даними, і вони грають важливу роль в організації та розумінні інформації. Ось кілька ключових аспектів метаданих:

Опис даних. Метадані містять інформацію про характеристики даних, такі як назва, автор, дата створення, опис вмісту, ключові слова тощо. Ця інформація допомагає користувачам та системам зрозуміти, що саме представляють собою дані.

Класифікація та категоризація. Метадані можуть включати класифікаційні та категоризаційні схеми, що допомагають організовувати дані за категоріями, тегами, темами або іншими параметрами. Наприклад, бібліотеки використовують метадані для класифікації книг за жанром або темою.

Історія та версіонність. Метадані можуть включати інформацію про історію даних, зміни та версії. Це корисно для відстеження та керування розвитком та змінами в даних.

Авторство та права. Метадані можуть включати інформацію про авторів даних, власників авторських прав, ліцензії та правила використання даних. Це важливо для забезпечення відповідності авторським правам та правилам доступу.

Місцезнаходження та ідентифікація: Метадані можуть включати інформацію про місцезнаходження даних, таку як URL, URI або інші ідентифікатори, що допомагають знайти і звертатися до них.

Формати та структура. Метадані можуть надавати інформацію про формати та структуру даних, наприклад, які поля містяться в базі даних, або як виглядає структура XML-документа.

Семантика та взаємодія. У випадках, коли метадані містять семантичну інформацію, вони допомагають системам розуміти смисл та взаємозв'язки між даними. Це особливо важливо у семантичному вебі та семантичних мережах.

Метадані використовуються в багатьох сферах, включаючи бібліотечну справу, науковий дослід, архівування даних, управління контентом, електронний комерцію, інтернет-пошук, галузі зв'язку та інші. Вони допомагають забезпечити організований та ефективний доступ до інформації та дозволяють системам краще розуміти контекст та семантику даних.

Розширювана мова розмітки (XML) - це текстовий формат, який використовується для представлення та обміну даними в структурованому форматі. XML дозволяє створювати власні теги та правила для опису даних, що робить його дуже гнучким та універсальним інструментом для обміну інформацією між різними системами та платформами.

XML

Синтаксис тегів: XML використовує пари відкриваючих та закриваючих тегів для визначення структури даних.

Вкладеність: XML дозволяє вкладати один елемент в інший, створюючи ієрархічну структуру даних.

Атрибути: Внутрішні теги можуть містити атрибути, які надають додаткову інформацію про елемент.

Розширеність: Один з основних принципів XML - це можливість визначати власні елементи та структури даних, створюючи власні схеми і онтології.

Міжплатформенність: XML є незалежним від платформи та мови програмування форматом, що дозволяє обмінюватися даними між різними системами.

Зручність для обробки: XML-документи легко оброблюються за допомогою програм та бібліотек для парсингу та створення.

Документованість: XML-документи можуть бути добре документовані, що полегшує розуміння їх структури та змісту.

XML знаходить застосування у багатьох галузях, таких як веб-розробка (для передачі та збереження даних на веб-сайтах), обмін даними між різними програмами та системами, конфігурація та налаштування програмного забезпечення, а також у різних стандартах та протоколах, таких як RSS, SOAP, XHTML, і багатьох інших.

RDF Schema (RDFS) - це розширення мови Resource Description Framework (RDF), призначене для створення онтологій та визначення структури та семантики даних у форматі RDF. RDF Schema є ключовою компонентою Семантичного Вебу і дозволяє створювати контрольовані словники та таксономії для опису ресурсів і їх відносин.

Основні поняття та функції RDF Schema включають:

- Класи та екземпляри: RDF Schema дозволяє визначати класи (типи) ресурсів і створювати екземпляри (екземпляри цих класів). Наприклад, можна визначити клас "Людина" і створити екземпляр "John Doe", який належить до цього класу.

- Властивості та предикати: RDF Schema дозволяє визначати властивості, або предикати, які вказують на відносини між ресурсами. Наприклад, властивість "має автора" може вказувати на зв'язок між книгою і її автором.

Ієрархія класів: Можна встановлювати ієрархічні відносини між класами. Наприклад, клас "Собака" може бути підкласом класу "Тварина".

Домени та області значень: RDF Schema дозволяє вказати, які властивості можуть бути застосовані до яких класів (домени) і які значення можуть мати ці властивості (області значень).

Підтримка для іншого RDF: RDF Schema може використовуватися разом з іншими RDF-даними та онтологіями для створення більш складних семантичних моделей.

Легковага онтологія: RDF Schema не є так потужним як OWL (Web Ontology Language), але він надає базові можливості для створення простих семантичних моделей.

RDF Schema допомагає впорядковувати та формалізувати дані у форматі RDF, роблячи їх більш зрозумілими для комп'ютерів і сприяючи розвитку семантичного пошуку, інтеграції даних та обробці знань.

Важливою особливістю стандарту RDF, який лежить в основі XML, є розширюваність. На RDF можна задати структуру опису джерела, використовуючи і розширюючи вбудовані поняття RDF-схем, такі як класи, властивості, типи, колекції.

Онтології

Онтології, в загальному вигляді, визначаються, як спільно використовувані формальні концепції конкретних предметних областей, вони дають загальне уявлення про поняття, інформацією, з яких, можуть обмінюватися люди та програми. Вони дозволяють скласти в концепцію домен фіксуванням сутностей і зв'язків у домені. Вказівка, в яких зв'язках бере участь

сутність, частково дозволяє зрозуміти і її значення, оскільки це надає можливість бачити, де дана сутність входить у відносини з іншим доменом.

Онтологія - це формальна специфікація концепцій, об'єктів та відносин у певній області знань. В інших словах, це структурована модель, яка визначає семантику понять та їх взаємозв'язки у певній галузі або предметній області.

Онтології використовуються для опису та формалізації знань у комп'ютерних системах, особливо в контексті Семантичного Вебу. Онтології допомагають розуміти семантику даних та сприяють автоматизованому розумінню контексту та взаємодії між даними. Основні характеристики онтологій включають:

Концепції (Класи): Онтології визначають концепції або класи, які представляють собою сутності або категорії в предметній області. Наприклад, "Людина", "Автомобіль", "Компанія" можуть бути концепціями в онтології.

Відносини (Предикати): Онтології визначають відносини або предикати, які вказують на зв'язки між концепціями. Наприклад, "має автора", "розташований в", "належить до" - це приклади відносин.

Аксіоми та правила. Онтології можуть включати аксіоми і правила, які формалізують специфікації та умови для даних та їх взаємодії. Це допомагає встановлювати логічні обмеження на дані та забезпечувати їх консистентність.

Ієрархія класів. Онтології можуть створювати ієрархічні структури класів, де підкласи успадковують властивості та характеристики батьківських класів.

Інстанції. Інстанції - це конкретні об'єкти або сутності, які належать до певних класів у онтології.

Онтології використовуються для розв'язання проблеми розуміння інформації між різними системами, особливо в сферах як штучного інтелекту, семантичного пошуку, обробки природної мови, автоматичної класифікації даних та інших областях. Їх важливість зростає в контексті розвитку Семантичного Вебу та розуміння семантики даних в інтернеті.

Мови запитів до RDF сховищ

Говорячи про мови запитів, фактично мова йде про інтеграції різних мов (інформаційно- пошукових, баз даних, маніпулювання даними, обміну даними і т.ін.) в єдину мову запитів Web. При цьому всі фахівці об’єднуються в думці, що це має бути декларативна мова, побудована на моделі не повністю структурованих даних (semistructured).

Для запитів до RDF сховищ (RDF stores) часто використовуються мови запитів, які дозволяють отримувати дані з RDF-графів, проводити пошук, агрегацію та аналіз семантичних даних. Ось декілька основних мов запитів, які використовуються для роботи з RDF сховищами:

- SPARQL (SPARQL Protocol and RDF Query Language): SPARQL є стандартом для запитів до RDF-даних. Він надає розширену можливість пошуку, фільтрації та об'єднання даних у RDF-графах. SPARQL дозволяє виконувати складні запити, отримувати дані у форматі таблиць та графів, і взаємодіяти зі семантичними даними;
- RDQL (RDF Data Query Language): RDQL - це мова запитів, яка була однією з перших мов для роботи з RDF-даними. Вона схожа на SPARQL, але менш потужна і менш розповсюджена;
- RQL (RDF Query Language): RQL - це ще одна мова запитів, спеціально розроблена для роботи з RDF-графами. Вона надає можливість створювати запити для отримання даних з RDF-джерел, але менше відома, ніж SPARQL;
- GRDDL (Gleaning Resource Descriptions from Dialects of Languages): GRDDL - це мова та методологія, яка дозволяє видобувати RDF-дані з веб-сторінок та інших документів, які мають властивості для автоматичного перетворення в RDF;
- Cypher: Хоча Cypher більш відомий своєю використанням в графових базах даних, він також може використовуватися для роботи з RDF-

графами, особливо в ситуаціях, коли графові дані семантичні або включають в себе RDF-тріплети.

SQL/SPARQL об'єднані запити. В деяких системах можна виконувати об'єднані запити, які комбінують SQL (Structured Query Language) та SPARQL для одночасної роботи з реляційними та семантичними даними.

Обираючи мову запитів для роботи з RDF сховищами, важливо враховувати специфікації та функціональність конкретної системи, з якою ви працюєте, а також складність та типи запитів, які вам потрібно виконувати. SPARQL є найбільш розповсюдженою та потужною мовою для цільової роботи з RDF-даними, але в інших випадках інші мови також можуть бути корисними.

Принцип "логічного висновку"

Inference є важливим поняттям в області штучного інтелекту, особливо в семантичному вебі та семантичних технологіях. Цей принцип відноситься до здатності комп'ютерних систем або інтелектуальних агентів до виконання логічних резонувань на основі знань, які вони мають.

Основні аспекти принципу "логічного висновку" включають:

- Знання та факти. Система має доступ до бази знань, де зберігаються факти та інформація про об'єкти та їх взаємозв'язки. Ці факти можуть бути представлені у форматі, зрозумілому для системи, такому як RDF-тріплети в семантичному вебі.

- Логічні правила. Система також може мати набір логічних правил або обчислювальних алгоритмів, які визначають, які висновки можна зробити на основі знань та фактів. Ці правила можуть бути виражені в формі логічних виразів, правил або запитів.

- Логічний резонанс. Система використовує свої знання та логічні правила для виконання резонансу та висновків. Вона може автоматично визначати нові факти, робити логічні висновки, перевіряти співвідношення між даними та виконувати запити.

Керування знаннями: Система може також використовувати методи для динамічного керування своїми знаннями, додаванням, оновленням або видаленням фактів та правил.

Принцип "логічного висновку" дозволяє створювати інтелектуальні системи, які можуть розуміти та аналізувати дані, робити висновки на основі цих даних та виконувати складні завдання розуміння інформації. Цей принцип застосовується в багатьох галузях, включаючи семантичний аналіз тексту, автоматичну обробку мови, експертні системи, семантичний пошук і багато інших.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на магістерську роботу, реалізації підлягає програмне забезпечення, яке продемонструє роботу BI-систем.

В процесі розробки магістерської роботи необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран повідомлень про некоректні дії користувача та нестандартні ситуації;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи з охорони праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

Бізнес-аналітика, також відома як Business Intelligence (або BI), це обширний термін, що охоплює різноманітні програмні продукти і інструменти, розроблені для аналізу первинних даних в організаціях. Першим, хто використовував термін "Business Intelligence," був американський вчений і співробітник IBM Ханс Петер Лун у 1958 році. У той час Лун працював у галузі інформаційних наук, коли ще інформатика була на етапі розвитку, і використовувалися електромеханічні табулятори.

BI є орієнтованим на користувача процесом, який надає можливість доступу до інформації, її аналізу, формування інтуїтивного розуміння та допомагає приймати кращі рішення. Сьогодні BI включає в себе інструменти для аналізу даних, створення звітів і запитів, які сприяють обробці великих обсягів інформації.

В основі технології BI лежить організація доступу користувачів до даних, а також аналіз структурованих кількісних даних та інформації про бізнес. BI відіграє важливу роль у зборі різноманітної інформації про досліджуваний предмет. Термін "Business Intelligence" не має точного відповідника в українській та російській мовах, і інколи використовуються не зовсім коректні переклади, такі як "бізнес-інтелект" або "бізнес-аналітика". Можливо, варто розглянути переклад "Business Intelligence" як "засоби логічного аналізу, спрямовані на бізнес-застосування."

Бізнес-аналітика як діяльність включає в себе кілька пов'язаних процесів, включаючи інтелектуальний аналіз даних (data mining), аналітичну обробку в реальному часі (online analytical processing), отримання інформації з баз даних (querying) та створення звітів (reporting).

BI не лише технологічна платформа, але і засіб для ефективного використання ресурсів підприємства, включаючи інформацію та персонал. Основні можливості BI включають інтеграцію, доставку інформації та аналіз. Компанії використовують BI для прийняття обґрунтованих рішень, оптимізації витрат та виявлення нових можливостей для розвитку бізнесу.

З чого має розпочатися впровадження BI

Впровадження Business Intelligence (BI) є важливим процесом для покращення аналізу даних і прийняття кращих бізнес-рішень. Для успішного впровадження BI рекомендується розпочати з таких етапів:

- Аналіз потреби в BI. Першим кроком є визначення, чому ваша організація потребує BI. Ретельно проаналізуйте потреби в аналізі даних, прийнятті рішень та звітності. Визначте конкретні завдання, які BI має вирішити.
- Вибір правильної BI-платформи. Оберіть BI-платформу або інструмент, який найкраще відповідає потребам вашої організації. Враховуйте функціональність, масштабованість, сумісність з існуючими системами та вартість.
- Формування команди BI. Створіть команду, в якій є аналітики даних, IT-фахівці і представники бізнесу. Ця команда буде відповідальною за розробку, впровадження та підтримку BI-системи.
- Збір та очищення даних. Забезпечте доступ до всіх необхідних даних для BI-системи. Це може включати в себе інтеграцію даних з різних джерел, очищення даних від помилок та дублікатів, а також створення єдиного джерела правдивої інформації.
- Розробка BI-звітів та панелей. Розробіть звіти, аналітичні панелі та інші інструменти BI, які відповідають потребам користувачів. Вони повинні бути легкими у використанні та забезпечувати доступ до необхідної інформації.
- Навчання та підтримка користувачів. Забезпечте навчання користувачів щодо використання BI-інструментів. Також важливо створити механізми підтримки інтерактивного обміну досвідом між користувачами.

- Тестування і впровадження. Проведіть тестування BI-системи перед її повним впровадженням. Поступово впроваджуйте її для різних підрозділів або груп користувачів.
- Оцінка та оптимізація Постійно оцінюйте роботу BI-системи, збирайте відгуки від користувачів і вдосконалюйте її функціональність для досягнення найкращих результатів.
- Розвиток стратегії BI. Постійно вдосконалюйте стратегію BI відповідно до потреб і змін в бізнес-середовищі.
- Впровадження BI - це постійний процес, який допомагає організації залишатися конкурентоспроможною та приймати обґрунтовані рішення на основі даних.

3.1 Опис функціонування системи

Сьогоднішні засоби бізнес-аналітики (BI, OLAP) в основному сконцентровані на кількісному, статистичному аналізі бізнес-даних. Більшість подібних систем використовують жорстко задані в структурі бази даних набори параметрів і показників аналізу (для їх конфігурації потрібні дорогі фахівці).

За допомогою таких засобів складно описувати і аналізувати причинно-наслідкові зв'язки, важко формалізуються і різноманітні взаємовідносини об'єктів (наприклад, ступінь впливу тих чи інших факторів на результати певних процесів).

Семантична аналітика, побудована на принципі "відкритого світу", надає нові можливості вирішення подібних завдань. А в поєднанні з принципами нечіткої логіки, побудови симулятивних ("живих") моделей, її можливості стають практично безмежними.

Найпростішим засобом ілюстрації принципових відмінностей семантичної аналітики від інших засобів аналізу є інструмент Facebook Graph Search. Цей пошуковий інтерфейс, вбудований в соціальну мережу, може відповісти на запитання на кшталт "Які заклади подобаються моїм друзям?", "У

яких містах бували мої родичі?". Зрозуміло, що жодна система повнотекстового пошуку (якими є сучасні пошукові машини) не зможе дати відповіді на ці запитання. Якщо для вирішення схожих завдань використовувати системи, побудовані на реляційних базах даних (якими є кошти BI і OLAP), то, очевидно, набір питань, на які система зможе відповісти, жорстко визначено набором сутностей, представлених в базі даних (закладу, міста, родичі повинні бути представлені у відповідних таблицях і полях БД). Принципова відмінність семантичної аналітики полягає в тому, що ми можемо вводити в модель будь-які нові сутності (не родичі, а колеги, не міста, а готелі, і т.д.), і використовувати їх при "формулюванні питань" системі, не змінюючи при це структури і змісту інших частин моделі.

Основні вимоги до функціоналу проектованої BI платформи:

- можливість додавати sparql-endpoint;

- додавання власних даних до rdf-сховища(import/export csv);

- можливість робити sparql-запити до різних sparql-endpoint та до локального сховища;

- створення графіків по створеним даним;

- зручний веб-інтерфейс.

3.2 Розробка структурної схеми

Структурна схема надає інформацію про взаємодію майбутніх частин програми, за якій функціонал вони будуть відповідати, та як будуть взаємодіяти між собою. Не можна сказати що вони є інформативними, оскільки на них не можна відстежити як дані передаються та змінюються. Тому в залежності від технічного завдання, структурні схеми розробляють для окремих частин програми.

Для розробки структурної схеми зазвичай використовують метод покрокової деталізації, щоб вказати всі компоненти з яких складається схема, а також набори підпрограм та підключені бібліотеки.

Структурними компонентами програми можуть називатися підсистеми, бази даних, бібліотеки і т.ін. А за допомогою зв'язків можна продемонструвати як вони взаємодіють між собою, обмінюються інформацією, підключати нові бібліотеки. При розробці структурної схеми програмного забезпечення були розглянуті основні модулі програми, та їх взаємодія.

Дуже важливо під час розробки, особливо якщо проект великий і має складну архітектуру, поділити програму на структури, щоб відображати логіку програми, показати місця де підключаються бібліотеки, та за що вони відповідають.

В наведеній схемі, взаємодія модулів, відображена в схемі ієрархії. Де до головного модуля підключаються розроблені модулі, однак схема не відображає порядок функціонування системи. Схема доповнюється розшифровую функцій які виконують підключення модулів.

У структурній схемі (рисунок 3.1) позначено зовнішні специфікації програми, які дають розуміння функціональній частині програми, за що вони відповідають, та як вони взаємодіють з вхідними та вихідними даними. Особливо важливо розуміти тип структури даних.

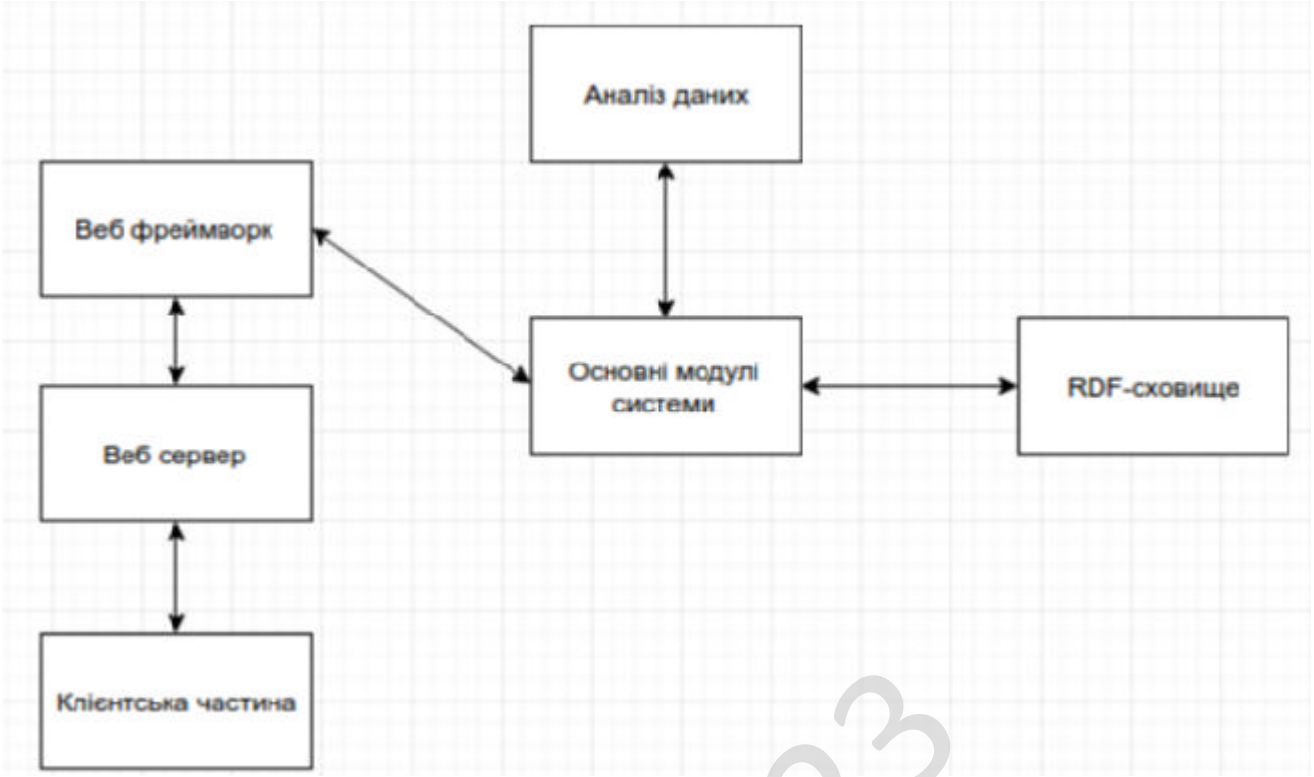

Рисунок 3.1 - Структурна схема

Розподіл програми на окремі модулі відбувався за принципом від загального до конкретного, де окремі модулі виступали у ролі сервісів, які виконували окремі задачі.

Структурна схема відображає головний принцип роботи програми, її основні та другорядні блоки, їх призначення, та їх взаємодію між собою. Це допомогає у розумінні роботи додатку, що полегшує подальшу його підтримку.

3.3 Розробка функціональної схеми

Функціональна схема, або також звана схемою даних, відображає взаємодію компонентів програми, вказуючи склад даних, а також їх призначення. При створенні такої схеми використовуються позначення, прийняті стандартом.

Розроблена функціональна схема допомагає відстежувати виконання існуючих вимог і функцій, відстежувати, які компоненти і модулі використовуються, як вони взаємодіють один з одним, за допомогою вхідних і

вихідних даних. Така схема відображає роль кожного функціонального елемента в додатку і з якими компонентами він взаємодіє. Для розробки даної схеми за основу була взята конструктивна схема.

Переглядаючи функціональну схему, ви можете прослідкувати принцип роботи всієї програми, як взаємодіють між собою клієнтська та серверна частини, а також які бібліотеки та технології використовуються для зберігання та передачі даних.

Цей тип діаграм зазвичай використовується для наочної демонстрації роботи алгоритму в спрощеній формі, щоб ви могли стежити за функціями та діями, які виконує алгоритм, які дані він використовує та який результат отримує.

Це допомагає оцінити складність алгоритму, зрозуміти принципи його роботи та контролювати можливі зміни та коригування. Подивіться, як функціональні схеми пов’язані одна з одною. Для опису компонентів та їх взаємодії використовувався національний стандарт. Там, де прямокутник показує окремі функціональні частини або як вони об’єднані у функціональні групи, причому кожна група має власну позначку та назву.

Є кілька можливостей показати функціональну схему, використовуючи принципову схему або змішану структуру, але в будь-якому випадку функціональна схема повинна відображати призначення пристрою.

На схемі (рисунок 3.2) вказано:

- умовне позначення у прямокутнику, якщо це функціональна частина;
- умовне графічне позначення, яке відображає позицію та дію функціональної частини.

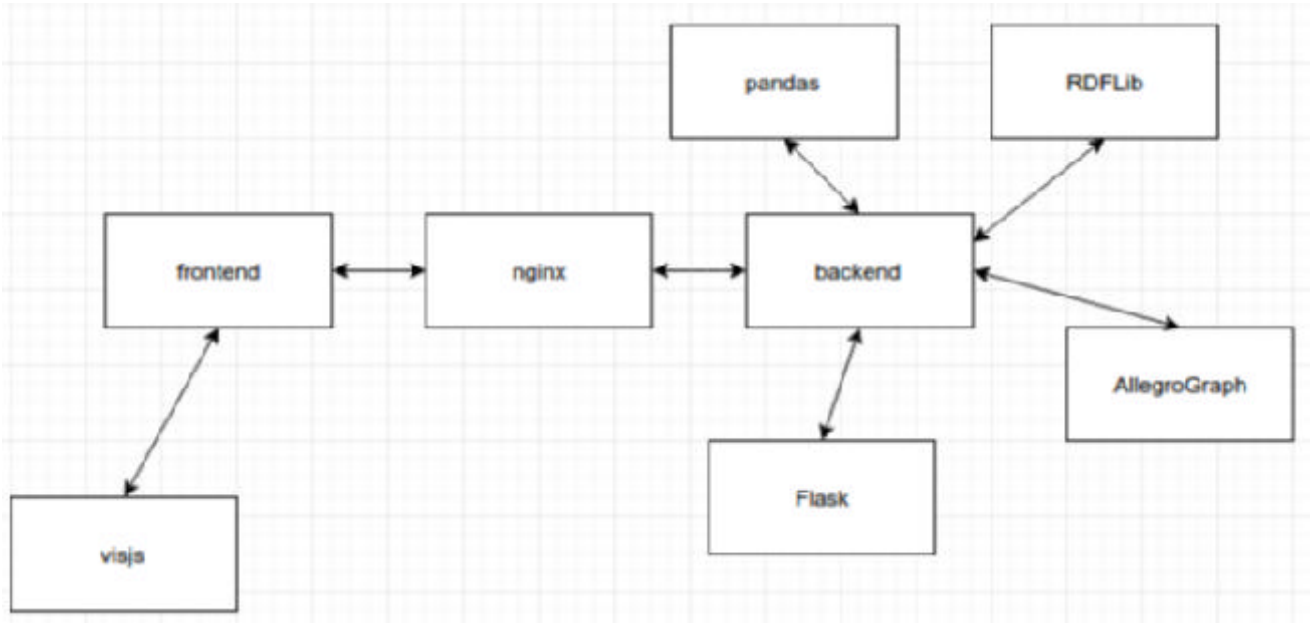

Рисунок 3.2 - Функціональна схема системи

У випадках, коли використовується принципова схема, функціональні частини разом із позначеннями розташування компонентів мають бути подібними до схеми. У таких випадках список елементів не використовується через використання цих принципових схем. Коли функціональна схема розробляється без використання схеми, використовуються загальні правила для зображення функціональних частин.

3.4 Розробка діаграми процесів

При моделюванні поведінки розробленої системи, можна простежити як змінювалася робота додатку, її стан, вхідні та вихідні дані та як все це впливало на розробку додатку. Як працюють алгоритми в додатку та які результати надають в залежності від вхідних даних.

У діаграми процесів, за допомогою графічних об'єктів можна побачити у зручному для сприйнятті форматі, які операції виконуються та як вони між собою пов'язані за допомогою стрілок.

На діаграмі (рисунок 3.3) можна побачити всі етапи роботи програми, від самого початку програми, коли користувач робить запит, як він приходить на сервер, потрапляє в контролер, обробляється, отримує дані з бази даних і повертає їх у вигляді відповіді користувачу. Об'єкти на діаграмі позначають процеси, які виконуються під час роботи програми, стрілки позначають як управління переходить від одного об'єкта до іншого. Є також структурні діаграми, вони схожі на діаграми процесів, але відображають статичний характер даних, такий опис більше описує взаємозв'язки об'єктів, коли діаграми процесів відображають динаміку роботи програми.

Головною рисою в діаграмі процесів є динаміка, на відмінно від інших, де переважає сценарій. В таких діаграмах сценарій не повинен суперечити послідовності виконання програми та бути безперервним. Проблеми виникають коли об'єкту потрібно бути в декількох місцях одночасно. Таке відбувається коли простежується нечітка логіка в послідовності дій в програмі.

Метою створення діаграм є розуміння послідовності дій процесів в програмі, щоб отримати кінцевий результат. Слідкуючи як працюють процеси, між собою, які дані вони отримують, як вони їх обробляють та передають користувачу, можна передбачити як повинна працювати програма.

Є різні методи побудови діаграм, які допомагають відстежити логіку та послідовність дії в процесах, але найкраще себе зарекомендував метод покрокового алгоритму побудови діаграм.

З усіх вище перелічених методів, та технічних рішень при проектуванні, та розробки програмного забезпечення, дивлячись на результати, можна зробити висновки, що я обрав оптимальні проектні рішення, які задовольняють головні потреби, які були поставлені переді мною на початку проекту. Оскільки з результатів можна побачити, що програма виконує всі поставленні задачі.

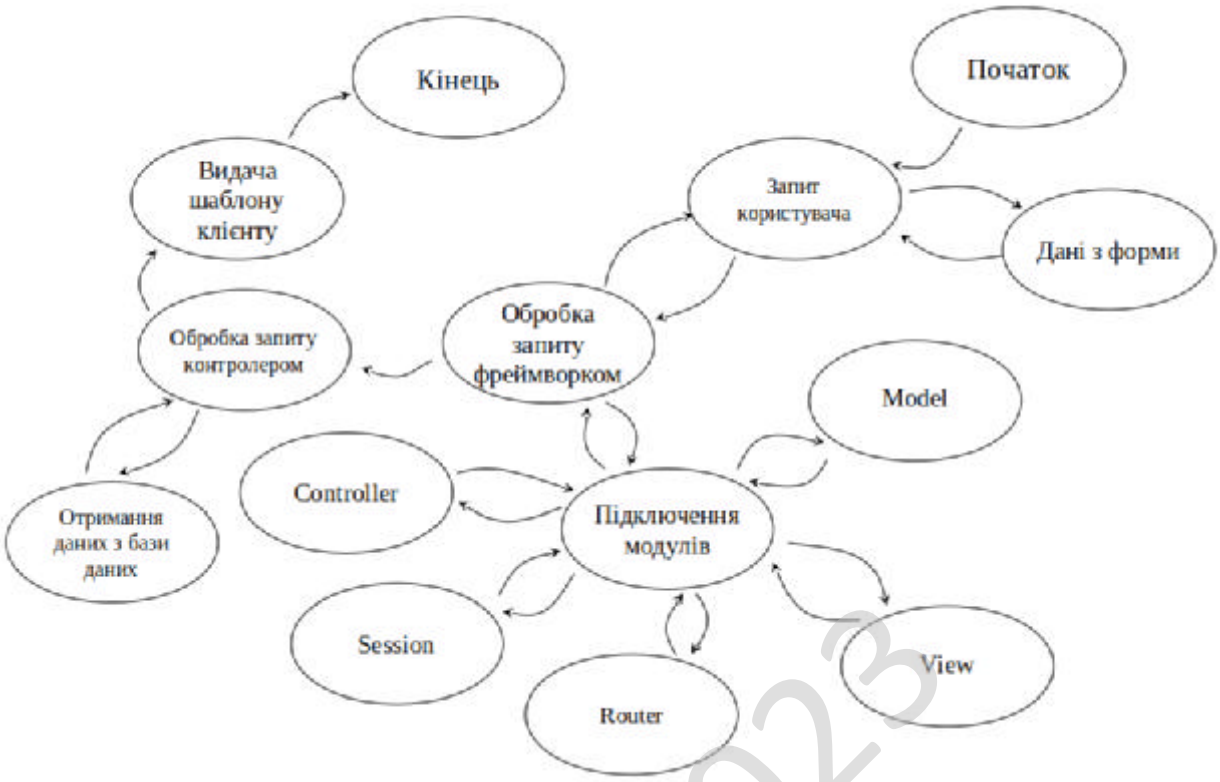

Рисунок 3.3 – Діаграма процесів системи

Розроблена діаграма дає представлення роботи додатку, як клієнтської так і серверної частини. Демонструє як взаємодіють між собою окремі частини додатку, та який результат отримує користувач при роботі з програмним забезпеченням.

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ ТА ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ І ПРОГРАМНИХ РІШЕНЬ

Існує багато готових рішень, які дозволяють впровадити BI технології у бізнес. Крім того, можна розробити індивідуальну систему, яка буде збудована персонально для вашого бізнесу. При цьому готові рішення також мають гнучкі налаштування, які дозволяють оптимізувати та персоналізувати функціонал. Для якісного вибору програмного забезпечення враховуйте наступне:

Платформа має бути єдиною. Головний плюс роботи з BI системами – можливість зберігати всі дані в одному місці та працювати лише з одним інструментом. При цьому він має бути комплексним. Звертайте увагу, щоб це була єдина платформа з великим функціоналом.

Використання хмарних рішень. Локальне зберігання можна назвати більш звичним, але хмарне зберігання даних дозволяє використовувати систему будь-де з будь-якого пристрою. Ви зможете керувати бізнесом прямо зі свого смартфона, та отримувати дані швидко та ефективно.

Готові інтеграції. Системи BI інтегруються з різними програмами та іншими технологіями. Краще, якщо налаштування будуть одразу заточені під певні інтеграції. Це значно спростить використання інструменту в бізнес.

Звертайте увагу на надання інформації. Візуалізація даних – це те, з чим ви працюватимете. Чим вона простіше і зрозуміліше, тим краще. При цьому візуалізація повинна бути інтерактивною і легко настроюваною.

Просте обслуговування. Система автоматизації BI ефективна тим, що не потребує технічної підтримки програмістів. Вона має зрозумілий інтерфейс, просте керування, систему навчання.

Мобільна версія. Не всі BI здатні працювати зі смартфонами. Це важлива умова у сучасних реаліях.

Проектування BI-системи

Сьогоднішні засоби бізнес-аналітики (BI, OLAP) в основному сконцентровані на кількісному, статистичному аналізі бізнес-даних.

Більшість подібних систем використовують жорстко задані в структурі бази даних набори параметрів і показників аналізу (для їх конфігурації потрібні дорогі фахівці).

За допомогою таких засобів складно описувати і аналізувати причинно-наслідкові зв'язки, важко формалізуються і різноманітні взаємовідносини об'єктів (наприклад, ступінь впливу тих чи інших факторів на результати певних процесів).

Семантична аналітика, побудована на принципі "відкритого світу", надає нові можливості вирішення подібних завдань. А в поєднанні з принципами нечіткої логіки, побудови симулятивних ("живих") моделей, її можливості стають практично безмежними.

Найпростішим засобом ілюстрації принципових відмінностей семантичної аналітики від інших засобів аналізу є інструмент Facebook Graph Search. Цей пошуковий інтерфейс, вбудований в соціальну мережу, може відповісти на запитання на кшталт "Які заклади подобаються моїм друзям?", "У яких містах бували мої родичі?". Зрозуміло, що жодна система повнотекстового пошуку (якими є сучасні пошукові машини) не зможе дати відповіді на ці запитання.

Якщо для вирішення схожих завдань використовувати системи, побудовані на реляційних базах даних (якими є кошти BI і OLAP), то, очевидно, набір питань, на які система зможе відповісти, жорстко визначено набором сутностей, представлених в базі даних (закладу, міста, родичі повинні бути представлені у відповідних таблицях і полях БД).

Принципова відмінність семантичної аналітики полягає в тому, що ми можемо вводити в модель будь-які нові сутності (не родичі, а колеги, не міста, а

готелі, і т.д.), і використовувати їх при "формулюванні питань" системі, не змінюючи при це структури і змісту інших частин моделі.

Основні вимоги до функціоналу проектованої BI платформи:

- можливість додавати sparql-endpoint;

- додавання власних даних до rdf-сховища(import/export csv);

- можливість робити sparql-запити до різних sparql-endpoint та до локального сховища;

- створення графіків по створеним даним;

- зручний веб-інтерфейс.

Вибір інструментів

Мова програмування: Python Python (найчастіше вживане прочитання — «Паайтон») - інтерпретована об'єктно-орієнтована мова програмування високого рівня з динамічною семантикою. Розроблена в 1990 році Гвідо ван Россумом. Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її привабливою для швидкої розробки програм, а також як засіб поєднання існуючих компонентів.

Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільованій так і у вихідній формі на всіх основних платформах.

В мові програмування Python підтримується декілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована.

Бібліотека для аналізу даних: pandas Pandas — вільна бібліотека Python, створена для проведення зручних маніпуляцій з даними та аналізу. Зокрема, дана бібліотека пропонує можливості роботи з числовими таблицями та часовими рядами. Назву бібліотека отримала від терміну з економетрії «panel data» — багатовимірного струкрутованого набору данних.

Можливості бібліотеки:

- DataFrame - тип данних з інтегрованою індексацією, створений для маніпуляцій над данними;

- Інструменти для зчитування та запису данних таких розширень, як: CSV, Excel, JSON, SAS, FWF;

- обробка відсутніх даних;

- підтримка операцій типу Group by;

- вставка та видалення строк/стовпців;

- злиття та приєднання датасетів;

- ієрархічна індексація осей для роботи з високорозмірними даними в низькорозмірних структурах даних;

- функціональність для роботи з часовими рядами;

- бібліотека Pandas оптимізована для отримання максимальної продуктивності при роботі, тому критичні частини коду написані на Cython та C.

Веб-фрейморк: Flask Flask — мікрофреймворк для створення веб-додатків на мові програмування Python, використовує набір інструментів Werkzeug, а також шаблонизатор Jinja2.

Веб-сервер: nginx nginx (engine x) — вільний веб-сервер і проксі-сервер. Є версії для сімейства Unix-подібних операційних систем (FreeBSD, GNU/Linux, Solaris, Mac OS X) та Microsoft Windows. Розробляється Ігорем Сисоєвим з 2002-го року для компанії Rambler і постійно вдосконалюється. Восени 2004 року вийшов перший публічно доступний реліз.

З 2011 року розробкою програми опікується заснована Ігорем Сисоєвим компанія Nginx Inc., яка розвиває вільну та комерційні версії продукту. Візуалізація даних: visjs Динамічна бібліотека візуалізації. Бібліотека покликана бути проста у використанні, можливість обробляти великі обсяги динамічних даних. RDF-сховище: allegrograph AllegroGraph – це сховище триплетів із закритим кодом, яке призначене для зберігання RDF триплетів(стандартний формат для Linked Data). Python драйвер для allegrograph: AllegroGraph Python Sesame API Робота з RDF: RDFlib.

Мова запитів SPARQL

Оскільки запити до системи здійснюються з допомогою SPARQL доречно оглянути синтаксис цієї мови. Ймовірно, самі по собі мови представлення онтологій не були б таксильно потрібні, якщо б не виникало необхідності автоматично обробляти, поповнювати вміст онтологій і виконувати до них запити. Найбільш популярним серед мов запитів до RDF-сховищ на сьогоднішній день є мова SPARQL.

Елементи мови SPARQL – Суб'єкт-Предикат-Об'єкт

Розглянемо дещо спрощений синтаксис SPARQL-запиту. Припустимо, онтологія містить наступні RDF-триплети:

(Foo1, category, "Total Members");

(Foo1, rdf:value, 199);

(Foo2, category, "Total Members");

(Foo2, rdf:value, 200);

(Foo2, category, "CATEGORY X");

(bar, category, "CATEGORY X");

(bar, rdf:value, 358).

Простежимо за ходом виконання запиту

SELECT ?cat ?val WHERE {?x rdf:value ?val. ?x category ?cat.

FILTER (?val>=200).}

Семантика запиту: видайте всі об'єкти cat предиката category, суб'єкт якого (x) є також суб'єктом предиката rdf:value зі значенням val, не меншим 200. Разомзі значеннями cat видати відповідні значення val.

Хід виконання запиту:

На місце змінної x можуть бути підставлені Foo1, Foo2 та bar (звихідною онтології), причому Foo2 може бути підставлений двічі, оскільки має 2 властивості category).

При підстановці Foo1 значення змінної val не задовольняєобмеження в реченні FILTER. У всіх інших випадках всеумови запиту виконані (див. результат).

Результат виконання запиту 3 пари значень (cat, val)

[

["Total Members", 200],

["CATEGORY X", 200],

["CATEGORY X", 358]

]

4.1 Розробка блок-схем та опис алгоритмів функціонування системи

Блок-схема надає можливість за допомогою графічних блоків та елементів, представити роботу додатку у зручній для сприйняття формі. За допомогою графічних блоків можна представити потоки, операції, дані, як вони обробляються та модифікуються. Дані які використовуються для вводу або виводу інформації прийнято позначати паралелограмом, дані які обробляються та модифікуються — прямокутником, графічний блок для прийняття рішення за допомогою заданої умови - ромбом, початок та кінець алгоритму — еліпсом. Всі ці графічні позначення використовуються згідно стандарту ДСТУ ISO 5807:2016. Використовуючи схему зручно простежити як виконується програма, з якими даними повинна працювати, та як вони послідовно змінюють свій стан. Це допомогає зрозуміти головне призначення програми, та її функціонал.

На блок-схемі (рисунок 4.1) можна простежити послідовність виконання дій розробленого клієнтського додатка, як в сукупності завантажується в браузері, як він приймає дії користувача, обробляє їх та надсилає на сервер.

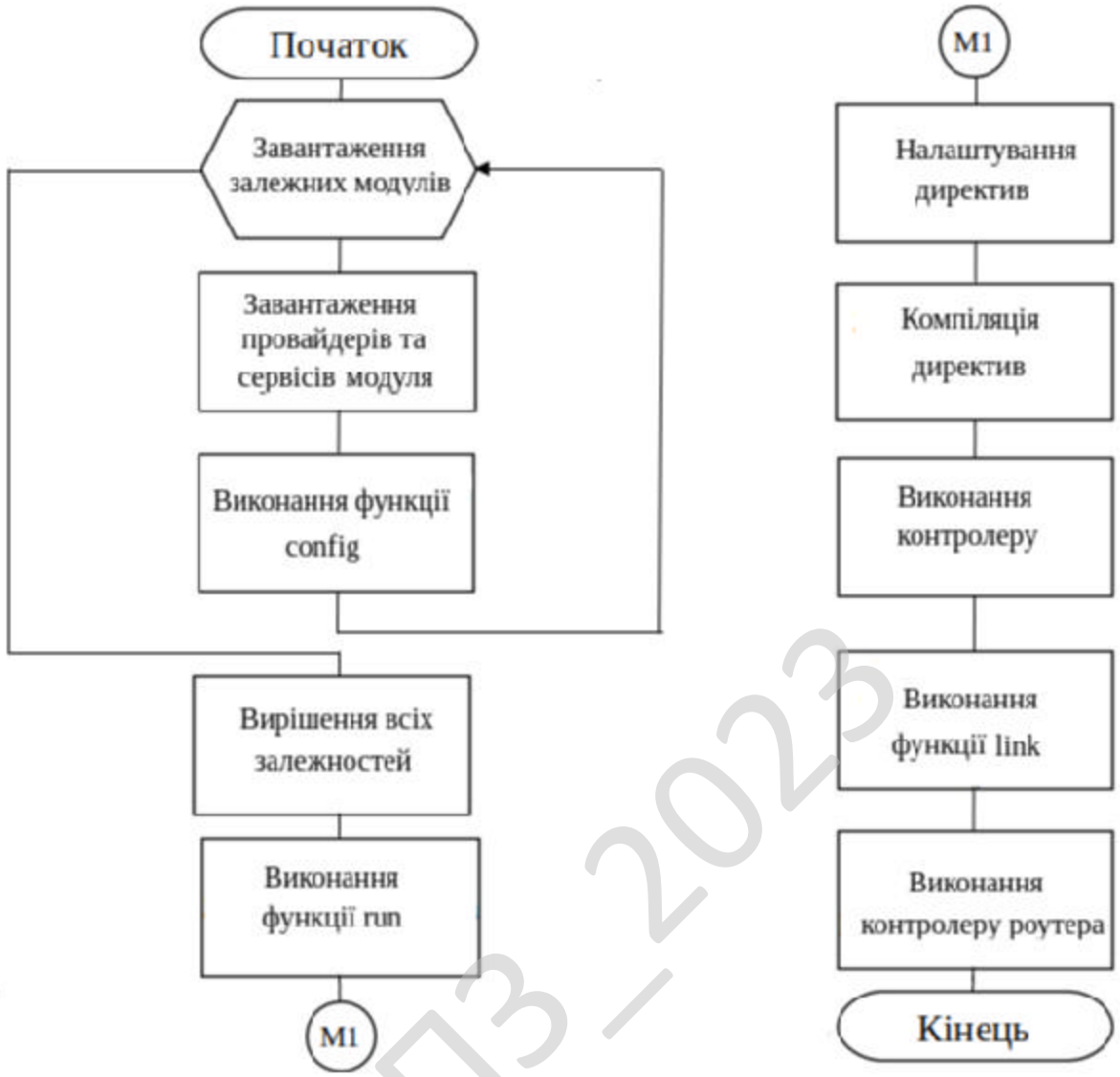

Рисунок 4.1 - Блок-схема головного модулю

Схема (рисунок 4.2) містить головні функції програми, та побудована так, щоб при її перегляді було зрозуміло функціонал та призначення системи. Вона є досить інформативна, щоб зрозуміти як побудована програма.

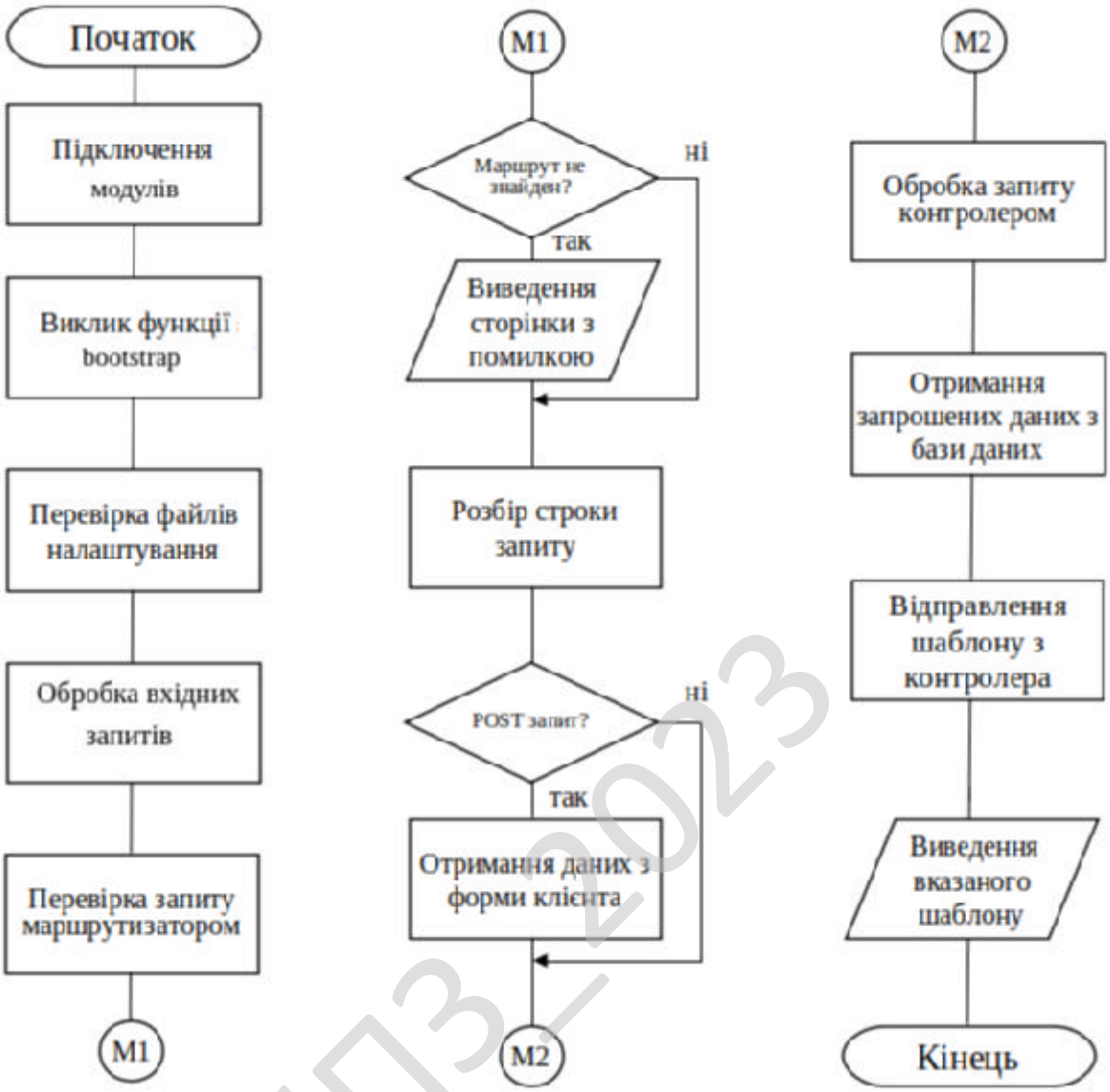

Рисунок 4.2 - Блок-схеми роботи системи

Всі запити виконувалися за допомогою HTTP-методів: GET, POST, PUT і DELETE, які використовувалися для створення, оновлення, читання та видалення даних з сервера. Кожний метод відповідав за власну операцію:

- POST - створював та додавав нові дані в базу;
- GET - отримував список заданих полів з сервера;
- DELETE - видаляв дані з бази;
- PUT - оновлював дані в базі.

Результати роботи прототипу

В роботі як було зазначено вище розроблявся прототип BI-платформи. Для цього нам потрібно зв’язати всі компоненти разом, та створити платформу. За основу взята мова програмування Python та веб-фреймворк Flask.

Зробимо з допомогою мови SPARQL запит до бази знань. В даному прототипу використовується загально відома dbpedia, але передбачена можливість інтеграції та подальшого використання. Зробимо тестовий запит до бази(рисунок 4.3):

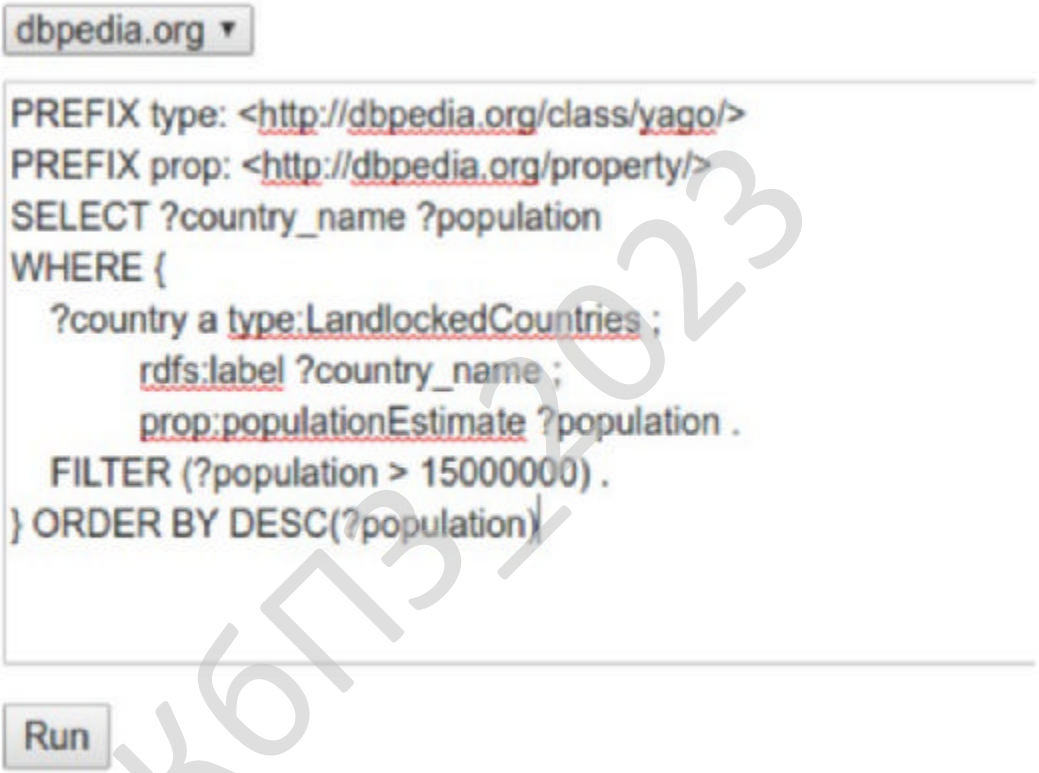

Рисунок 4.3 Тестовий запит до бази

Після надсилання запиту ми отримуємо відповідь у вигляді таблиці. Таблицю ми можемо редагувати, щоб мати змогу редагувати можливі неправильні данні. Приклад таблиці на рисунку 4.4.

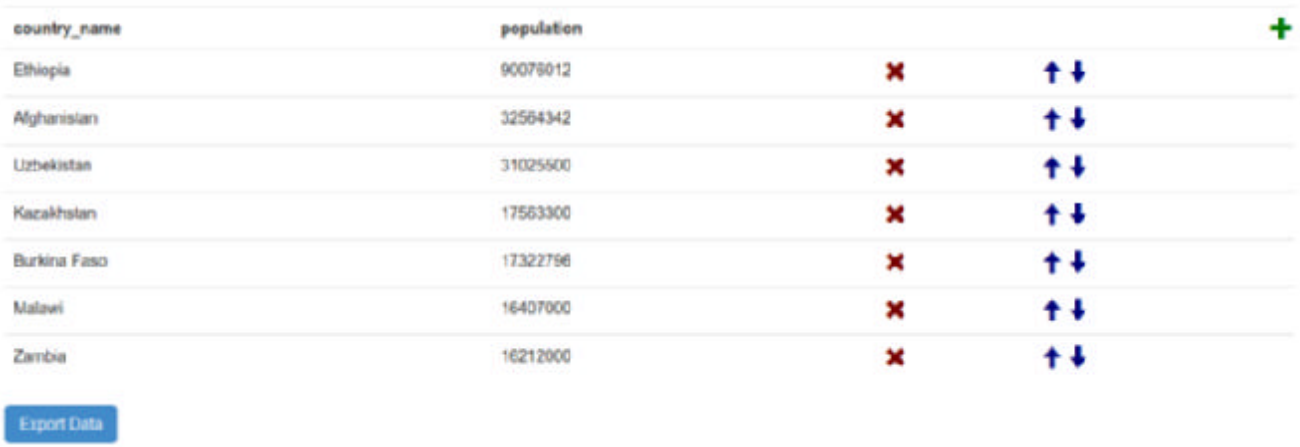

Рисунок 4.4 - Вихідна таблиця

Якщо вихідних даних забагато, і необхідна обробка за допомогою якихось сторонніх засобів є можливість скачати дані у вигляді CSV-файлу. Приклад вмісту на рисунку 4.5.

```
[Ethiopia , 90076012 ],
[Afghanistan , 32564342 ],
[Uzbekistan , 31025500 ],
[Kazakhstan , 17563300 ],
[Burkina Faso , 17322796 ],
[Malawi , 16407000 ],
[Zambia , 16212000 ],
```

Рисунок 4.5 - Приклад експортованого файлу з вихідними даними

Якщо дані правильно відформатовані, то передбачена можливість створення графіків.

Приклад створений на даних зазначених вище зображений на рисунку 4.6.

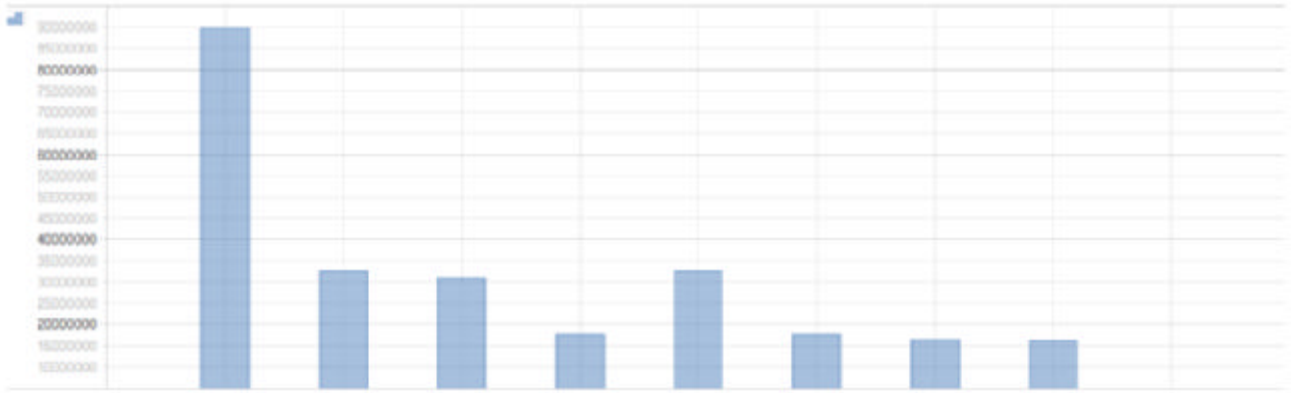

Рисунок 4.6 - Візуалізація даних

4.2 Захист розробленого програмного забезпечення

В першу чергу варто перевірити розроблений додаток автоматизованими засобами, вони зможуть перевірити додаток на найрозповсюдженіші помилки. Вони перевіряють не тільки код написаний розробником, а й бібліотеки та модулі які використовує розробник, а вони складають найбільшу частку зломів. Згідно з останніми перевірками, понад 50% npm модулів, не оновлюються більше декількох років.

Є декілька способів перевірити модулі перед їх використанням, в першу чергу варто переглянути код, та перевірити чи повністю модуль покриває потреби розробника. Він повинен перевірити які ще бібліотеки використовує цей модуль і коли вони останній раз оновлювалися, чим менше він має залежностей, тим краще.

Великі компанії мають спеціальних працівників, які проводять ревізії пакетів перед їх використанням, та складають спеціальні білі списки дозволених модулів.

Сама npm занепокоєна тим, як часто у бібліотеки почали впроваджувати майнерів та віруси для збору приватних даних. Тому вони випустили спеціальний модуль npm audit. Він сканує встановленні модулі в проект і порівнює їх з чорним списком модулів, які містять уразливості. Сьогодні навіть команда npm install підкаже, чи мають встановленні модулі вразливості. А нова команда npm

audit fix, пропонує замінити вразливі пакети, або оновити до більш захищених версій, якщо такі існують. Але так можна знайти лише ті модулі, про які вже повідомили користувачі та розробники.

Також для перевірки безпеки додатку, важливим є написання тестів. Зазвичай розробники на AngularJS використовують концепцію Jasmine's Behavior-Driven Development (BDD), при написанні тестів. Jasmine − це вільний фреймворк для тестування коду написаного за допомогою JavaScript, його можна запустити на будь-якій платформі. Його перевагою є зручний інтерфейс, та зрозуміле налаштування. Для тестування одиничних тестів часто використовують бібліотеку Karma, головна мета якого наблизити тестування додатку до його налаштувань в продакшен.

Також важливо використовувати протокол HTTPS, він є набагато надійнішим HTTP. HyperText Transfer Protocol Secure (HTTPS) — забезпечує шифрування даних і дозволяє надійно захищати дані користувачів при передачі в Інтернеті. Сьогодні більшість сайтів використовують саме цей протокол, оскільки він є обов'язковим при передачі приватних даних на сервер, особливо коли передаються паролі або дані кредитних карт. Також варто бути обережним с cookie файлами, які передаються під час авторизації. Зловмисник може перехопити їх та підробити запит на сервер, щоб цього уникнути потрібно використовувати HTTPS протокол.

Також слід брати до уваги поширені CORS та CSRF атаки. CSRF- атака це коли на сторінки, робиться непомітна форма, щоб відправляє запит з приватними даним користувача. Якщо на сайті авторизація відбувається тільки за допомогою кукі, то така атака може бути успішна. Щоб цього уникнути потрібно використовувати спеціальні токени. Для підпису XMLHttpRequest токен також зберігається в кукі. Тоді JavaScript може прочитати його з домену і додати в заголовок, а сервер - перевірити, що в заголовку міститься коректний токен.

При крос-домених запитів, браузер додає заголовок Origin, який зберігає домен, з якого відбуваються запити. Сервер у цьому випадку повинен

перевірити домен і відповісти спеціальним заголовком. Якщо сервер дозволяє доступ, він повинен відправити заголовок Access-Control-Allow-Origin, що зберігає домен запиту. Якщо Access-Control-Allow-Origin відсутній, то сервер завершує запит з помилкою. При таких запитах не передаються куки і заголовки HTTP-авторизації.

Висновки до розділу

Було спроектовано BI-платформу, яка відповідає сучасним вимогам бізнесу. При проектуванні враховані можливі сценарії розвитку системи (наприклад додання можливості комбінувати різні джерела даних). Для розроблення прототипу використовувались мови програмування Python, Javascript.

Python відіграє ключову роль в системі, адже відповідає за надання доступу до даних, їх обробка, зберігання та т.п. JavaScript допомагає створити інтерактивний та зручний інтерфейс для кінцевого користувача, вирішуючи проблему редагування даних в браузері.

5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Модуль роботи із джерелами даних

Даний модуль розроблено таким чином, щоб при необхідності у систему аналізу бізнес-даних було можливо легко інтегрувати інші джерела даних. Досягнуто це за допомогою поліморфізму – як одного з базових принципів ООП, та за допомогою Spring-фреймворку, а саме завдяки технології впровадження залежностей (Dependency Injection). Архітектуру класів клієнтів наведено на рисунку 5.1

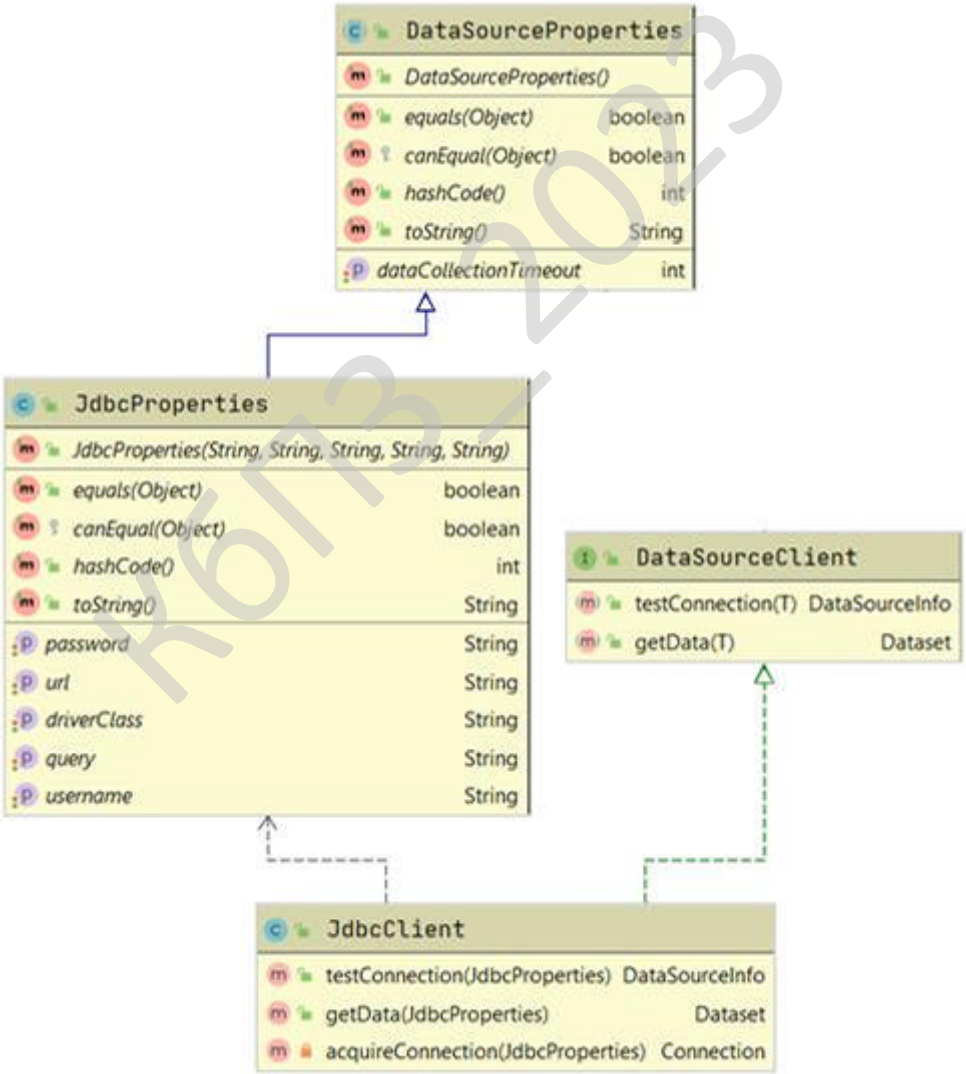

Рисунок 5.1 – Діаграма класів клієнтів джерел даних

Інтерфейс DataSourceClient описує загальну поведінку усіх джерел даних та містить два методи:

– testConnection;

– getData.

Перший метод потрібен для того, щоб адміністратор системи міг переконатися, чи є введені ним дані для підключення до джерела даних валідними та чи є можливість у системи підключитися до нього. У якості результату проведення перевірки з'єднання цей метод повертає об'єкт класу DataSourceInfo. Для кожної імплементації клієнта є можливість розширення цього класу шляхом додавання специфічної інформації конкретного джерела даних. На рисунку 5.2 наведено приклад для JDBC клієнта.

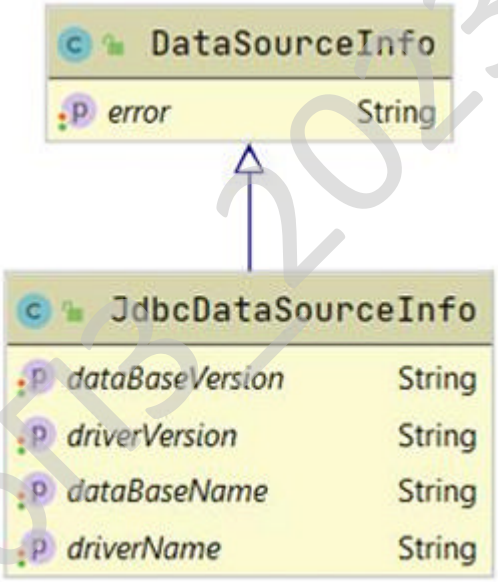

Рисунок 5.2 – Діаграма класів інформації про джерело даних

Загальним полем для всіх джерел даних є поле error, яке є незаповненим у разі вдалого встановлення з'єднання, або ж містить інформацію про помилку, яка виникла при зворотньому випадку.

Специфічною інформацією для джерела даних JDBC є:

– dataBaseVersion;

– версія БД;

– driverVersion;

– версія JDBC драйвера;

– dataBaseName – назва БД;

– driverName – назва JDBC драйвера.

Усі ці поля отримуються з об’єкту DatabaseMetaData, який у свою чергу отримується з об’єкту Connection. Слід зазначити, що DatabaseMetaData та Connection – це частина Java API, що описує роботу із базами даних (переважно реляційними), і кожен JDBC драйвер імплементує ці інтерфейси для кожної БД окремо. На рисунку 5.3 наведено діаграму класів, що демонструє результат роботи методу getData.

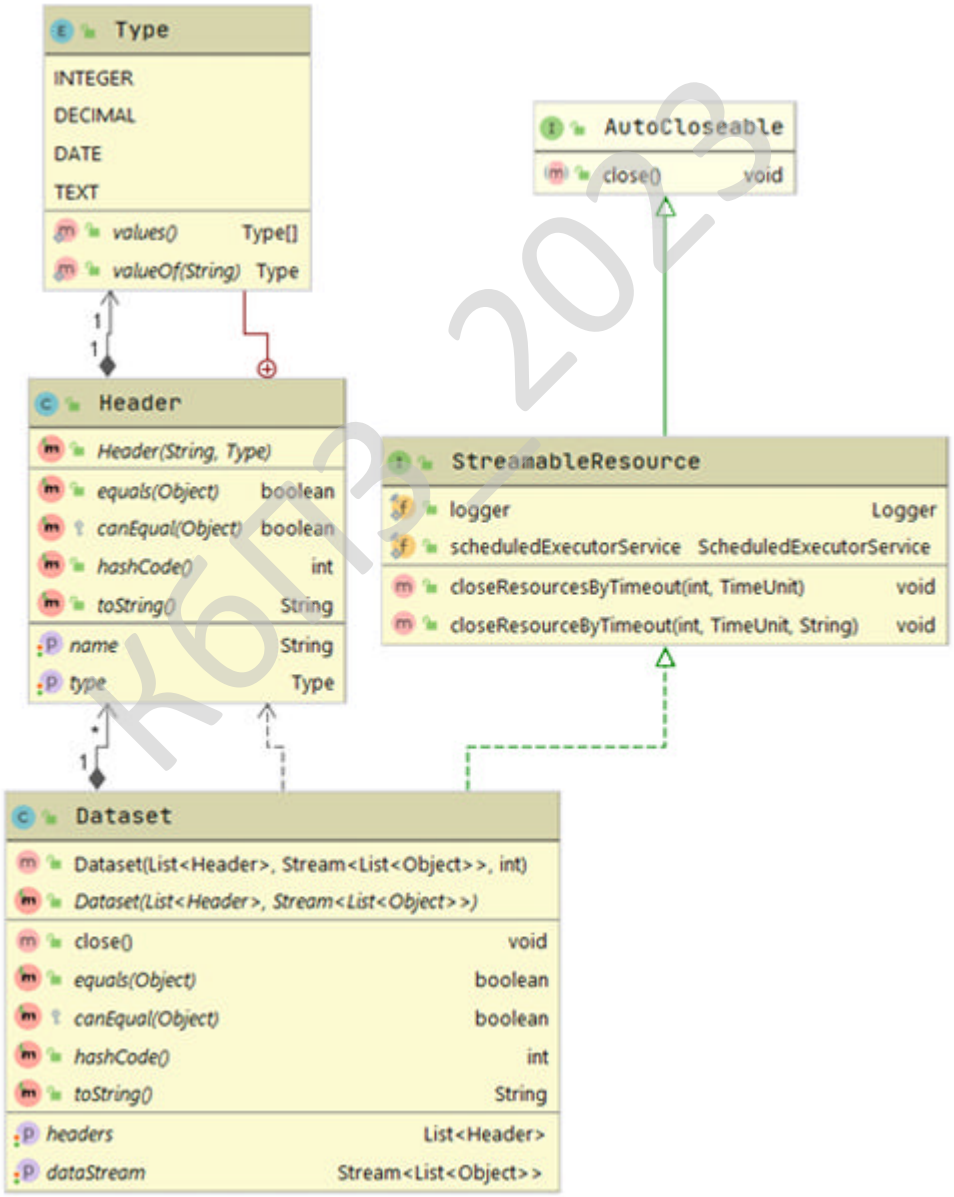

Рисунок 5.3 – Діаграма класів набору даних

Другий метод інтерфейсу DataSourceClient – getData, виконує безпосередньо вилучення даних з джерела. На відміну від методу testConnection, результатом роботи методу getData є об’єкт класу Dataset, який є уніфікованим для всіх джерел даних і для кожної реалізації клієнта немає необхідності його розширення. Завдяки такому підходу спрощується подальша обробка цих даних, а саме збереження їх у сховищі та представлення цих даних користувачу.

Клас Dataset має два поля:

- headers;

- dataStream.

З точки зору реляційної моделі даних кожен об’єкт класу Dataset є відношенням, де поле headers – це список атрибутів, а поле dataStream – це набір кортежів.

Перше поле – це список об’єктів типу Header, який містить інформацію про ім’я колонки та її тип. Кожна реляційна база даних має різноманітну кількість типів даних, які відрізняються певними властивостями, але для всіх можна виділити основні типи даних:

– цілочисельні;

– десяткові;

– текстові;

– дати.

У програмній реалізації описані вище типи визначені за допомогою спеціального типу даних – перечислення (enum), який дозволяє змінній бути набором заздалегідь визначених констант. Також на діаграмі класів видно, що перечислення Type є внутрішнім типом класу Header. Поле dataStream має тип Stream<List<Object>>, що можна трактувати як потік зі списку об’єктів.

Типом цього поля обрано Stream, тому що це дозволяє динамічно отримувати дані з датасету без повного їх завантаження у пам’ять. Деякі імплементації Stream вимагають явного закриття ресурсів, що змушує розробника викликати метод close або поміщати об’єкт цього типу у

конструкцію try-withresources, яка автоматично викличе метод close. Для того, щоб не виникало ситуації, коли за якихось обставин ресурси не були закриті (наприклад, виникнення помилкі до обробки потоку даних), створено інтерфейс StreamableResource. Він містить стандартний метод closeResourcesByTimeout, призначення якого примусово закривати ресурси через заданий проміжок часу. Досягається це за допомогою ScheduledExecutorService з пакету java.util.cuncurrent, який дозволяє сконфігурувати інтервал або час, на яке буде відкладено виконання задачі. На рисунку 5.4 наведено лістинг коду, який демонструє описаний вище метод.

```
default void closeResourcesByTimeout(final int timeout, final TimeUnit timeUnit) {
    scheduledExecutorService.schedule(() -> {
        try {
            logger.info("Closing stream resource after timeout: {} {}", timeout, timeUnit);
            this.close();
        } catch (Exception e) {
            logger.error("Couldn't close streamable resource: {}", e.getMessage());
            throw new DataSourceException("Couldn't close streamable resource: " + e.getMessage(), e);
        }
    }, timeout, timeUnit);
}
```

Рисунок 5.4 – Фрагмент коду метода closeResourcesByTimeout

Цей метод викликається у конструкторі класу Dataset, тобто одразу після створення об’єкту датасету починається відлік часу, після якого ресурси будуть закриті та буде неможливо вичитати дані з датасету. Інтерфейс DataSourceClient є параметризованим.

Узагальнення (generics) у мові програмування Java дозволяють типам або методам працювати із об’єктами різних типів, при цьому забезпечуються «безпека типів» на етапі компіляції. У реалізованому в системі інтерфейсі даний підхід використовується для узагальнення аргументів, визначених у методах. На рисунку 5.5 представлено, як це реалізовано синтаксично.

```java
public interface DataSourceClient<T extends DataSourceProperties> {
    DataSourceInfo testConnection(T properties);
    Dataset getData(T properties);
}
```

Рисунок 5.5 – Фрагмент коду інтерфейса DataSourceClient

Такий підхід забов’язує розробника не тільки імплементувати методи інтерфейсу, а й наслідувати клас DataSourceProperties та визначати його у якості параметра при оголошенні класу клієнта.

Конструкція «T extends DataSourceProperties» говорить про те, що параметром класу, який реалізує інтерфейс, може бути лише тип, який є нащадком класу DataSourceProperties, у інакшому випадку виникне помилка компіляції. Ідея даного параметру полягає у тому, що для кожної імплементації клієнта джерела даних при виконанні будь-якого з методів, аргумент properties містить усю необхідну інформацію, що використовується для підключення до джерела даних та отримання датасетів.

У випадку JDBC клієнта, створено сутність JdbcProperties, яка містить наступні поля:

– url;

– username;

– password;

– driverClass;

– query.

Перші чотири параметри необхідні для встановлення підключення із базою інструментами JDBC та створення об’єкту Connection. Останній параметр містить SQL-запит, за допомогою якого отримуються дані та формується об’єкт Dataset. Базовий для всіх клієнтів клас DataSourceProperties містить одне єдине поле – dataCollectionTimeout, яке визначає таймаут на закриття потоку даних об'єкту Dataset, що було зазначено вище.

Цей параметр має цілочисельний тип та завжди становить триста секунд (що дорівнює п'яти хвилинам), але при необхідності адміністратору можна надати можливість кастомізації даного параметру. На рисунку 5.6 наведено діаграму класів, що демонструють принцип роботи методу getData у імплементації клієнта джерел даних JdbcClient. Алгоритм вилучення даних за допомогою інструментів JDBC:

а) встановлення підключення до БД. На даному етапі викликається приватний метод acquireConnection, який на основі необхідної для встановлення з'єднання інформації з сутності JdbcProperties створює об'єкт Connection та повертає його у якості результату;

б) виконання SQL-запиту. На основі об'єкту Connection, отриманого на попередньому етапі, створюється об'єкт Statement, якому передається запит у вигляді строки. Результатом цього шагу є об'єкт ResultSet;

в) створення об'єкту Dataset. Допоміжний клас ResultSetConverter на основі об'єкту ResultSet ініціалізує новий об'єкт Dataset. Методи класу ResultSetConverter є статичними, тому ініціалізації об'єкту не виконується. Більш того, оскільки ResultSetConverter є Utility класом, для того щоб не можна було створювати екземпляри даного класу, він має приватний конструктор без аргументів.

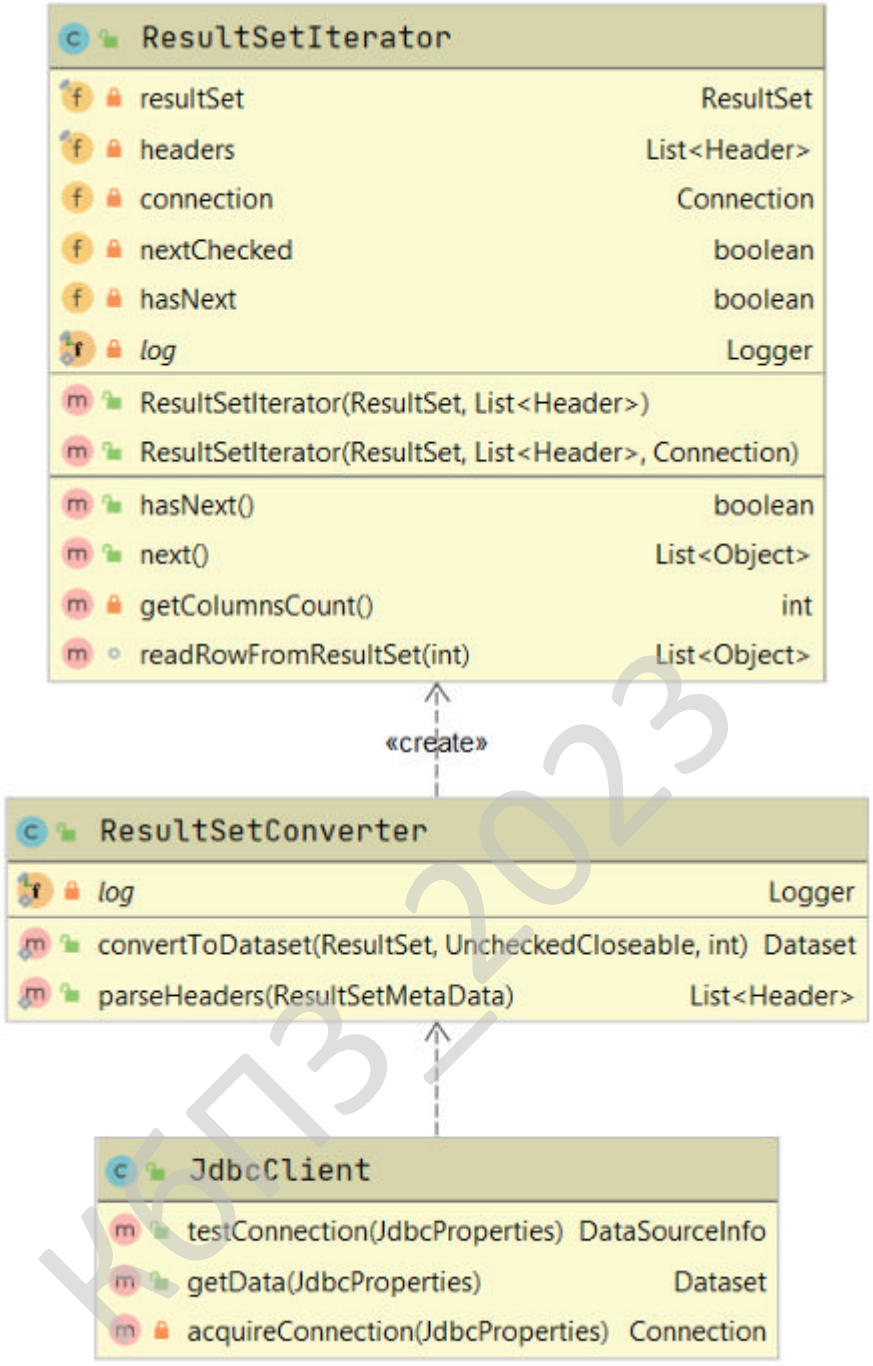

Рисунок 5.6 – Діаграма класів, що демонструє принцип роботи методу getData

Наступним кроком виконується створення потоку даних на основі об’єкту ResultSet. Для цього використано патерн поведінки – ітератор. Цей шаблон надає послідовний доступ до всіх елементів складеного об’єкта без розкриття його

внутрішнього представлення [16]. Як само реалізовано цей патерн зображує діаграма класів з рисунку 5.7.

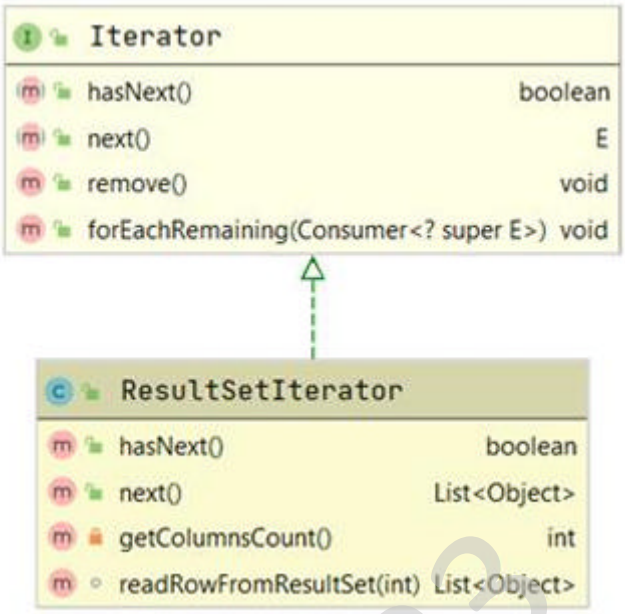

Рисунок 5.7 – Діаграма класів патерну ітератор

Інтерфейс Iterator є параметризованим, де E – це тип, який повертає ітератор при виклику методу next. У випадку класу ResultSetIterator цим типом виступає список об’єктів (List<Object>). Об’єкт ResultSetIterator містить у собі об’єкт ResultSet, з якого виконується зчитування даних. Недоліком такого підходу є те, що поки ітератор не поверне усі елементи послідовності (тобто поки метод hasNext повертає значення true), об’єкт ResultSet не буде закрито, а це у свою чергу означає, що із БД буде встановлено з’єднання. Але даний підхід дозволяє уникнути обмеження розміру датасету за пам’ятью. При розробці системи аналізу бізнес-даних перевагу було віддано саме тому підходу, який не обмежує розмір даних за пам’яттю, оскільки більш важливим показником системи є те, який саме об’єм даних може бути опрацьованим при їх завантаженні у сховища даних.

Кожен клас-імплементація інтерфейсу клієнта джерела даних має містити анотацію Service. Це анотація Spring-фреймворку, за допомогою якої реалізується патерн фасад. Шаблон фасад – це структурний шаблон

проектування, який дозволяє скрити складність системи завдяки зведенню усіх можливих зовнішніх викликів до одного об’єкту, що делегує їх відповідним об’єктам у системі. Також дана анотація дозволяє використовувати впровадження залежностей. Завдяки цьому немає необхідності кожен раз створювати новий об’єкт, натомість Spring-фреймворк впровадить цей об’єкт, взявши готовий екземпляр з DI-контейнеру або створить новий, в залежності від того, який обрано цикл життя біна. Налаштовується це за допомогою анотації Scope. За замовчуванням – це singleton, тобто при роботі системи буде існувати тільки один екземпляр класу. На рисунку 5.8 наведено діаграму класів, на якій продемонстровано залежність між класами, де використовується впровадження залежностей, а саме клієнту джерела даних.

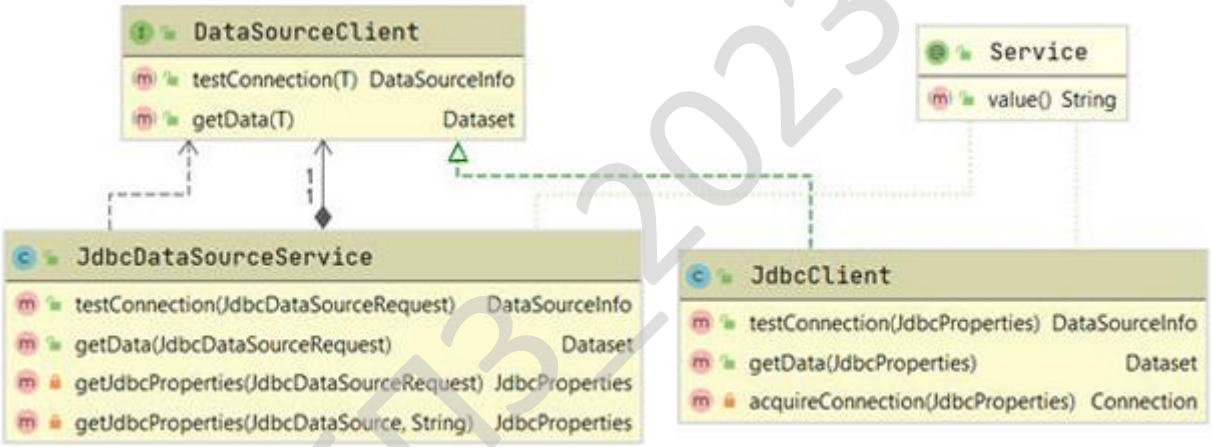

Рисунок 5.8 – Діаграма класів, що демонструє використання впровадження залежностей.

Для клієнт-серверної взаємодії користувацького інтерфейсу з сервером створено контролер, який має два POST методи, що відповідають двом сценаріям використання – перевірка підключення до джерела даних та отримання даних з нього. Обидва методи у якості тіла запиту приймають об’єкт JdbcDataSourceRequest, який містить ідентифікатор, якому відповідає запис у БД із інформацією про джерело даних, та SQL-запит, якщо поточним методом є отримання даних. Екземпляр класу JdbcDataSourceRequest є об’єктом передачі

даних (DTO). Даний патерн використовується для передачі даних між модулями системи і на відміну від об'єкту бізнес-логіки, він не має поведінки.

Модуль завантаження даних, як і модуль роботи із джерелами даних, взаємодіє з БД, але останній обмежується лише операцією читання. Окрім цієї операції, модуль завантаження даних надає інтерфейс для виконання операцій створення й видалення об'єктів БД, та запису й видалення даних. Інтерфейс StorageDao містить методи, які відповідають наведеним операціям. У таблиці 5.1 наведено методи інтерфейсу та відповідні їм SQL-оператори.

Таблиця 5.2 – Методи інтерфейсу StorageDao

Метод	Аргументи	Призначення	SQL-оператор
create(String, List<Header>)	Ім'я датасету, список колонок.	Створює таблицю датасету у сховищі.	CREATE
insert(String, Dataset)	Ім'я датасету, датасет.	Додає дані у таблицю датасету. Якщо таблиці не існує, викликається метод create.	INSERT
select(String, List<Header>)	Ім'я датасету, список колонок.	Виконує вибірку даних на основі необхідних колонок.	
select(String)	Ім'я датасету.	Виконує вибірку даних з усіх колонок датасету.	
selectByQuery(String)	SQL-запит.	Виконує вибірку даних за попередньо сформованим запитом. При обробці запиту природньою мовою генерується SQL-запит, який виконується за допомогою даного методу.	
delete(String)	Ім'я датасету.	Видаляє таблицю датасету зі сховища.	DROP

Даний інтерфейс розроблено за принципом шаблону проектування DAO, тобто існує абстрактний інтерфейс, який надає доступ до БД. У якості реалізації

даного інтерфейсу виступає клас SqlStorageDao, який інкапсулює механізми взаємодії із реляційними БД.

Таким чином, у систему закладено можливість масштабування шляхом додавання нових типів сховищ даних, наприклад NoSQL БД, взаємодія із якими відрізняється від реляційних БД, що вимагає розробки власної імплементації StorageDao. У розробленій системі у якості сховищ даних обрано дві реляційні БД – MySQL та PostgreSQL. Кожному типу сховища відповідає константа с перечислення StorageType, яке містить єдине поле – queryProviderClass, яке містить класпостачальник специфічних запитів для конкретного діалекту SQL. Екземпляр класу SqlStorageDao створюється за допомогою єдиного конструктора, який приймає два аргументи: – HikariDataSource, який містить пул об'єктів Connection для взаємодії зі сховищем; – StorageType, що визначає тип сховища даних, із яким буде взаємодіяти даний екземпляр. Для отримання об'єктів StorageDao використано породжуючий шаблон проектування – фабричний метод. Клас StorageDaoFactory є компонентом Spring та за замовчуванням життєвий цикл цього об'єкта – сінглтон. Даний клас має єдиний публічний метод – getStorageDao, який на основі об'єкту Storage, який отримується із БД, повертає екземпляр StorageDao. Фабричний метод, що надає інтерфейс створення екземплярів, розроблено таким чином, щоб кожній сутності сховища відповідав лише один екземпляр StorageDao, оскільки він містить пул з'єднань із БД і треба уникати випадків створення цих об'єктів, тому що вони займають вагому частину ресурсів. Для цього створено поле storageDaoById, що представляє собою колекцію Map, яка містить ідентифікатори сховищ та відповідні їм екземпляри StorageDao. У випадку, коли у колекції немає екземпляру StorageDao для сховища, виконується його ініціалізація. Створюється об'єкт HikariDataSource, який на основі параметрів підключення створює пул з'єднань. Стандартний розмір пулу – 12.

Наступник кроком ініціалізується об'єкт StorageDao, який зберігається у колекцію та повертається у якості результату методу. У випадку, коли колекція

містить відповідний об’єкт для ідентифікатора сховища, ініціалізація не виконується і об’єкт одразу повертається. Для запобігання ситуацій, коли декілька потоків звертаються до одного і того ж ресурсу (у даному випадку – колекція Map), виконано синхронізацію на рівні об’єкту при зверненні до колекції.

Після першого звертання до колекції, у випадку, коли відповідний екземпляр StorageDao не знайдено, виконується синхронізація на рівні об’єкту самого екземпляру, що виконує даний метод. Даний механізм гарантує, що тільки один потік зможе виконувати даний блок коду для даного екземпляру класу. Оскільки фреймворк Spring гарантує, що екземпляр класу StorageDaoFactory буде єдиним об’єктом у системі, то синхронізацію виконано саме на рівні об’єкту замість синхронізації на рівні класу. Алгоритм роботи кожного метод класу SqlStorageDao, незалежно від того, яка операція виконується зі сховищем даних, складається з наступних кроків: − побудова SQL-запиту на основі вхідних даних; – виконання SQL-запиту за допомогою класу JdbcTemplate – компоненту фреймворку Spring; – отримання результату виконання SQL-запиту. На рисунку 5.9 наведено діаграму класів, що демонструє зв’язок між імплементаціями інтерфейсів StorageDao та SqlQueryBuilder.

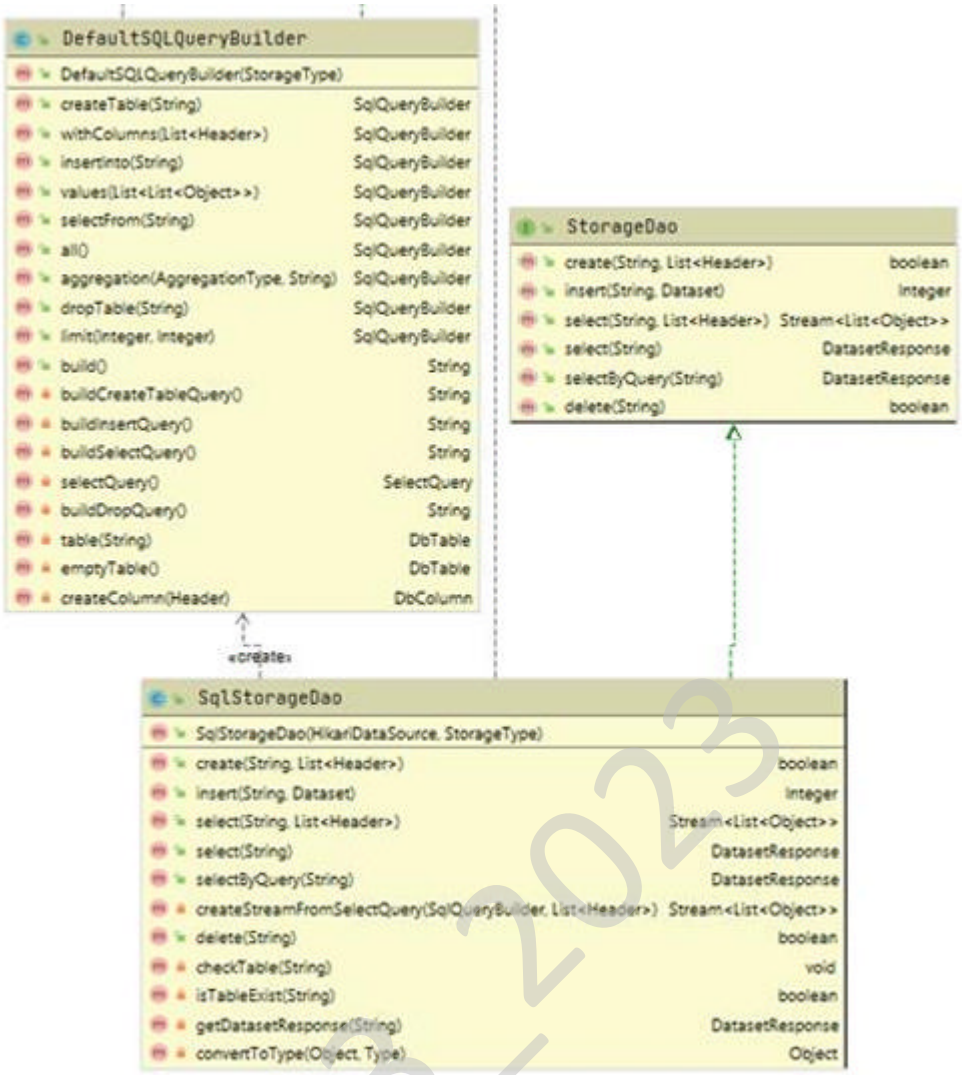

Рисунок 5.9 – Діаграма класів, що демонструє зв’язок між імплементаціями StorageDao та SqlQueryBuilder

Для побудови SQL-запитів для специфічного типу сховища даних розроблено інтерфейс SqlQueryBuilder та відповідний клас-реалізацію DefaultSqlQueryBuilder. Останній є обгорткою над бібліотекою SqlBuilder, який надає спрощений інтерфейс побудови запитів, реалізований за принципом породжуючого шаблону проектування Builder.

Таким чином, логіку взаємодії із використаною бібліотекою повністю інкапсульовано у класі DefaultSqlQueryBuilder. Бібліотека SqlBuilder не покриває усі необхідні функціональні можливості, що необхідні для реалізації бізнес-

логіки системи, тому додатково було розроблено пакет queryprovider, який надає додатковий функціонал, що використовується у класі DefaultSqlQueryBuilder.

Однією з основних відмінностей між сховищами даних є типи колонок, які використовуються при побудові датасетів. У таблиці 5.3 наведено відповідність між типами, які підтримуються розробленою системою аналізу бізнес-даних, та типами у сховищах даних (MySQL та PostgreSQL). Типам даних системи відповідають константи перечислення Type, що розроблено у модулі роботи із джерелами даних.

Таблиця 5.3 – Відповідність між типами даних у системі та сховищами даних

Система аналізу бізнес-даних	MySQL	PostgreSQL
DATE	DATETIME	timestamp without time zone
DECIMAL	DOUBLE	double precision
INTEGER	BIGINT	bigint
TEXT	TEXT	text

Таким чином, для цього розроблено інтерфейс SqlQueryProvider, який має дві реалізації, що відповідають кожному підтримуваному у системі сховищу даних. Метод convertParameterTypeToColumnType виконує конвертацію типу даних системи до типу даних сховища, та повертає інформацію про тип, що інкапсульовано у сутності SqlDataType.

Діаграма розгортання системи

Діаграму розгортання системи аналізу бізнес-даних наведено у додатку И. На діаграмі присутні три вузли пристрої:

− сервер системи;

− HTTP-клієнт для адміністрування;

− пристрій користувача.

Сервер системи складається з трьох вузлів середи виконання. Перший вузол – це чатбот, сервіс, який взаємодіє із зовнішнім slack-сервером та із модулем інтеграції даних. Slack-сервер у свою чергу взаємодіє із пристроєм користувача. Наступний вузол – це модуль інтеграції даних, що виконує функції по взаємодії із джерелами та сховищами даних та оброблення запитів природною мовою.

Останній вузол – це БД PostgreSQL, яка використовується системою для збереження інформації та виступає у ролі стандартного сховища даних.

6 НАУКОВА НОВИЗНА

У магістерській роботі розглянуто загальну архітектуру платформ для бізнес-аналізу, розроблено нову архітектуру зважаючи на необхідність інтеграції з BI, розроблено прототип семантичної системи для задач бізнес-аналітики.

Метою розробки є програмна реалізація застосування технологій семантичного веб для вирішення задач BI.

Об'єктом дослідження є системи аналізу бізнес-даних.

Предметом дослідження бізнес аналіз та семантичні технології в сучасних IT та способи їх взаємодії.

Методи дослідження базуються на методах теорії кодування, паттернах проектування, методологіях розробки програмного забезпечення та технологіях взаємодії клієнт-серверних додатках, розробках BI-систем.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– розроблено концепцію побудови системи яка об'єднує технології семантичного вебу та задач бізнес аналізу;

– сформульовано основні засоби інтеграцій технологій SW;

– система може стати альтернативою сучасним BI-системам.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

7.1 Техніко-економічне обґрунтування теми дипломного проекту

Після ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 60 днів (три місяці).

В магістерській роботі було проведене дослідження та виконана програмна реалізація застосування технологій семантичного веб для вирішення задач бізнес-аналітики.

Розроблене програмне забезпечення має достатню надійність і задовольняє усім поставленим умовам, а саме:

а) невеликий розмір;

б) невеликі системні потреби;

в) незалежність від встановлених на комп’ютері баз даних;

г) зручність у користуванні та надійність.

Таблиця 7.1 – Початкові дані

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт.	N	1
2. Кількість екземплярів програм, шт.	Ne	99
3. Запланований термін розробки, днів	Fpq	60 (3 місяці)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2

Продовження таблиці 7.1

1	2	3
7. Кількість макетів вхідної інформації	–	3
8. Кількість форм вихідної інформації.	–	4
9. Мова програмування (1-6)	–	2
10. Попередній досвід (1-6)	–	3
11. Гнучкість проекту ПП (1-6)	–	3
12. Детальність проекту ПП (1-6)	–	2
13. Рівень спрацьованості колективу (1-6)	–	2
14. Ступінь вимірності процесів (1-6)	–	3
15. Необхідна надійність програмного забезпечення (1-6)	–	2
16. Розмір бази даних (порівняно з розміром програми) (1-6)	–	2
17. Складність кінцевого програмного продукту (1-6)	–	2
18. Необхідний рівень забезпечення повторного використання (1-6)	–	2
19. Документованість відповідно до планованого життєвого циклу (1-6)	–	2
20. Вимоги до швидкодії ПП (1-6)	–	2
21. Обмеження на розміри основного сховища даних (1-6)	–	2
22. Різноманітність використовуваних обчислювальних платформ (1-6)	–	2
23. Професійний рівень аналітиків (1-6)	–	2
24. Професійний рівень програмістів (1-6)	–	2
25. Постійність складу команди розробників (1-6)	–	2
26. Досвід розробки додатків (1-6)	–	2
27. Досвід роботи з обчислювальною платформою (1-6)	–	2

Продовження таблиці 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПЗ для декількох серверів одночасно (1-6)	–	2
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн.	–	100000
33. Норматив додаткової зарплати, % :	$Н_д$	10
34. Норматив відрахувань у соціальні фонди, %	Нс	22
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	55
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

Значення трудомісткості розробки програмного забезпечення для стадій ТЗ, ЕК, ТП та ВП визначаємо по типовим нормам часу приведеним в додатках МВ. Стадія РП є найбільш тривалою і трудомісткою, що робить значний вплив на інші стадії проекту.

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

$$T_{ном} = A \, \text{Size}^B, \tag{7.1}$$

де: A – коефіцієнт Боема, $A = 2{,}45$;

Size – загальний об'єм відлагодженого програмного коду, тис. рядків;

B – показник ступеня, що визначається співвідношенням:

$$B = 1{,}01 + 0{,}001\sum W_i\,, \qquad (7.2)$$

де: *Wi* – сумарне значення п'яти показників (МВ, додаток 2), що відображають особливості розробки проекту програмного продукту (ПП) і колективу розробників.

$$B = 1{,}01+0{,}001(2{,}43+3{,}64+3{,}38+3{,}95+2{,}73) = 1{,}027.$$

$$T_{ном} = 2{,}45\cdot 2{,}7^{1{,}026} = 6{,}78 \text{ люд-міс.}$$

Визначаємо уточнені (з урахуванням приведених в МВ додатку 3 сімнадцяти додаткових коефіцієнтів) трудовитрати, люд-міс.:

$$T_{уточн} = T_{ном}\, ПV_j, \qquad (7.3)$$

де: $ПV_j$ – добуток сімнадцяти додаткових коефіцієнтів, приведених в МВ додатку 3.

$$T_{уточн} = 6{,}78\cdot(0{,}88\cdot 0{,}93\cdot 0{,}88\cdot 0{,}91\cdot 0{,}95\cdot 1\cdot 1\cdot 0{,}87\cdot 1{,}22\cdot 1{,}16\cdot 1{,}1\cdot 1{,}1\cdot$$
$$\cdot 1{,}12\cdot 1{,}1\cdot 1\cdot 1{,}1\cdot 1{,}1) = 9{,}37 \text{ люд-міс.}$$

Ці коефіцієнти дозволяють диференційовано оцінювати результати роботи програмістів, беручи до уваги швидкодію програми, використання різноманітних обчислювальних платформ і інструментів розробки, взаємодію декількох серверів, вимоги до об'ємів баз даних і ін.

Визначаємо підсумкові трудовитрати по стадії робочий проект, люд-дні:

$$T_{РП} = 0{,}3CT_{уточн}^{0{,}33+0{,}2(B-1{,}01)}S\,, \qquad (7.4)$$

де: *C* – визначений емпірично коефіцієнт, запропонований авторами методики, (МВ, додаток 4);

S – коефіцієнт стиснення (або подовження) графіка робіт %, що дозволяє коректувати терміни розробки ПЗ згідно встановленим вимогам. Вибираємо в межах (25…350)%.

$$T_{РП} = 0{,}3\cdot 2{,}66\cdot 9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)}\cdot 33 = 56 \text{ люд/день.}$$

Для зручності визначення загальної трудомісткості на розробку програмного забезпечення результати розрахунків по стадіям зводимо до таблиці 7.2.

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

Стадії розробки	Трудомісткість за типовими нормами та розрахунками	
	Величина, люд/дні	Підстава
Технічне завдання	9	Д5
Ескізний проект	10	Д6
Технічний проект	9	Д7
Робочий проект	56	Ф 7.1-7.4
Впровадження	13	Д13
Всього	97	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

Чисельність ставок інженерів-програмістів для розробки програмного забезпечення визначається за формулою:

$$Ч = \frac{T_{пз} N}{F_{pq} - H_{ев}}, \quad (7.5)$$

де: F_{pq} – плановий фонд робочого часу одного спеціаліста, днів;

$T_{пз}$ – трудомісткість розробки програмного забезпечення люд-дні.

$$Ч = \frac{97 \cdot 1}{60 - 5} = 1{,}8 \text{ ставки.}$$

Чисельність інженерів-електронщиків для проведення технічного обслуговування та ремонту комп'ютерних мереж визначається в залежності від наявності технічних засобів і норм витрат часу на виконання профілактичних робіт на протязі року.

Визначаємо затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за період розробки. Результати розрахунку зводимо до таблиці 7.3.

Таблиця 7.3 – Затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за розрахунковий період

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість облад-нання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	90	5	450	7,5
Монітор	60	5	300	5
Клавіатура	30	5	150	2,5
Маніпулятор «мишка»	30	5	150	2,5
Принтер матричний	60	0	0	0,0
Принтер лазерний	120	1	120	2
Принтер струминний	60	1	60	1
Сканер	20	1	20	0,33
Концентратор–маршрутизатор	30	2	60	1
Кабельні господарства ЛВС на 1 м. п.	2,5	200	500	8,33
Копіювальний апарат	140	1	140	2,33
Усього за рік:			$З_ч$	32,49

Час на профілактику обладнання в загальному балансі робочого часу інженерів-електронщиків неповинен складати більше 10%.

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

$$Ф_{др}^{с} = \frac{З_ч \cdot n_{міс}}{1,2}, \quad (7.6)$$

$$Ф_{др}^{с} = \frac{32,49 \cdot 3}{1,2} = 81,23 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

$$Ч_{ел} = \frac{Ф^{с}_{др}}{F_{др} \cdot T_{зм}}, \quad (7.7)$$

$Ч_{ел}$ = 81,23/(60·8) = 0,2 ставки.

Для забезпечення нормального технічного обслуговування засобів ТО та мереж, необхідно прийняти найбільше ціле значення розрахункової чисельності інженерів-електронщиків.

Чисельність інженерів-системотехніків, адміністраторів мережі, дизайнерів WEB вузлів, системних програмістів (аналітиків), бухгалтерів-економістів визначається за потребою в залежності від функціональних обов'язків. Після визначення чисельності персоналу складається штатний розклад.

Таблиця 7.4 – Розрахунок чисельності штатного персоналу сектору системного та адміністративного обслуговування засобів ОТ та комп'ютерних мереж

Посада	Вид роботи	Час	К-ть штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, доменного контролеру Windows Server, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN, PPPoE, Frame Relay, Wi-Fi	2	0,5
	Налаштування і конфігурування базової станції безпровідного зв'язку (CMTS)	0,5	
	Розробка та впровадження проектів з організації зв'язку між віддаленими об'єктами, ЛОМ	0,5	
	Забезпечення цілодобової роботи зв'язку клієнтів до мережі Інтернет	1	
Всього		4	

Продовження таблиці 7.4

Посада	Вид роботи	Час	К-ть штатних одиниць
Продакт-менеджер	Презентації нової продукції, пошук каналів збуту	1	0,25
	Підтримка постійних клієнтів	0,5	
	Оформлення договорів, ведення тендерів	0,25	
	Контроль взаєморозрахунків з постачальниками	0,25	
Всього		2	
Дизайнер WEB	Розробка концепції оформлення та інтерфейсу сайту, оптимізація дизайну існуючих, проектує їх структуру та навігацію	1	0,25
	Створення графічних і стилістичних елементів сайту	0,5	
	Оформлення банерів і промо-сторінок	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	
Інженер верстальник	Розробка та верстка макетів рекламної продукції та технічної документації	1	0,25
	Верстка друкованих видань	0,5	
	Додрукова підготовка макетів	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	

Складемо штатний розклад виконавців.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньомісячний оклад, грн.	Всього за період розробки, грн.
Керівник (ІТ-менеджер)	1	13440	40320
Продакт-менеджер	0,25	12000	9000
Інженер-програміст	1,8	14000	75600
Інженер - електронщик	0,2	12000	7200
Інженер-системотехнік	0,25	12000	9000
Адміністратор мережі	0,5	12000	18000
Системний програміст	0,25	12000	9000
Дизайнер WEB	0,25	12000	9000
Інженер-верстальник	0,25	12000	9000
Бухгалтер-економіст	0,5	12000	18000
Всього за період розробки	$R_{сп}$ = 5,25	-	$Ф_{роб}$ = 204120

Розрахуємо середньоденну зарплату одного виконавця:

$$З_{сд} = \frac{Ф_{роб}}{R_{сп} F_{рq}}, \quad (7.8)$$

де: $Ф_{роб}$ – загальна сума зарплати за плановий період, грн.

$$З_{сд} = \frac{204120}{5,25 \cdot 60} = 648 \; грн.$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

Балансова вартість будівель визначається з урахуванням кількості робочих місць виконавців, питомої площі на одне робоче місце, та вартості одного квадратного метра виробничої площі:

$$Б_{уд} = R^1_{сп} S_у Ц_{пл}, \quad (7.9)$$

де: $R^1_{сп}$ – кількість робочих місць виконавців, шт. Приймаємо 8 робочих місць;

$S_у$ – питома площа на одне робоче місце, $м^2$;

$Ц_{пл}$ – вартість одного квадратного метра площі, грн.

Згідно даних ТОВ науково-дослідницького консалтингового підприємства «Пектораль» (м. Кіровоград) ціна одного квадратного метра площі новобудови, вік якої не перевищує 25 років, по місту складає 500…1600 $у.о./м^2$. Враховуючи, що курс складає 1 у.о. = 37 грн. приймаємо для розрахунку вартість одного метра квадратного рівною 29000 грн./м^2. На кожне робоче місце у середньому потрібно $8 м^2$. З урахуванням цього:

$$Б_{уд} = 8 \cdot 8 \cdot 29000 = 1858000 \text{ грн.}$$

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку вона складе: 185800 грн.

Балансова вартість інвентарю розраховується за нормою 3500 грн. на одне робоче місце. Тобто:

$$І_{нв} = R^1_{сп} \cdot Ц_м, \quad (7.10)$$

де: $Ц_м$ – ціна меблів для одного робочого місця, грн.

$$І_{нв} = 8 \cdot 3500 = 28000 \text{ грн.}$$

Балансова вартість обчислювальної техніки визначається по оптовим цінам постачальника з врахуванням витрат на транспортування.

Специфікація на обчислювальну техніку наведена в таблиці 7.7.

Дані по оптовій ціні на обладнання та комплектуючі вибирались по прайсу фірми Суперкомп за 12.11.23 – джерело https://supercomp.kiev.ua/.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		10947
Системний блок		7347
Процесор	INTEL Pentium G6405 1200, 2 ядра, 4 потоки, 4.1 GHz, TDP - 58 Вт, 14nm, Intel Smart Cache, 8 GT/s, Intel UHD Graphics 610, Comet Lake, BOX	-
Системна плата	ASUS PRIME H510M-K сокет - 1200, DDR4, 3200 MHz, LAN - 1 Гбіт/с, D-Sub (VGA), HDMI, 1 x M.2 2280, 4 x Sata 6.0 Gb/s, Micro-ATX	-
Відеокарта	Intel UHD Graphics 610	-
Жорсткий диск	SSD M.2 2280 512GB LEVEN (JP600PCIE512GB) Серія - JP600, 512 GB, 3D TLC NAND, M.2, PCI Express 3.0 x4	-
Оперативна пам’ять	DDR4 8GB 3200 MHz Fury Beast Black Kingston Fury (ex.HyperX) (KF432C16BB/8)	-
Блок живлення	Gamemax 500W (GM-500B) ATX 12V v2.3, 500 Вт, 20+4 pin, CPU - 4+4pin, GPU - 1x6 pin, SATA - 3, Peripheral - 2, +12V1 - 20A, 1x120 мм, 150 x 140 x 86 мм	-
Корпус	Vinga CS210B, Miditower, ATX, Micro - ATX, Mini - ITX	-

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Кардрідер внутрішній	USB 2.0 Card reader STORM CR-35U1A4-B, int. 3.5", 1*USB2.0+AUDIO+1394, multi: All Type Cards, black	220
інше	Клавіатура, мишка	Подару-нок
Монітор	22" LG 22MP58VQ-P 5 мс IPS 1920x1080 250/1000M:1 178/178 D-Sub+HDMI+DVI	3600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USB cable	5500
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.7 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп’ютери	15	10947	16420,5	180625,5
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	199177

Витрати на транспорт, монтаж та випробування можуть бути прийняті в межах до 10% від оптової ціни.

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відраху-вання, грн.
1	2	3	4
Група 3			
1. Будівлі	1858000	-	-
2. Передавальні пристрої	185800	-	-
Всього по групі	2043800	5	102190
Група 4			
3. Обчислювальна техніка	199177	-	-
Всього по групі	199177	50	99588,5
4. Нематеріальні активи	100000	10	10000
Група 5, 6			
5. Вимірювальні пристрої	9031	25	2257,75
6. Транспортні засоби	143000	20	28600
7. Господарський інвентар	28000	25	7000
Всього по групі	180031	-	37857,75
Разом	$К_р$ = 2523008		$А_р$ = 249636,25

Примітка: вартість автомобіля Sens (Standard+) взята по даним з автосалону «Кіровоград-Авто», джерело http://kirovograd-avto.ukravto.ua /catalog/tm-9/model-80/description, складає 143000 грн.

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців:

$$З_о = \frac{З_{сд} \cdot Т_{пз}}{N_е}, \quad (7.11)$$

де: $N_е$ – кількість екземплярів програм, шт.

$$З_о = 648 \cdot 97/99 = 635 \text{ грн.}$$

Визначимо додаткову зарплату (оплата відпусток, виконання державних та суспільних обов'язків) на рівні 10%:

$$З_д = З_о \cdot Н_q \cdot 0{,}01, \quad (7.12)$$

де: $Н_q$ – норматив додаткової зарплати, %.

$$З_д = 635 \cdot 10 \cdot 0{,}01 = 64 \text{ грн.}$$

Відрахування на соціальні потреби за нормативом $Н_с = 22\%$ від суми основної та додаткової зарплати:

$$С_{оц} = 0{,}01 \cdot Н_с (З_о + З_д), \quad (7.13)$$

де: $Н_с$ – відрахування на соціальні потреби, %.

$$С_{оц} = 0{,}01 \cdot 22(635+64) = 154 \text{ грн.}$$

Визначимо загальногосподарські витрати (електроенергію, ремонт і утримання приміщень і т.д) за нормативом $Н_г = 15\%$ від основної зарплати:

$$Г_{осп} = З_о \cdot Н_г \cdot 0{,}01, \quad (7.14)$$

де: $Н_г$ – загальногосподарські витрати, %.

$$Г_{осп} = 635 \cdot 15 \cdot 0{,}01 = 95 \text{ грн.}$$

Визначимо витрати на матеріали для розробки програмної продукції за нормами споживання та діючими цінами за одиницю виміру:

$$З_М = (З_{М1} + З_{М2} + З_{М3})/N_е, \quad (7.15)$$

де: $З_{М1}$ – вартість паперу, грн.;

$З_{М2}$ – вартість запам'ятовуючих пристроїв, грн.;

$З_{М3}$ – вартість фарби, картриджів, тонеру, грн.;

N_e – кількість екземплярів програм, шт.

Згідно виданих норм приймаємо 1/6 пачку паперу на місяць розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п = 206$ грн., визначаємо вартість паперу за період розробки $N_м = 3$ міс:

$$З_{М1} = Ц_п \cdot N_м. \quad (7.16)$$

$$З_{М1} = 206 \cdot 3 \cdot 1/6 = 309 \text{ грн.}$$

Згідно виданих норм до вартості запам'ятовуючих пристроїв входить вартість CD/DVD дисків в кількості 45 примірників:

$$З_{М2} = \sum Ц_д , \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR TDK 700Mb, 80Min, 52x Cake box – 27 грн./шт., DVD-R LG 4,7Gb, 16x speed Cake box – 27 грн./шт.

$$З_{М2} = 27 \cdot 45 = 1215 \text{ грн.}$$

Згідно виданих норм одноразовій заправці підлягають усі друкуючі пристрої і становить:

$$З_{М3} = \sum Ц_з., \quad (7.18)$$

де: $Ц_з$ – вартість розхідних матеріалів друкуючих пристроїв: відновлення та заправка картриджу для Canon i-SENSYS LBP6030W – 574 грн.; картридж для Epson Stylus Photo P50 – 558 грн.; відновлення картриджу для MF217W – 570 грн.

$$З_{М3} = 574+558+570 = 1702 \text{ грн.}$$

$$З_М = (309 + 1215 + 1702)/99 = 33 \text{ грн.}$$

Визначимо витрати на освоєння нових мов програмування або операційних систем за нормативом ($Н_п = 15\%$) від основної зарплати виконавців:

$$О_п = З_о \cdot Н_п \cdot 0{,}01, \quad (7.19)$$

де: $Н_п$ – норматив витрат на освоєння нових мов програмування, %.

$$О_п = 635 \cdot 15 \cdot 0{,}01 = 95 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_e = 99$ прим.):

$$A_м = \frac{A_р \cdot N_{міс}}{N_е \cdot 12}, \quad (7.20)$$

де: $A_р$ – загальна річна сума амортизаційних відрахувань, грн.

$$A_м = 249636 \cdot 3/(99 \cdot 12) = 631 \text{ грн.}$$

Величини ціна підприємства, податок на додану вартість, відпускна ціна програмної продукції визначаються за формулами, приведеними в таблиці 7.9

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн.
1	2	3
1. Основна зарплата виконавців	$З_о$	635
2. Додаткова зарплата виконавців	$З_д$	64
3. Відрахування на соціальні потреби	$С_{оц}$	154
4. Загальногосподарські витрати	$Г_{осп}$	95
5. Витрати на матеріали	$З_М$	33
6. Освоєння нових операційних систем, мов програмування	$О_п$	95
7. Амортизація основних фондів	$А_м$	631
8. Повна собівартість програмного забезпечення	$С_п$	1707
9. Плановий прибуток	$П_р$	940
10.Ціна підприємства $Ц_п = С_п + П_р$	$Ц_п$	2647
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	529,4
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	3176,4

Повна собівартість ПЗ визначається як сума витрат за попередніми статтями калькуляції:

$$С_п = З_о + З_д + С_{оц} + Г_{осп} + З_м + О_п + А_м. \quad (7.21)$$

$$C_n = 635+64+154+95+33+95+631 = 1707 \text{ грн.}$$

Визначимо плановий прибуток за рівнем рентабельності (P_n) програмної продукції, яка залежить від складності програми та ступеня новизни задачі.

Для даного програмного забезпечення рівень рентабельності складає 55%.

$$П_p = 0{,}01 \cdot P_n \cdot C_n, \quad (7.22)$$

де: P_n – рівень рентабельності, %.

$$П_p = 0{,}01 \cdot 55 \cdot 1707 = 940 \text{ грн.}$$

7.6 Визначення об'єму капітальних вкладень у споживача програмної продукції

Об'єм капітальних вкладень у споживача програмної продукції визначаємо на основі балансової вартості основних фондів, яка враховує ціну, транспортно-заготівельні витрати, вартість будівель, монтажних та пусконалагоджувальних робіт, а також витрати на випробування у виробничих умовах. Результати розрахунків зводимо у таблицю 7.9.

Таблиця 7.10 – Розрахунок об'єму капітальних вкладень у споживача програмної продукції

Найменування капітальних вкладень	Сума за варіантами, грн.	
	Базовий	Новий
Вартість програмної продукції	–	3176
Всього капітальних витрат	–	3176

7.7 Визначення експлуатаційних витрат

Експлуатаційні витрати у споживача програмної продукції визначаємо при умові роботи підсистеми на протязі року. Результати зводимо до таблиці 7.11.

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на технічне обслуговування	$З_р$	59048	33550
2. Витрати на електроенергію	$З_{ел}$	753	428
3. Витрати на амортизацію	$З_{ам}$	0	794
Всього витрат за рік	$І$	59801	34772

Витрати на обслуговування системи:

$$З_р = Т_р \cdot З_г \cdot (1 + 0{,}01 \cdot Н_q) \cdot (1 + 0{,}01 \cdot Н_с), \qquad (7.23)$$

де: $Т_р$ – кількість годин обслуговування системи на рік, *год.*;

$З_г$ – заробітна плата обслуговуючого персоналу, *грн / год*.

Після купівлі нового програмного забезпечення кількість годин на технічне зменшилася з 440 годин на рік до 250 годин на рік, тому витрати на технічне обслуговування зменшилися з:

$$З_{р\,баз} = 440 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 59048 \text{ грн},$$

до:

$$З_{р\,нов} = 250 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 33550 \text{ грн}.$$

Витрати на електроенергію визначаються з урахуванням спожитої потужності ($П_{ел}$) в кіловатах, часу експлуатації технічних засобів ($Т_р$) в годинах та ціни однієї кіловат-години ($Ц_{ел}$):

$$З_{ел} = П_{ел} \cdot Т_р \cdot Ц_{ел}. \qquad (7.24)$$

$$З_{ел\,баз} = 0{,}45 \cdot 440 \cdot 3{,}8 = 753 \text{ грн}.$$

$$З_{ел\,нов} = 0{,}45 \cdot 250 \cdot 3{,}8 = 428 \text{ грн}.$$

Витрати по амортизації визначаються на основі норм амортизаційних відрахувань, вартості програмної продукції і основних фондів. Для розрахунку складаємо таблицю 7.12.

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн., за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	25	–	3176	–	794
Всього відрахувань	-	–	3176	–	794

7.8 Визначення економічної ефективності програмної продукції

Економічна ефективність програмного забезпечення визначається для виготовлювача і споживача за такими показниками.

Величина економічного ефекту при виготовленні програмної продукції, розраховуємо за формулою:

$$E_в = (Ц_п - C_п) \cdot N_е - \sum_{i=1}^{m} E_{p_m} \cdot K_{p_m}, \quad (7.25)$$

де: $K_р$ – балансова вартість основних фондів розробника, грн.; $E_р$ – розрахунковий коефіцієнт капіталовкладень.

$$Ев = (2647\text{-}1707) \cdot 99\text{-}(0{,}05 \cdot 2043800 + 0{,}5 \cdot 199177 + 0{,}25 \cdot 37031 +$$
$$+0{,}1 \cdot 100000 + 0{,}2 \cdot 143000) \cdot 3/12 = 30651 \; грн.$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

$$T_в = \frac{K_p^*}{(Ц_п - C_п) \cdot N_е}, \quad (7.26)$$

де: K_p^* – балансова вартість основних фондів розробника без врахування вартості ОФ третьої групи, так як їх строк служби на порядок більший ніж період розробки ПЗ.

$$T_в = \frac{479208}{(2647\text{-}1707) \cdot 99 \cdot 12 / 3} = 1{,}29 \; років.$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	99
2. Повна собівартість розробленої програми	Грн.	1707
3. Ціна розробленої програми	Грн.	2647
4. Плановий прибуток від реалізації розробленої програми	Грн.	940
5. Рентабельність програмної продукції	%	55
6. Об’єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	2523008
7. Загальний прибуток від реалізації програмної продукції	Грн.	93060
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	30651
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років	1,29
10.Об’єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	3176
11.Величина економічного ефекту у користувача програмної продукції	Грн.	24235
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,13

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

$$E_{сп} = (I_б - I_н) - E_н(K_н - K_б), \qquad (7.27)$$

де: $I_б$, $I_н$ – величина експлуатаційних витрат за базовим и новим варіантом відповідно;

$K_б$, $K_н$ – об’єм капітальних вкладень за варіантами, що порівнюються.

$$E_{сп} = (59801\text{-}34772)\text{-}0{,}25 \cdot 3176 = 24235 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у споживача програмної продукції за рахунок зниження експлуатаційних витрат:

$$T_{сп} = \frac{K_н - K_б}{I_б - I_н}, \tag{7.28}$$

$$T_{сп} = \frac{3176}{59801 - 34772} = 0{,}13 \ року .$$

7.9 Висновки

Розроблена програма економічно вигідна. За рахунок впровадження програмного забезпечення досягається скорочення часу обробки інформації, підвищується культура праці, підвищення якості приймаючих управлінських рішень.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Аналіз умов праці програміста

Інтернет відіграє важливу роль у житті сучасної людини. Кожного дня мільйони людей використовують Інтернет для пошуку необхідної інформації, спілкуванні у соціальних мережах, перегляду новин. Багато людей користуються Інтернетом у професійних цілях, оскільки завдяки Інтернету зявилося багато нових професій. Тому для веб-розробника так важливо розробити зручний інтерфейс для зручного сприйняття інформації, та необхідний функціонал, який буде відповідати необхідним вимогам та навантаженням. Все це вимагає багато багато часу та великого навантаження з боку розробників.

Тому так важливо слідкувати за умовами праці, в яких відбувається робочий процес. Оскільки захворювання можуть бути спричинені надмірним фізичним або розумовим навантаженням, через велику нервово-емоційну напругу, або через виробниче середовище. В даному розділі магістерської роботи проведемо аналіз основних чинників при роботі програміста.

При роботі за компютером, розробник має велике зорове навантаження, тому йому необхідне належне освітлення приміщення. Якщо в приміщенні недостатньо природного освітлення, потрібно використовувати спеціальні світильники. Також оскільки розробник значний час працює з електричними приборами є можливість, бути ураженим електричним струмом, тому потрібно дотримуватись всіх необхідних норм. Серед основних чиників, які впливають на розробників під час трудової діяльності можна виділити[46]:

1. Рівень освітлення в приміщенні.
2. Температура, вологість в приміщенні.
3. Рівень шуму на робочому місці.

4 .Напруга в електричному ланцюзі, електричні показники.

Розберемо кожний з чиників окремо, проаналізуємо які повинні бути стандарти кожного з чиників, відповідно до правил з охорони праці.

Рівень штучного освітлення

Головним документом для встановлення норм необхідних показників освітлення є ДБН В.2.5-28:2018 «Природне та штучне освітлення» [47].

Сьогодні найбільш розповсюдженішими є світлодіодні ламп. В середньому світловіддача від таких ламп знаходиться на рівні 80-120 Лм/Вт [48]. Джерелом живлення прийнято вважати електричну мережу у 220В. А освітленіть робочого приміщення повина бути Е =300-500 Лк, оскільки робота програміста відноситься до робіт середньої точності з присвоєнням розряду зорових робіт IV[49].

Рівень сітла повинен бути достатнім, щоб працівник міг працювати без навантаження на зір. Це залежить від системи освітлення, кількості світильниув, їх типу та розміщення у приміщенні. Допустиме значення освітленості робочої поверхні приймається Е = 400 лк [50].

Для покращення освітлення комп'ютерній лабораторії будуть використовуватися світлодіодні лампи, а саме FL-LED T8-900 світловий потік яких F=1500лм.

Мікроклімат робочої зони: температура, відносна вологості, швидкість руху повітря

Головним документом для встановлення норм мікроклімату робочої зони є ДСН 3.3. 6.042 -99 «Державні санітарні норми мікроклімату виробничих приміщень» [51].

Праця програміста за важкістю відноситься до легкої фізичної роботи категорії Іа [49]. Де вказано, що в приміщенні, я кому знаходиться компютерне обладнення, повнині бути встановлені певні норми, оскільки офісна техніка є джерелом тепловиділень, що може спричинити підвищення температури. Серед

потимальних параметрів для роботи встановлена температура 23 – 25^{0}С і вологість на рівні 40 – 60% в залежності ві періоду року.

Для підтримки комфортної температури можна використовувати як організаційні методи, наприклад розпорядок дня, так і технічне обладнення, наприклад кондиціонери, вентіляцію. Як правило в холодний період часу використовуються додаткове опалення для підтримання комфортної температури, а в літку встановлюються кондеціонери.

Рівень шуму на робочому місці

Як правило при використанні великої кількості компютерів в одному приміщенні, через гудіння, рівень шуму має значення більше норми. Допустима норма становить менше 50 дБ [52].

Гучний шум негативно впливає на умови праці та організм людини. Якщо шум триває тривалий час цу може спричинии головні болі, біль у вухах, підвищення стомлюваності, зниження концентрації та уваги. Такі симптоми можуть викликати стресові ситуації у людини. Все це шкодить продуктивності працініка та його стану здоровья.

Щоб встановити необхідний рівень шуму, використовують додаткову звукоізоляцію. Для цього найчатіше використовуються мати та плити із скляного та мінерального волокна, м'які плити з деревних стружок, картон, гуму, утеплений лінолеум, а також заміна вікон на звукоізолюючі.

8.2 Заходи профілактики при роботі з комп’ютерною технікою

Санітарно-гігієнічні норми є важливим критерієм при роботі в приміщенні. Від них залежить здоровє працівників, їх рівнень працездатності, втомлюваність. Щоб всього цього уникнути потрібно стежити за нормами на робочому місці.

Якщо говорити про електробезпеку в приміщенні, то в приміщенні необхідно устаткування розподільних щитів спеціальними розетками з

заземлюючими контактами, повині бути заземлені всі прилади і пристрої, час від часу повина проводитися перевірка всіх приладів, щорічна здача іспитів з охорони праці.

Для оптимальних показників мікроклімату та освітленості потрібні використовувати дефлектор, для організації вентиляції, та повітрообміну. Та перевірку освітленості в приміщенні згідно відділом охорони праці, щоб відповідати номам для зорової роботи .

Ще однією проблемою з якою часто зустрічаються програмісти є мала рухливість та повний сидячий робочий день. Тому рекомендується час від часу робити невеликі перерви, під час обіднього перериву вживати їжу не на робочому місці. У окремих випадках, коли при дотриманні всіх санітарних норм, працівник все одно себе погано почуває, дозволяється індивідуальний підхід для обмеження роботи з обчислювальними пристроями. Тривалість роботи за компютером не повина безперервно триввати більше 4 годин.

Для зменшення зорового та нервово-емоційного навантаженняє, та поліпшення мозкової діяльності рекомендується робити перерви для психологічного та фізичного розвантаження.

В приміщеннях також поині бути протипожежне обладнення, та інструкція у разі надзвичайних ситуаціх. Повина бути особа, яка відповідає за пожежну безпеку, перевіряє обладнення, та системи протипожежного захисту а також щорічне проведення інструктажів серед працивників.

Автоматична пожежна сигналізація повина відповідати вимогам ДБН ДБН В.2.5-56:2014, яке вимагає використання вогнестійких кабелів та автоматичну роботу системи оповіщення та евакуації людей у випадку надзвичайної ситуації [53].

При перевірці, приміщення повино відповідати всім нормам пожежної безпеки. Це виконується за допомогою перевірки пожежної охорони та техніки, проведенню інструктажу і своєчасне інформування пожежної охорони про несправність пожежної техніки, впровадження систем протипожежного

захисту. Організаційні та технічні заходи, спрямовані на попередження виникнення пожежі, обмеження поширення вогню та успішної евакуації людей.

8.3 Розрахнок занулення глухозаземленої нейтралі

Занулення, як основний засіб захисту, застосовується в електроустановках до 1 кВ з глухозаземленою нейтраллю трансформатора або генератора [54]. Початкові дані для розрахунку занулення глухозаземленої нейтралі трансформатора виробничого приміщення:

Загальна потужність: P = 5 кВт.

Кількість електродвигунів: m = 10

Потужність освітлювальних приладів: Po = 3 кВт.

Довжина магістрального кабеля: LM =70 м.

Довжина розгалудження: l=22м.

Лінійна напруга U=380 В.

Фазна напруга Uф=220 В.

Визначаємо силу номінального струму електроустановки:

$$Iном = Imax = (P * 1000) / (\sqrt{3} * Uл * cosф) \quad (8.1)$$

$$Iном = Imax = (5 * 1000)/(\sqrt{3} * 380 * 0,85) = 8,9 А$$

де: P − номінальна сумарна потужність електроприладів, кВт;

Uл − лінійна напруга, В;

cosф − коефіцієнт потужності, приймається в залежності від типу електрообладнання в межах 0,8..0,87.

Визначаємо силу пускового струму електродвигуна:

$$Iпус = 5 * Iном \quad (8.2)$$

$$Іпус = 5 * 8,9 = 44,5 А$$

Визначаємо номінальну силу струму апарата захисту:

$$Ін = Іпус/b \quad (8.3)$$

$$Ін = 44,5 /2,5 = 17,8 А$$

b − коефіцієнт пуску електродвигуна − для легких умов пуску − 2,5..3.

Вибираємо запобіжник ПН 2-100 з плавкою вставкою Іном = 50 А. Визначаємо найменше допустиме по умовам спрацьовування захисту значення сили струмукороткого замикання:

$$Іктіп = Ін * К \quad (8.4)$$

$$Іктіп = 50 * 3 = 150 А$$

Ін−номінальний струм апарата захисту;

К − коефіцієнт надійності;

Знаходимо переріз провода або кабеля розгалуження з умови допустимого нагрівання:

$$Ідоп = Івст/а \quad (8.5)$$

$$Ідоп = 50 / 3 = 16,6 А$$

Вибираємо площу перерізу 10 мм^2 (Sф) при числі проводів і = 4 розташований у повітрі. Визначаємо максимальний робочий струм:

$$Iроб = Ко(Кз*(Р*1000)/(\sqrt{3}*Uл*cosф)*m+Кз*(Ро*1000)/(\sqrt{3}*Uл*cosф)) \quad (8.6)$$

$$Iроб = 0,75*(0,85*((5*1000)/(1,73*380*0,85)*10+(3*1000)/(1,73*380*0,85))=60,4 \text{ А}$$

Ко − коефіцієнт одночасності роботи групи електроприймачів;

Ко = 0.7...0.8; Кз = 0.8...0.9;

Кз – коефіцієнт завантажених електродвигунів;

Ро =3кВт − потужність освітлювальної мережі;

Визначається струм короткочасного перевантаження магістрального кабеля:

$$Iпер = Ко(Кз*(Р*1000)/(\sqrt{3}*Uл*cosф)*n+Кз*(Ро*1000)/(\sqrt{3}*Uл*cosф))+Iпус \quad (8.7)$$

$$Iпер = 0,75*(0,85 *(5*1000)/(1,73*380*0,85)*9+0,85*(30*1000)/(1,73*380 * 0,85)) + 44,5 = 130,0643 \text{ А}$$

Струм спрацювання електромагнітного розчеплювача додатково перевіряємо по максимальному струму перевантаження лінії:

$$Iспр >= 1,25 * Iпер \quad (8.7)$$

$$Iспр >=1,25 * 130,0643 = 162,5803 \text{ А}$$

Приймаємо Iспр = 162,5803 А. Вимикач : А3714Б.

Вибираємо площу перерізу Sф магістрального кабеля (провідника) по Iдоп. Sф = 70mm^2 , − кабель АВРГ прокладений в землі, і=3 (число проводів). Вибрану площу перерізу перевіряємо для автоматів з електромагнітним розчеплювачем:

$$Iдоп >= Iспр/4,5 \quad (8.8)$$

$$Iдоп >= 162/4,5 = 36 А$$

Проводимо узгодження з номінальним струмом автомата:

$$Iдоп = Iспр/3 \quad (8.9)$$

$$Iдоп = 162/3 = 54 А$$

Значення 36 і 54 А. менше ніж Imax= 60,4 А., значить площа перерізу кабеля вибрана вірно. Визначаємо потужність трансформатора:

$$Nтр = ((Кп * Рном)/cosф) \quad (8.10)$$

$$Nтр = (0,7 * 53)/0,8 = 46,375 кВ*А$$

Рном − сумарна потужність електроприймачів, кВт;

Cosф − середній коефіцієнт потужності електроприймачів (0,8);

Кп − коефіцієнт попиту (0,7);

Одержане значення потужності трансформатора округляємо до ближчого стандартного значення. Визначаємо опір трансформатора Z_T. Вибираємо трансформатор на 40 кВА (Z_T =0,562 Ом). визначаємо орієнтовно площу перерізу провідника. Для магістрального кабеля:

$$Sn_1 >= 0,5 * Sф \quad (8.11)$$

$$Sn_1 >= 0,5 * Sф = 0,5*70=35 мм^2$$

Визначаємо для розгалуження:

$$Sn_2 >= 0{,}5*10 = 5 \text{ мм}^2$$

Округляємо ці значення до найближчих більших 35 мм^2 (Sn1), і 6 мм^2 (Sn2). Визначаємо активний і індуктивний опір фазного і нульового захисного провідників на дільницях 1 і 2:

$$Rф = p * (Lм/Sф_1) + p * (L/Sф_2) \quad (8.12)$$

$$Rф = 0{,}028 * (70/70) + 0{,}028 * (22/10) = 0{,}0896 \text{ Ом}$$

$$Rn = p * (Lm/Sn_1) + p * (L/Sn_2) \quad (8.13)$$

$$Rn = 0{,}028 * (70/35) + 0{,}028 * (22/6) = 0{,}1586 \text{ Ом}$$

Для окремо проложених нульових провідників иого приимають рівним 0,6 Ом/км. При прокладці кабелем, або в сталєвих трубах індуктивним опором нехтують.

Знаходимо дійсне значення (модуль) струма однофазного короткого замикання:

$$Iкр = Uф/((Zт/3) + \sqrt{((Rф + Rn)^2 + (Xф + Xn + X'n^2))} \quad (8.14)$$

$$Iкр = 220/((0{,}562/3) + \sqrt{(0{,}0896 + 0{,}1586)^2}) = 418{,}18 \text{ А}$$

Визначення максимальноі напруги на корпусі обладнання відносно землі при замиканні фази на корпус.

$$Uктах = Iкр * Zн \quad (8.15)$$

$$Uкmax = 418,18 * 0,1586 = 66,32 В$$

Z_H − повний опір нульового провідника.

Умова не виконується. Необхіно збільшити перерізи Sn 1 та Sn 2 до Sф1 та Sф2 і зробити перерахунок:

$$Rn = 0,028 * (70/70) + 0,028 * (22/10) = 0,0896 Ом,$$

$$Iкр = 220/((0,562/3) + \sqrt{(0,0896 + 0,0896)^2}) = 600,21 А,$$

$$Uкmax = 600,21 * 0,0896 = 53,77 В.$$

Умова не виконується, необхідно або замінити запобіжник з плавкою вставкою на автоматичний вимикач із струмовим реле, що дає можливість зменшити час замикання на копус і підвищити допустиму напругу на корпусі або застосувати повторне заземлення нульового захисного провідника. Повторне заземлення нульового захисного провідника:

$$Rn=(Uдоп * Ro)/((Iкр * Zн)−Uдоп) \quad (8.15)$$

$$Rn= 36 * 4/((600,21 * 0,0896) - 36) = 8,09 Ом.$$

8.4 Висновки

Існує багато факторів, які можуть вплинути на роботу розробника, через некомфортні або навіть небезпечні умови праці, які знаходяться в приміщенні. Серед основних причин виділяється: недостатній рівень світла, гучний шум, високий рівень навантаження, умови мікроклімату. Під час дослідження теми, були переглянуті можливі шкідливі та небезпечні ситуації, які можуть виникнути на робочому місці та спооби їх уникнення та ліквідації.

В результаті можна зробити висновки, які умови повині бути для продуктивної роботи працівника, як повино бути організоване приміщення і його робоче місце. За допомогою проведених обчислень, можна становити необхідні для комфортної роботи умови.

9 ОСНОВНІ ВИСНОВКИ

Магістерська робота присвячена дослідженню методам інтеграції технологій Semantic Web для Business Intelligence.

Досліджено сьогоднішні тенденції розвитку цих технологій. На сьогодні комп'ютери беруть досить обмежену участь у формуванні й обробці інформації в мережі Інтернет. Функції комп'ютерів в основному зводяться до збереження, відображення і пошуку інформації.

У той же час створення інформації, її оцінка, класифікація й актулізація — усе це як і раніше виконує людина. Як включити комп'ютер у ці процеси? Якщо комп'ютер поки не можна навчити розуміти людську мову, то потрібно використовувати мову, що була б зрозумілою комп'ютеру. Тобто, в ідеальному варіанті, вся інформація в Інтернеті повинна розміщуватись двома мовами: людською мовою для людини і комп'ютерною мовою для розуміння комп'ютера.

Семантична павутина — це концепція мережі, у якій кожен ресурс людською мовою був би доповнений описом, зрозумілим комп'ютеру. Було названо і проаналізовано ряд існуючих BI-систем, що найбільше сьогодні використовуються для вирішення бізнес задач. І було зроблено висновок, що вони всі є досить великими, зручними, та мають звісно багато переваг, проте всі вони є дуже складними та дорогими рішеннями, що перешкоджає інтеграцію для середнього та малого бізнесу.

Тому було вирішено створити власну BI-систему, що задовільняла б середній та малий бізнес.

В ході роботи було спроектовано систему, що задовільняє сучасним необхідностям бізнесу, використовує новітні технології, має логічну та масштабовану архітектуру.

Основою реалізованого прототипу є мова програмування Python, який відповідає за backend додатку(з допомогою веб-фреймворку Flask), обробку та

трансформацію даних, створення запитів до RDF-сховищ. В прототипі системі реалізовано наступний функціонал:

- додавання нових знань до локального RDF-сховища;
- створення та обробка SPARQL-запитів до локального сховища та віддаленого;
- передобробка отриманою інформації(видалення, редагування);
- візуалізація даних (на даний момент реалізовано графік та стовпчата діаграма).

В подальшому розвитку система перспективною є автоматизація процесів введення інформації, можливо з допомогою технологій machine learning.

Проте це не було метою роботи, тому залишимо це для майбутніх досліджень.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання.

Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі *24235* грн. Враховуючи вартість розробки та необхідне обладнення, строк окуплення становить 0,13 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Мова програмування JavaScript [Електронний ресурс] – Режим доступу до ресурсу: https://learn.javascript.ru/

2. Introduction to Node.js [Електронний ресурс] – Режим доступу до ресурсу:https://nodejs.dev/

3. Серверне програмування веб сайтів [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/ru/docs/Learn/Server-side

4. Node.js v14.0.0 Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://nodejs.org/api/

5. М. Кантелон , М. Хартер, Т. Головайчук, Н. Райлих // “Node.js в действии”.

6. Янг А., Мек Б., Кантелон М. // “Node.js в действии. 2-е издание”.

7. John Resig, Bear Bibeault, Josip Maras // “Secrets of theJavaScript Ninja”.

8. Express [Електронний ресурс] – Режим доступу до ресурсу: http://expressjs.com/

9. Фреймворк AngularJS [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/AngularJS

10. AngularJS [Електронний ресурс] – Режим доступу до ресурсу: https://angularjs.org/

11. Bootstrap [Електронний ресурс] – Режим доступу до ресурсу: https://getbootstrap.com/

12. React.js [Електронний ресурс] – Режим доступу до ресурсу: https://https://uk.reactjs.org/

13. AngularJS MVC [Електронний ресурс] – Режим доступу до ресурсу: https://www.tutorialspoint.com/angularjs/angularjs_mvc_architecture.htm

14. Vue.js [Електронний ресурс] – Режим доступу до ресурсу: https://vuejs.org/

15. Angular 2 [Електронний ресурс] – Режим доступу до ресурсу: https://angular.io/

16. Односторінковий застосунок [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Односторінковий_застосунок

17. Что такое MVC [Електронний ресурс] – Режим доступу до ресурсу: https://ru.hexlet.io/blog/posts/chto-takoe-mvc-rasskazyvaem-prostymi-slovami

18. Build Node.js Apps [Електронний ресурс] – Режим доступу до ресурсу: https://code.visualstudio.com/docs/nodejs/nodejs-tutorial

19. REST [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/REST

20. What is REST [Електронний ресурс] – Режим доступу до ресурсу: https://restfulapi.net/

21.Тренды веб-разработки [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/post/451572/

22.Веб додаток [Електронний ресурс] – Режим доступу до ресурсу: https://webcase.com.ua/blog/cho-takoe-web-prilozhenie-vse-vidy/

23.Мова розмітки гіпертексту [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/HTML

24. HTML [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/uk/docs/Web/HTML

25. Каскадні таблиці стилів [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/CSS

26. Стек MEAN [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/MEAN_(веброзробка)

27. MongoDB [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/MongoDB

28. Веб-технології для розробників [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/uk/docs/Web

29. CORS, XSS [Електронний ресурс] – Режим доступу до ресурсу: https://dev.to/maleta/cors-xss-and-csrf-with-examples-in-10-minutes-35k3

30. AJAX [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/AJAX

31. Fetch API [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/ru/docs/Web/API/Fetch_API

32. AngularJS Tutorial [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3schools.com/angular/default.asp

33. jQuery [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/JQuery

34. Основи Web-технологій [Електронний ресурс] – Режим доступу до ресурсу:https://pidruchniki.com/1243020547796/informatika/web-tehnologiyi_pidpriyemstvah

35. npm [Електронний ресурс] – Режим доступу до ресурсу: https://www.npmjs.com/

36. Install MongoDB [Електронний ресурс] – Режим доступу до ресурсу: https://docs.mongodb.com/manual/administration/install-on-linux/

37. Як встановити Node.js [Електронний ресурс] – Режим доступу до ресурсу: https://www.digitalocean.com/community/tutorials/node-js-ubuntu-18-04-ru

38. Linux [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Linux

39. npm-audit [Електронний ресурс] – Режим доступу до ресурсу: https://docs.npmjs.com/cli/audit

40. TypeScript [Електронний ресурс] – Режим доступу до ресурсу: https://www.typescriptlang.org/

41. SQL [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/SQL

42. MongoDB Compass [Електронний ресурс] – Режим доступу до ресурсу: https://www.mongodb.com/products/compass

43. Postman [Електронний ресурс] – Режим доступу до ресурсу: https://www.postman.com/

44. SQL [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/SQL

45. SPA (Single-page application) [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Glossary/SPA

46. Охорона праці [Електронний ресурс]: реферат – Режим доступу до ресурсу: https://revolution.allbest.ru/life/00468031_0.html

47. Природне і штучне освітлення ДБН В.2.5-28:2018: державні будівельні норми України [Електронний ресурс] / Ю. Громадський, С. Облакевич, М.Громадський, Г. Фаренюк, Є. Фаренюк, О. Підгорний, О. Сергейчук, Є.Рейцен, В. Єгорченков, Л. Коваль, Д. Радомцев, В. Злоба, Н. Кучеренко, Г.Кожушко, О. Гончар, О. Козенко, Б. Шабашкевіч, Ю. Добровольський, В.Акіменко, С. Гозак, А. Яригін, В. Назаренко, В. Мартиросова, В. Сорокін, Є.Пугачов - Київ 2018 - Режим доступу до ресурсу: https://ledeffect.com.ua/images/__branding/dbn2018.pdf

48. Розрахунок світлодіодного освітлення кімнати [Електронний ресурс] – Режим доступу до ресурсу: https://luxled.biz.ua/rozrahynok-svitlodoidnogo-osvitlennja-kimnatu-v-kvarturi-abo-bydunky

49. Охорона праці, охорона праці та безпека в надзвичайних ситуаціях : метод. вказ. до викон. розділів у дипломних роботах / [укл. В.М. Челябієва, О.Л. Гуменюк] - Чернігів ЧДТУ 2013 - [Електронний ресурс] – Режим доступу до ресурсу: http://ir.stu.cn.ua/bitstream/handle/123456789/12461/Охорона праці та безпека. в надзв. ситуац;метод.вказ..pdf?sequence=1&isAllowed=y

50. Освітленість робочих місць [Електронний ресурс] – Режим доступу до ресурсу:https://ua-referat.com/Освітленість_робочих_місць_сучасні_підходи_до_вимірів_і_оцінки

51. Температурний режим праці [Електронний ресурс] – Режим доступу до ресурсу: http://poltava.medprof.org.ua/poltava/zakhist-trudovikh-ta-socialno-

ekonomichnikh-prav-pracivnikiv-galuzi/pravova-dopomoga/temperaturnii-rezhim-praci-jakim-vin-maje-buti/

52. Шум. Методи захисту від його дії : метод. вказ. до лабораторної роботи / [укл. В. І. Шмирко, С. М. Журавель] — Запоріжжя: ЗНТУ, 2014. - 14 с. [Електронний ресурс] – Режим доступу до ресурсу: https://zp.edu.ua/sites/default/files/konf/oop_shum-2014.pdf

53. Системи протипожежного захисту ДБН В.2.5-56:2014 / Б. Платкевич, В. Носач, В. Федюк, В. Мусійчук, В. Євстіфєєв, Г. Дубінський, В. Сокол, А.Бушиленко, В. Дунюшкін, Р. Уханський, С. Пономарьов, В. Приймаченко, А.Приймаченко, С. Пітайчук, Н. Морозова, І. Колосов, О. Лагода, П. Мізін, В.Савченко, М. Федорович, П. Шаповалов, Л. Фесенко — Київ 2015 — [Електронний ресурс] – Режим доступу до ресурсу: http://kbu.org.ua/assets/app/documents/dbn2/98.1. ДБН В.2.5-56~2014. Системи протипожежного захисту.pdf

54. Охорона праці. Ч. 2. Занулення : метод. вказ. до викон. розрахунків з викор. персон. ЕОМ IBM−сумісного типу / [укл. О. В. Оришака, Є. К. Солових, В. О. Оришака, А. Е. Солових, С. Е. Катеринич] ; Центральноукраїн. нац. техн. ун-т. - 2−ге вид., перероб. та доп. - Кропивницький : ЦНТУ, 2019. - 27 с.[Електронний ресурс] – Режим доступу: http://dspace.kntu.kr.ua/jspui/handle/123456789/8769

55. The Top JavaScript Frameworks [Електронний ресурс] – Режим доступу до ресурсу: https://www.freecodecamp.org/news/complete-guide-for-front-end-developers-javascript-frameworks-2019/

56. JavaScript Frameworks 2020 [Електронний ресурс] – Режим доступу до ресурсу: https://www.npmjs.com/package/migrate-mongo

57. Web Technology [Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/web-technology/

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку програмного забезпечення BI-системи для вирішення завдань бізнес аналітики.

2 Підстава для розробки

Підставою для розробки служить завдання на магістерську роботу, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 34-13 від 04.08.2023 року).

3 Мета та призначення розробки

Метою роботи є застосування технологій семантичного веб для вирішення задач BI. В роботі розглянуто загальну архітектуру платформ для бізнес-аналізу, розроблено нову архітектуру зважаючи на необхідність інтеграції з BI, розроблено прототип семантичної системи для задач бізнес-аналітики.

4 Джерела розробки

Джерелом цієї магістерської роботи є відносна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– Розробку додатку;

– систему підключення та тестування баз даних ;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати в ОС Windows XP/Vista/7/8/10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows XP/Vista/7/8/10/11.

5.8.1 Обладнання

Комп’ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище Phyton 3.1 та Flask

Бази даних MySQL та PostgreSQL

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Економічні вимоги

7.1 Для ПЗ необхідно виробити функціонально-вартісний аналіз варіантів розробки.

7.2 Виконати розрахунок витрат показників економічного ефекту з урахуванням цін на 3 вересня 2023 року.

8 Вимоги щодо охорони праці

В частині охорони праці та техніки безпеки в магістерській роботі повинен бути розглянутий аналіз умов праці програміста та розрахунок штучного захисного заземлення.

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 109 аркушів.

10 Етапи розробки

10.1 Збір і обробка інформації по темі бакалаврської дипломної роботи. Постановка задачі на виконання бакалаврської дипломної роботи (складання ТЗ).

10.2 Проведення досліджень або експериментальних робіт для уточнення основних положень бакалаврської дипломної роботи.

10.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

11.1 Подання бакалаврської дипломної роботи на попередній захист 10.12.2023 р.

1.2 Подання магістерської роботи на захист 20.12.2023 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ
Керівник випускної кваліфікаційної роботи
за другим (магістерським) рівнем вищої освіти
_____ Босько В.В.

Дослідження та програмна реалізація застосування технологій семантичного веб для вирішення задач бізнес-аналітики

Лістинг програми

Код документу 12

Носій: CD/DVD-диск

Загальна кількість аркушів: 28

Літера: РП

Кропивницький – 2023

```
import argparse  # Command line parsing
import configparser  # Saving the models parameters
import datetime #
Chronometer import os #
Files management import
tensorflow as tf
import numpy as np
import math


from tqdm import tqdm # Progress bar
from tensorflow.python import debug as tf_debug


from chat.textdata import TextData
from chatbot.model import Model


class Chat:
  """
  Main class which launch the training or
  testing mode """
def___init__(self):
  """
  """
  # Model/dataset parameters
  self.args = None


  # Task specific object
  self.textData = None  # Dataset
```

```
  self.model = None  # Sequence to sequence model


  # Tensorflow utilities for convenience saving/logging

  self.writer = None

  self.saver = None

  self.modelDir = ''  # Where the model is saved

  self.globStep = 0  # Represent the number of iteration for the current model


  # TensorFlow main session (we keep track for the daemon)

  self.sess = None


  # Filename and directories constants
  self.MODEL_DIR_BASE = 'save' + os.sep +
  'model' self.MODEL_NAME_BASE = 'model'
  self.MODEL_EXT = '.ckpt'
  self.CONFIG_FILENAME = 'params.ini'
  self.CONFIG_VERSION = '0.5'

  self.TEST_IN_NAME = 'data' + os.sep + 'test' + os.sep +
  'samples.txt' self.TEST_OUT_SUFFIX = '_predictions.txt'
  self.SENTENCES_PREFIX = ['Q: ', 'A: ']

def main(self, args=None):

  """

  Launch the training and/or the
  interactive mode """

  print('Welcome to DeepQA v0.1
  !') print()

  print('TensorFlow detected: v{}'.format(tf.__version___))

  # General initialisation
  self.args =
  self.parseArgs(args) if
  not self.args.rootDir:

    self.args.rootDir = os.getcwd() # Use the current working directory
  #tf.logging.set_verbosity(tf.logging.INFO) # DEBUG, INFO, WARN (default), ERROR,
  or FATAL

  self.loadModelParams() # Update the self.modelDir and self.globStep, for now,
not used when loading Model (but need to be called before _getSummaryName)


  self.textData = TextData(self.args)

  # TODO: Add a mode where we can force the input of the decoder // Try to
visualize the predictions for

  # each word of the vocabulary / decoder input

  # TODO: For now, the model are trained for a specific dataset (because of
the maxLength which define the

  # vocabulary). Add a compatibility mode which allow to launch a model
trained on a different vocabulary (

  # remap the word2id/id2word variables).
```

```
if self.args.createDataset:
    print('Dataset created! Thanks for using this
    program') return  # No need to go further
```

```
  # Prepare the model
  with tf.device(self.getDevice()):
    self.model = Model(self.args, self.textData)

  # Saver/summaries
  self.writer =
  tf.summary.FileWriter(self._getSummaryName())
  self.saver = tf.train.Saver(max_to_keep=200)

  # TODO: Fixed seed (WARNING: If dataset shuffling, make sure to do that after
  saving the
  # dataset, otherwise, all which cames after the shuffling won't
  be replicable when # reloading the dataset). How to restore the
  seed after loading ??
  # Also fix seed for random.shuffle (does it works globally for all files ?)

  # Running session
  self.sess = tf.Session(config=tf.ConfigProto(
    allow_soft_placement=True, # Allows backup device for non GPU-available
operations (when forcing GPU)
    log_device_placement=False)  # Too verbose ?
  )  # TODO: Replace all sess by self.sess (not necessary a good idea) ?

  if self.args.debug:
    self.sess =
    tf_debug.LocalCLIDebugWrapperSession(self.sess)
    self.sess.add_tensor_filter("has_inf_or_nan",
    tf_debug.has_inf_or_nan)

  print('Initialize variables...')
  self.sess.run(tf.global_variables_initializer())

  # Reload the model eventually (if it exist.), on testing mode, the models are
not loaded here (but in predictTestset)
  if self.args.test !=
    Chatbot.TestMode.ALL:
    self.managePreviousModel(self.se
    ss)

  # Initialize embeddings with pre-trained word2vec vectors
  if
    self.args.initEmbeddings
    :
    self.loadEmbedding(self.
    sess)
```

```
if self.args.test:

  if self.args.test ==
    Chatbot.TestMode.INTERACTIVE:
    self.mainTestInteractive(self.sess)

  elif self.args.test ==
    Chatbot.TestMode.ALL: print('Start
    predicting...')
    self.predictTestset(self.sess)

    print('All predictions done')

  elif self.args.test ==
    Chatbot.TestMode.DAEMON: print('Daemon
    mode, running in background...')

  else:

    raise RuntimeError('Unknown test mode: {}'.format(self.args.test))  # Should
    never happen

else:

  self.mainTrain(self.sess)


if self.args.test != Chatbot.TestMode.DAEMON:
```

```
    self.sess.close()
    print("The End! Thanks for using this program")


def mainTrain(self, sess):
  """ Training
  loop Args:
    sess: The current running
  session """

  # Specific training dependent loading

  self.textData.makeLighter(self.args.ratioDataset) # Limit the number of training
  samples

  mergedSummaries = tf.summary.merge_all() # Define the summary operator (Warning:
Won't appear on the tensorboard graph)
  if self.globStep == 0: # Not restoring from previous run
    self.writer.add_graph(sess.graph)  # First time only

  # If restoring a model, restore the progression bar ? and current batch ?

  print('Start training (press Ctrl+C to save and exit)...')

  try:  # If the user exit while training, we still try to save the model
    for e in range(self.args.numEpochs):

      print()
      print("----- Epoch {}/{} ; (lr={})".format(e+1, self.args.numEpochs,
self.args.learningRate))

      batches = self.textData.getBatches()

      # TODO: Also update learning parameters eventually

      tic = datetime.datetime.now()
      for nextBatch in tqdm(batches, desc="Training"):
        # Training pass
        ops, feedDict = self.model.step(nextBatch)
        assert len(ops) == 2 # training, loss
```

```
        _, loss, summary = sess.run(ops + (mergedSummaries,),
        feedDict) self.writer.add_summary(summary,
        self.globStep)

        self.globStep += 1


        # Output training status

        if self.globStep % 100 == 0:

          perplexity = math.exp(float(loss)) if loss < 300 else float("inf")

          tqdm.write("----- Step %d -- Loss %.2f -- Perplexity %.2f" %
          (self.globStep, loss,

perplexity))


        # Checkpoint

        if self.globStep %
          self.args.saveEvery == 0:
          self._saveSession(sess)
```

```
      toc = datetime.datetime.now()


      print("Epoch finished in {}".format(toc-tic)) # Warning: Will overflow if
an epoch takes more than 24 hours, and the output isn't really nicer

  except (KeyboardInterrupt, SystemExit): # If the user press Ctrl+C while testing
  progress

    print('Interruption detected, exiting the program...')
  self._saveSession(sess) # Ultimate saving before complete
  exit
```

Model.py

```
import tensorflow as tf


from chatbot.textdata import Batch


class ProjectionOp:

  """ Single layer perceptron

  Project input tensor on the output
  dimension """

  def___init__(self, shape, scope=None, dtype=None):

    """

    Args:

      shape: a tuple (input dim,
      output dim) scope (str):
      encapsulate variables dtype:
      the weights type

    """

    assert len(shape)
    == 2 self.scope =
    scope

    # Projection on the keyboard

    with tf.variable_scope('weights_' +
      self.scope): self.W_t =
      tf.get_variable(

        'weights
        ',
        shape,

        # initializer=tf.truncated_normal_initializer() # TODO: Tune value (fct of
input size: 1/sqrt(input_dim))

        dtype=dtype

      )

      self.b = tf.get_variable(

        'bias',

        shape[0],
        initializer=tf.constant_initiali
        zer(), dtype=dtype

      )
```

```
    self.W = tf.transpose(self.W_t)


def getWeights(self):
  """ Convenience method for some tf
  arguments """
  return self.W, self.b


def___call__(self, X):
```

```
    """ Project the output of the decoder into the
    vocabulary space Args:

      X (tf.Tensor): input
    value """

    with tf.name_scope(self.scope):

      return tf.matmul(X, self.W) + self.b


class Model:

  """

  Implementation of a seq2seq
  model. Architecture:

    Encoder/decod
    er 2 LTSM
    layers

  """


  def___init__(self, args, textData):

    """

    Args:

      args: parameters of the
      model textData: the
      dataset object

    """

    print("Model creation...")


    self.textData = textData # Keep a reference on
    the dataset self.args = args # Keep track of the
    parameters of the model self.dtype = tf.float32


    # Placeholders

    self.encoderInputs  = None

    self.decoderInputs  = None  # Same that decoderTarget plus the <go>

    self.decoderTargets = None

    self.decoderWeights = None  # Adjust the learning to the target sentence size


    # Main
    operators
    self.lossFct =
    None self.optOp
    = None

    self.outputs = None # Outputs of the network, list of probability for each
    words


    # Construct the graphs
```

```
    self.buildNetwork()


  def buildNetwork(self):
    """ Create the computational
    graph """


    # TODO: Create name_scopes (for better graph visualisation)

    # TODO: Use buckets (better perfs)


    # Parameters of sampled softmax (needed for attention mechanism and a
large vocabulary size)
    outputProjection = None

    # Sampled softmax only makes sense if we sample less than vocabulary size.
```

```
if 0 < self.args.softmaxSamples <
  self.textData.getVocabularySize(): outputProjection =
  ProjectionOp(

    (self.textData.getVocabularySize(),
    self.args.hiddenSize),
    scope='softmax_projection',

    dtype=self.dtype

  )


  def sampledSoftmax(labels, inputs):

    labels = tf.reshape(labels, [-1, 1])  # Add one dimension (nb of true
    classes, here 1)


    # We need to compute the sampled_softmax_loss using
    32bit floats to # avoid numerical instabilities.

    localWt  = tf.cast(outputProjection.W_t,
            tf.float32) localB          =
    tf.cast(outputProjection.b,
            tf.float32) localInputs =
    tf.cast(inputs,                     tf.float32)


    return tf.cast(
      tf.nn.sampled_softmax_l
      oss(

        localWt,  # Should have shape [num_classes, dim]

        localB,
        labels,
        localInpu
        ts,

        self.args.softmaxSamples,  # The number of classes to randomly sample
        per batch

        self.textData.getVocabularySize()), # The number of classes

      self.dtype)


# Creation of the rnn cell

def create_rnn_cell():

  encoDecoCell = tf.contrib.rnn.BasicLSTMCell(  # Or GRUCell,
  LSTMCell(args.hiddenSize)

    self.args.hiddenSize,

  )

  if not self.args.test:  # TODO: Should use a placeholder instead

    encoDecoCell =
      tf.contrib.rnn.DropoutWrapper(
      encoDecoCell,

      input_keep_prob=1.0,
      output_keep_prob=self.args.dropout
```

```
        )

      return encoDecoCell

    encoDecoCell = tf.contrib.rnn.MultiRNNCell(
      [create_rnn_cell() for _ in
      range(self.args.numLayers)],

    )


    # Network input (placeholders)


  with tf.name_scope('placeholder_encoder'):

    self.encoderInputs = [tf.placeholder(tf.int32,  [None, ])
                          for _ in

range(self.args.maxLengthEnco)] # Batch size * sequence length *
                          input dim


  with tf.name_scope('placeholder_decoder'):

self.decoderInputs = [tf.placeholder(tf.int32,  [None, ], name='inputs')
                                                          for _ in

    range(self.args.maxLengthDeco)] # Same sentence length for input and
                                                    output (Right ?)
```

```
      self.decoderTargets = [tf.placeholder(tf.int32,  [None, ], name='targets')
      for _ in
range(self.args.maxLengthDeco)]
      self.decoderWeights = [tf.placeholder(tf.float32, [None, ], name='weights')
      for _ in
range(self.args.maxLengthDeco)]


    # Define the network
    # Here we use an embedding model, it takes integer as input and convert them
    into word vector
fo
r
    # better word representation
    decoderOutputs, states = tf.contrib.legacy_seq2seq.embedding_rnn_seq2seq(
      self.encoderInputs, # List<[batch=?, inputDim=1]>, list of size
      args.maxLength self.decoderInputs, # For training, we force the correct
      output (feed_previous=False) encoDecoCell,
      self.textData.getVocabularySize(),
      self.textData.getVocabularySize(), # Both encoder and decoder have the same
      number of
clas
s
      embedding_size=self.args.embeddingSize, # Dimension of each word
      output_projection=outputProjection.getWeights() if outputProjection else
      None, feed_previous=bool(self.args.test)  # When we test (self.args.test),
      we use previous output as
next input (feed_previous)
    )


    # TODO: When the LSTM hidden size is too big, we should project the LSTM
output into a smaller space (4086 => 2046): Should speed up
    # training and reduce memory usage. Other solution, use sampling softmax


    # For testing only
    if self.args.test:
      if not outputProjection:
        self.outputs =
        decoderOutputs
      else:
        self.outputs = [outputProjection(output) for output in decoderOutputs]


      # TODO: Attach a summary to visualize the output


    # For training only
    else:
      # Finally, we define the loss function
```

```
        self.lossFct =
          tf.contrib.legacy_seq2seq.sequence_loss(
          decoderOutputs,

          self.decoderTargets,
          self.decoderWeights,
          self.textData.getVocabulary
          Size(),

          softmax_loss_function= sampledSoftmax if outputProjection else None # 
If None, use default SoftMax

        )

        tf.summary.scalar('loss', self.lossFct)  # Keep track of the cost


        # Initialize the optimizer

        opt = tf.train.AdamOptimizer(
          learning_rate=self.args.learningRate,
          beta1=0.9,
```

```
        beta2=0.999
        ,
        epsilon=1e-
        08

      )

      self.optOp = opt.minimize(self.lossFct)


  def step(self, batch):

    """ Forward/training step operation.

    Does not perform run on itself but just return the operators to do so. Those
    have then to be run Args:

      batch (Batch): Input data on testing mode, input and target
    on output mode Return:

      (ops), dict: A tuple of the (training, loss) operators or (outputs,)
in testing mode with the associated feed dictionary

    """


    # Feed the dictionary

    feedDict =
    {} ops =
    None


    if not self.args.test: # Training

      for i in range(self.args.maxLengthEnco):
        feedDict[self.encoderInputs[i]] =
        batch.encoderSeqs[i]

      for i in range(self.args.maxLengthDeco):
        feedDict[self.decoderInputs[i]] = batch.decoderSeqs[i]
        feedDict[self.decoderTargets[i]] = batch.targetSeqs[i]
        feedDict[self.decoderWeights[i]] = batch.weights[i]


ops = (self.optOp, self.lossFct)

  else:  # Testing (batchSize ==
                              1)

      for i in range(self.args.maxLengthEnco):
        feedDict[self.encoderInputs[i]] =
        batch.encoderSeqs[i]

      feedDict[self.decoderInputs[0]] =
      [self.textData.goToken] ops = (self.outputs,)

    # Return one pass operator

    return ops, feedDict
```

Textdata.py

```
import numpy as np

import nltk  # For tokenize

from tqdm import tqdm # Progress bar

import pickle # Saving the
data import math # For float
comparison import os # Checking
file existance import random

import string
```

```
import collections


from chatbot.corpus.cornelldata import
CornellData from chatbot.corpus.opensubsdata
import OpensubsData from
chatbot.corpus.scotusdata import ScotusData

from chatbot.corpus.ubuntudata import UbuntuData

from chatbot.corpus.lightweightdata import LightweightData


class Batch:

  """Struct containing
  batches info """

  def___init__(self):
    self.encoderSeqs
    = []
    self.decoderSeqs
    = []
    self.targetSeqs =
    [] self.weights =
    []


class TextData:

  """Dataset class

  Warning: No vocabulary
  limit """


  availableCorpus = collections.OrderedDict([ # OrderedDict because the first
element is the default choice

    ('cornell', CornellData),
    ('opensubs', OpensubsData),
    ('scotus',  ScotusData),
    ('ubuntu', UbuntuData),
    ('lightweight',
    LightweightData),

  ])


  @staticmethod

  def corpusChoices():

    """Return the dataset
    availables Return:

      list<string>: the supported
    corpus """

    return list(TextData.availableCorpus.keys())


  def___init__(self,
    args): """Load all
    conversations Args:
```

```
  args: parameters of the
model """

# Model parameters

self.args = args


# Path variables

self.corpusDir = os.path.join(self.args.rootDir, 'data',
self.args.corpus) basePath = self._constructBasePath()

self.fullSamplesPath = basePath + '.pkl'  # Full sentences length/vocab

self.filteredSamplesPath = basePath + '-length{}-filter{}-
vocabSize{}.pkl'.format(
```

```
      self.args.maxLength,
      self.args.filterVoca
      b,
      self.args.vocabulary
      Size,

    )  # Sentences/vocab filtered for this model


    self.padToken = -1 # Padding
    self.goToken = -1 # Start of
    sequence self.eosToken = -1  #
    End of sequence

    self.unknownToken = -1  # Word dropped from vocabulary


    self.trainingSamples = []  # 2d array containing each question and his answer
    [[input,target]]


    self.word2id = {}

    self.id2word = {} # For a rapid conversion (Warning: If replace dict by
list, modify the filtering to avoid linear complexity with del)

    self.idCount = {} # Useful to filters the words (TODO: Could replace dict by
list or use collections.Counter)


    self.loadCorpus()


    # Plot some stats:

    self._printStats()


    if
      self.args.playDat
      aset:
      self.playDataset(
      )


  def _printStats(self):

    print('Loaded {}: {} words, {} QA'.format(self.args.corpus,
len(self.word2id), len(self.trainingSamples)))


  def _constructBasePath(self):

    """Return the name of the base prefix of the
    current dataset """

    path = os.path.join(self.args.rootDir, 'data' + os.sep +
    'samples' + os.sep) path += 'dataset-
    {}'.format(self.args.corpus)

    if self.args.datasetTag:

      path += '-' + self.args.datasetTag

    return path
```

```
def makeLighter(self, ratioDataset):
  """Only keep a small fraction of the dataset,
  given by the ratio """
  #if not math.isclose(ratioDataset, 1.0):
  #  self.shuffle()  # Really ?
  #  print('WARNING: Ratio feature not implemented !!!')
  pass


def shuffle(self):
  """Shuffle the training
  samples """
  print('Shuffling the dataset...')
  random.shuffle(self.trainingSamples)
```

```
def _createBatch(self, samples):
    """Create a single batch from the list of sample. The batch size is
automatically defined by the number of
    samples given.
    The inputs should already be inverted. The target should already have
    <go> and <eos> Warning: This function should not make direct calls to
    args.batchSize !!!
    Args:
      samples (list<Obj>): a list of samples, each sample being on the
    form [input, target] Return:
      Batch: a batch
    object en """

    batch = Batch()
    batchSize =
    len(samples)

     # Create the batch
       tensor for i in
     range(batchSize): #
      Unpack the sample
     sample = samples[i]

        if not self.args.test and self.args.watsonMode:  # Watson mode: invert
                                    question and answer
                sample =
          list(reversed(sample))

      if not self.args.test and self.args.autoEncode: # Autoencode: use either
the question or answer for both input and output
        k = random.randint(0, 1)
        sample = (sample[k], sample[k])
      # TODO: Why re-processed that at each epoch ? Could precompute that
      # once and reuse those every time. Is not the bottleneck so won't change
      # much ? and if preprocessing, should be compatible with autoEncode & cie.
      batch.encoderSeqs.append(list(reversed(sample[0]))) # Reverse inputs (and
not outputs), little trick as defined on the original seq2seq paper
      batch.decoderSeqs.append([self.goToken] + sample[1] + [self.eosToken]) #
Add the <go> and <eos> tokens
      batch.targetSeqs.append(batch.decoderSeqs[-1][1:]) # Same as decoder, but
shifted to the left (ignore the <go>)

      # Long sentences should have been filtered during the dataset creation
      assert len(batch.encoderSeqs[i]) <= self.args.maxLengthEnco
      assert len(batch.decoderSeqs[i]) <= self.args.maxLengthDeco
```

```
        # TODO: Should use tf batch function to automatically add padding and batch
        samples

        # Add padding & define weight

        batch.encoderSeqs[i] = [self.padToken] *
(self.args.maxLengthEnco - len(batch.encoderSeqs[i])) +
batch.encoderSeqs[i]  # Left padding for the input

        batch.weights.append([1.0] * len(batch.targetSeqs[i]) + [0.0] *
(self.args.maxLengthDeco - len(batch.targetSeqs[i])))

        batch.decoderSeqs[i] = batch.decoderSeqs[i] + [self.padToken] *
        (self.args.maxLengthDeco

- len(batch.decoderSeqs[i]))

        batch.targetSeqs[i] = batch.targetSeqs[i] + [self.padToken] *
(self.args.maxLengthDeco - len(batch.targetSeqs[i]))


    # Simple hack to reshape the batch
```

```
    encoderSeqsT = [] # Corrected orientation
    for i in
      range(self.args.maxLengthEnco):
      encoderSeqT = []
      for j in range(batchSize):
        encoderSeqT.append(batch.encoderSeqs[
        j][i])
      encoderSeqsT.append(encoderSe
    qT) batch.encoderSeqs =
    encoderSeqsT


    decoderSeqsT =
    [] targetSeqsT
    = [] weightsT
    = []
    for i in
      range(self.args.maxLengthDeco):
      decoderSeqT = []
      targetSeqT =
      [] weightT =
      []
      for j in range(batchSize):
        decoderSeqT.append(batch.decoderSeqs[
        j][i])
        targetSeqT.append(batch.targetSeqs[j]
        [i])
        weightT.append(batch.weights[j][i])
      decoderSeqsT.append(decoderSe
      qT)
      targetSeqsT.append(targetSeqT
      ) weightsT.append(weightT)
    batch.decoderSeqs =
    decoderSeqsT batch.targetSeqs
    = targetSeqsT batch.weights
    = weightsT


    # # Debug
    # self.printBatch(batch) # Input inverted, padding
    should be correct #
    print(self.sequence2str(samples[0][0]))
    # print(self.sequence2str(samples[0][1]))  # Check we did not modified the
    original sample


    return batch


def getBatches(self):
    """Prepare the batches for the
    current epoch Return:
      list<Batch>: Get a list of the batches for the
    next epoch """
```

```
self.shuff
le()
batches =
[]

def genNextSamples():

  """ Generator over the mini-batch
  training samples """

  for i in range(0, self.getSampleSize(), self.args.batchSize):

    yield self.trainingSamples[i:min(i + self.args.batchSize,
    self.getSampleSize())]


# TODO: Should replace that by generator (better: by tf.queue)


for samples in
  genNextSamples(): batch =
  self._createBatch(samples)
```

```
    batches.append(batch)

  return batches


def getSampleSize(self):
  """Return the size of
  the dataset Return:

    int: Number of
  training samples """

  return len(self.trainingSamples)


def getVocabularySize(self):

  """Return the number of words present
  in the dataset Return:

    int: Number of word on the
  loader corpus """

  return len(self.word2id)


def loadCorpus(self):

  """Load/create the
  conversations data """

  datasetExist = os.path.isfile(self.filteredSamplesPath)

  if not datasetExist:  # First time we load the database: creating all files

    print('Training samples not found. Creating dataset...')


    datasetExist = os.path.isfile(self.fullSamplesPath) # Try to construct
the dataset from the preprocessed entry

    if not datasetExist:
      print('Constructing full
      dataset...')


      optional = ''

      if self.args.corpus ==
        'lightweight': if not
        self.args.datasetTag:

          raise ValueError('Use the --datasetTag to define the
        lightweight file to use.') optional = os.sep +
        self.args.datasetTag  # HACK: Forward the filename


      # Corpus creation

      corpusData =
      TextData.availableCorpus[self.args.corpus](self.corpusDir +
      optional) self.createFullCorpus(corpusData.getConversations())
      self.saveDataset(self.fullSamplesPath)

    else:
```

```
  self.loadDataset(self.fullSamplesP
ath) self._printStats()


print('Filtering words (vocabSize = {} and wordCount >
  {})...'.format( self.args.vocabularySize,

  self.args.filterVocab

))

self.filterFromFull()  # Extract the sub vocabulary for the given
maxLength and filterVocab


# Saving

print('Saving dataset...')
self.saveDataset(self.filteredSamplesPath)  #
```