

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіоелектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ, РОЗРОБКА ТА ПРОГРАМНА РЕАЛІЗАЦІЯ
ІНТЕРАКТИВНОЇ ПЛАТФОРМИ ДЛЯ ІНТЕЛЕКТУАЛЬНОГО
АНАЛІЗУ ДАНИХ
RESEARCH, DEVELOPMENT AND SOFTWARE IMPLEMENTATION OF THE
INTERACTIVE PLATFORM FOR DATA MINING

Виконав: студент 2 курсу, групи КНТ-119м
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Кулик Л.І.

(прізвище та ініціали)

Керівник Табунщик Г.В.

(прізвище та ініціали)

Рецензент Шитікова О.В.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування вищого навчального закладу)

Інститут, факультет ІРЕ, ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 121 Інженерія програмного забезпечення
(код і назва)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.

С.О. Субботін
“ ” 20 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

Кулику Леву Іллічу

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження, розробка та програмна реалізація інтерактивної платформи для інтелектуального аналізу даних. Research, Development and Software Implementation of the Interactive Platform for Data Mining

керівник проєкту (роботи) Табунщик Галина Володимирівна, к.т.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від “ 3 ” листопада 2020 року
№ 301

2. Строк подання студентом проєкту (роботи) 10 грудня 2020 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження. 2. Матеріали і методи. 3. Розробка архітектури програми. 4. Основні рішення щодо реалізації компонентів системи. 5. Експлуатація, тестування та експериментальне дослідження програми. 6. Організаційно-економічна частина. 7. Охорона праці та безпека у надзвичайних ситуаціях. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації.

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Табунщик Г.В., професор		
Організаційно- економічна частина	Пожуєва Т.О., професор		
Охорона праці та безпека у надзвичайних ситуаціях	Коробко О.В., ст. викладач		
Нормоконтролер	Камінська Ж.К., асистент		

7. Дата видачі завдання “4” вересня 2020 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	2–3 тижні	Розділ 1
3	Розробка та удосконалення методів, моделей й алгоритмів вирішення задачі.	4–5 тижні	Розділ 2
4	Розробка архітектури програми.	6 тиждень	Розділ 3
5	Розробка програми.	7–8 тижні	Розділ 4
6	Тестування та експериментальне дослідження програмного забезпечення.	9 тиждень	Розділ 5
7	Розробка організаційно-економічної частини	10 тиждень	Розділ 6
8	Розробка заходів з охорони праці та цивільної безпеки.	11 тиждень	Розділ 7
9	Оформлення пояснювальної записки та документів до неї.	12-13 тиждень	Додатки
10	Нормоконтроль та рецензування.	14–15 тижні	
11	Захист роботи.	16 тиждень	

Студент

Кулик Л.І.
(підпис) (прізвище та ініціали)

Керівник проєкту (роботи)

Табунщик Г.В.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
185 с., 56 рис., 15 табл., 5 дод., 41 джерела.

ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ, ПЛАТФОРМА, MVC, MVVM,
КОРИСТУВАЦЬКИ КОД, КОНТЕЙНЕРІЗАЦІЯ, DOCKER.

Об'єкт дослідження – процес інтелектуального аналізу даних.

Предмет дослідження – інтелектуальний аналіз даних з використанням вебтехнологій.

Мета роботи – покращити процес інтелектуального аналізу даних.

Моделі та методи: об'єктно-орієнтоване програмування, архітектурний шаблон модель-вид-контролер, архітектурний шаблон модель-вид-вид представлення, мови програмування Python та JavaScript, інтегроване середовище розробки PyCharm.

Результат дипломної роботи – програмне забезпечення для інтерактивної платформи для інтелектуального аналізу даних, що може використовуватися як інструмент в дослідницьких, комерційних та навчальних цілях.

Одержані результати можуть бути використані для створення як відкритої так і закритої площадки для реалізацій методів аналізу даних.

Коефіцієнт техніко-економічної ефективності дослідження складає 1,07 – з чого можна зробити висновок, що дана дослідницька робота – є ефективною.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 185 pages, 56 figures, 15 tables, 5 appendixes, 41 sources.

DATA MINING, PLATFORM, MVC, MVVM, USER`S CODE, CONTAINERIZATION, DOCKER.

Object of study is a process of data mining.

Subject of study is data mining with the help of web-technologies.

The purpose of the work is to improve the process of data mining.

Materials and methods: object-oriented programming, architectural pattern model-view-controller, architectural pattern model-view-view, programming languages Python and JavaScript, integrated development environment PyCharm.

Results. The system for an interactive platform for data mining that can be used as a tool for research, commercial and educational purposes.

The field of use are both open and closed platform for the implementation of data analysis methods.

Economic efficiency. The coefficient of technical and economic efficiency of the study is 1.07 from which it can be concluded that this research work is effective.

ЗМІСТ

Перелік скорочень та умовних познач	9
Вступ	10
1 Аналіз проблеми та постановка завдань дослідження	11
1.1 Огляд існуючого стану проблеми	11
1.2 Огляд існуючих рішень	12
1.3 Порівняння існуючих та розроблюваного програмних продуктів	20
1.4 Формальна постановка задачі	21
1.5 Висновки за розділом 1	22
2 Матеріали і методи	23
2.1 Виконання користувацького коду на стороні сервера	23
2.2 Виконання користувацького коду на стороні клієнта	28
2.3 Порівняння методів запуску користувацьких Python-скриптів	33
2.4 Висновки за розділом 2	34
3 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ	35
3.1 Опис основних вимог до програми	35
3.2 Проектування структури програмної системи	36
3.3 Моделювання логіки програми	41
3.4 Проектування бази даних	45
3.5 Вибір мови програмування	46
3.6 Вибір фреймворку для розробки	49
3.7 Обрання системи керування базою даних	52
3.8 Вибір середовища розробки	55
3.9 Вимоги до технічних засобів	57

3.10	Висновки за розділом 3	58
4	Основні рішення щодо реалізації компонентів системи.....	59
4.1	Структура бекенду	59
4.2	Структура фронтенду	63
4.3	Вхідні та вихідні дані бекенду.....	72
4.4	Вхідні та вихідні дані фронтенду	73
4.5	Організація даних.....	73
4.6	Висновки за розділом 4	75
5	Експлуатація, тестування та експериментальне дослідження програми	76
5.1	Опис застосування	76
5.2	Інструкція по експлуатації програми.....	76
5.3	Дослідження розробленої програми	77
5.4	Висновки за розділом 4	82
6	Організаційно-економічна частина	83
6.1	Концепція економічного обґрунтування	83
6.2	Планування розробки програмного продукту	84
6.3	Розрахунок основної заробітної плати	86
6.4	Розрахунок додаткової заробітної плати	86
6.5	Єдиний соціальний внесок.....	87
6.6	Визначення витрат на матеріали	87
6.7	Витрати на спеціальне обладнання	88
6.8	Інші прямі витрати.....	92
6.9	Розрахунок накладних витрат.....	93
6.10	Розрахунок економічної ефективності програмного продукту	94
7	Охорона праці та безпека у надзвичайних ситуаціях.....	97

7.1	Аналіз потенційних небезпек	97
7.2	Заходи із забезпечення безпеки	98
7.3	Заходи з забезпечення виробничої санітарії та гігієни праці	101
7.4	Заходи безпеки у надзвичайних ситуаціях	106
	Висновки	113
	Перелік джерел посилань	115
	Додаток А Технічне завдання	120
	Додаток Б Приклади відповідей вебсервера	125
	Додаток В Розроблені Python-скрипти	129
	Додаток Д Код програми	133
	Додаток Е Слайди презентації	175

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

БД	– база даних;
ВР	– віртуальна реальність;
КПО	– віртуальна реальність;
ПЗ	– програмне забезпечення;
ПК	– персональний комп'ютер;
ПП	– програмний продукт;
ООП	– об'єктно-орієнтоване програмування;
ОС	– операційна система;
СКБД	– система керування базами даних;
API	– application programing interface;
JWT	– JSON Web Token;
MVC	– Model-View-Controller;
MVVM	– Model-View-ViewModel;
REST	– representational state transfer;
SPA	– single-page application;
SQL	– structured query language;
SSH	– secure shell.

ВСТУП

В сучасну епоху оцифрування та стрімкої зміни способів взаємодії між людьми, суспільство починає, як ніколи раніше, широко використовувати різноманітні інформаційні технології у повсякденному і професійному житті.

В свою чергу, чергу це призводить до того, що ми починаємо продукувати таку кількість інформації, що людині проаналізувати її стає просто неможливо. А сучасний бізнес і наука для якісних рішень вимагають робити аналіз майже усього. Саме тут виникає потреба в інтелектуальному аналізі даних, за допомогою якого можна автоматизувати та значно прискорити цей процес.

Але тут виникає наступні проблеми – нестача фахівців, інструментів для швидкого прототипування та розповсюдження рішень з інтелектуального аналізу даних.

Актуальність теми пояснюється тим, що зараз ринок інтерактивних платформ для інтелектуального аналізу даних є досить порожнім, а існуючі рішення відносно складні у використанні та зазвичай вибагливі до ресурсів.

Об'єкт дослідження – процес розробки рішень для інтелектуального аналізу даних.

Предмет дослідження – розробка та прототипування рішень для інтелектуального аналізу даних.

Мета роботи – покращити процес розробки та прототипування рішень для інтелектуального аналізу даних.

Завдання роботи – дослідити предметну область, розробити програмне забезпечення для розробки та прототипування рішень для інтелектуального аналізу даних.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

Цей розділ спрямовано на дослідження проблематики області інтелектуального аналізу даних в інтерактивних середовищах та обґрунтуванню використання сучасних архітектурних підходів для розробки рішень для цієї області.

1.1 Огляд існуючого стану проблеми

На сьогоднішній день, в час коли кожна сферу життєдіяльності людей намагаються оцифрувати, генерується нескінченний потік даних, які необхідно якимось чином оброблювати та аналізувати. Бізнес в епоху поведінкової економіки вже не може просто спиратися на відгуки користувачів, йому необхідно відслідковувати та аналізувати поведінку користувачів на їх ресурсах і способи використання продуктів. В свою чергу, соціальні організації повинні раціонально використовувати свої обмежені ресурси, коли кількість людей, які потребують допомоги, постійно збільшується. Для вирішення всіх цих задач використовується інтелектуальний аналіз даних. Оскільки, економіка та світ швидко розвиваються, необхідно розробляти інструменти для зручного і ефективного вирішення задач з аналізу даних та навчання нових спеціалістів.

Один з типів інструментів, який підходить для вирішення поставлених задач – це платформи інтерактивних обчислень. Завдяки ним не потрібно детально розумітися в установленні та налаштуванні інструментів необхідних для початку роботи. Програмне забезпечення (ПЗ) даного типу дозволяє одразу приступати до вирішення задачі пропускаючи суто технічні етапи з налаштування середовища розробки. Зазвичай платформи інтерактивних обчислень надають можливість написання програмного коду за допомогою однієї з інтерпретованих мов програмування. Як правило цією мовою виступає Python.

Мова програмування Python стала популярна серед багатьох галузей інженерії та аналізу даних. Завдяки своїй простоті у використанні та низькому порогу входу для створення своїх перших прикладних програмних рішень.

Першим представником платформи інтерактивних обчислень був Sketchpad створений Іваном Едвардом Сазерлендом у 1963 році [1].

Найбільш популярними представниками платформ інтерактивних обчислень з якими зустрічалося більшість розробників та людей, яких цікавить програмування – це JSFiddle [2] та CodePen [3]. Це не просто платформи, а цілі інтернет-спільноти для тестування та демонстрації коду HTML, CSS та JavaScript (JS). Але нажаль вони зовсім не надають інструментів для інтелектуального аналізу даних. Ці можливості можна знайти лише на спеціалізованих платформах.

Сьогодні спеціалізовані платформи для інтелектуальний аналіз даних вже використовуються в багатьох сферах. В науковому та бізнес середовищі дані рішення застосовують для прототипування та поширення ідей вирішення тих чи інших задач.

Також всі навчальні заклади починають потроху використовувати подібні рішення в процесі навчання студентів. Адже налаштування середовища на багатьох комп'ютерах є досить складною задачею. І студентам новачкам, які не є фахівцями з адміністрування комп'ютерів може бути досить складно налаштувати свій домашній комп'ютер для роботи з даними за допомогою Python. Тому, їм може знадобитися у пригоді онлайн сервіс для аналізу даних.

Актуальність розробки для інтерактивної платформи для інтелектуального аналізу даних пояснюється тим, що вона може стати у пригоді як у бізнес секторі, науковому та навчальному середовищі.

1.2 Огляд існуючих рішень

Зараз існує безліч різноманітних інтерактивних платформ, які виконують програмний код на багатьох мовах програмування, але в даному розділі будуть

розглянуті лише ті платформи, за допомогою яких можна проводити інтелектуальний аналіз даних.

На сьогоднішній день існує декілька популярних інтерактивних платформи для інтелектуального аналізу даних, які гідні уваги, а саме:

- Project Jupyter;
- Apache Zeppelin;
- Iodide.

1.2.1 Project Jupyter

Project Jupyter – це некомерційна організація, створена для розробки програмного забезпечення з відкритим кодом, відкритих стандартів та послуг для інтерактивних обчислень на десятках мов програмування. У 2014 році Фернандо Перес відокремив Project Jupyter від батьківського проєкту IPython. Ім'я проєкту "Юпітер" – це посилення на три основні мови програмування, які підтримує Юпітер, – це Julia, Python і R, а також данину поваги блокнотам Галілея, що фіксують відкриття супутників Юпітера. В рамках Project Jupyter розроблено та підтримуються продукти інтерактивних обчислень Jupyter Notebook, JupyterHub та JupyterLab, наступна версія Jupyter Notebook [4]. Далі більш детально буде розглянуто Jupyter Notebook, бо цей продукт найбільш подібний до розроблюваного рішення.

Jupyter Notebook – це вебплатформа для інтерактивних обчислень для створення документів(програм) для Jupyter Notebook. Термін "Notebook" може визначати багато різних сутностей, головним чином вебзастосунок Jupyter, Jupyter Python вебсервер або формат документа Jupyter залежно від контексту [5]. Саме через цю полісемантичність багатьом новачкам досить складно розібратися в продуктах Project Jupyter (рис. 1.1).

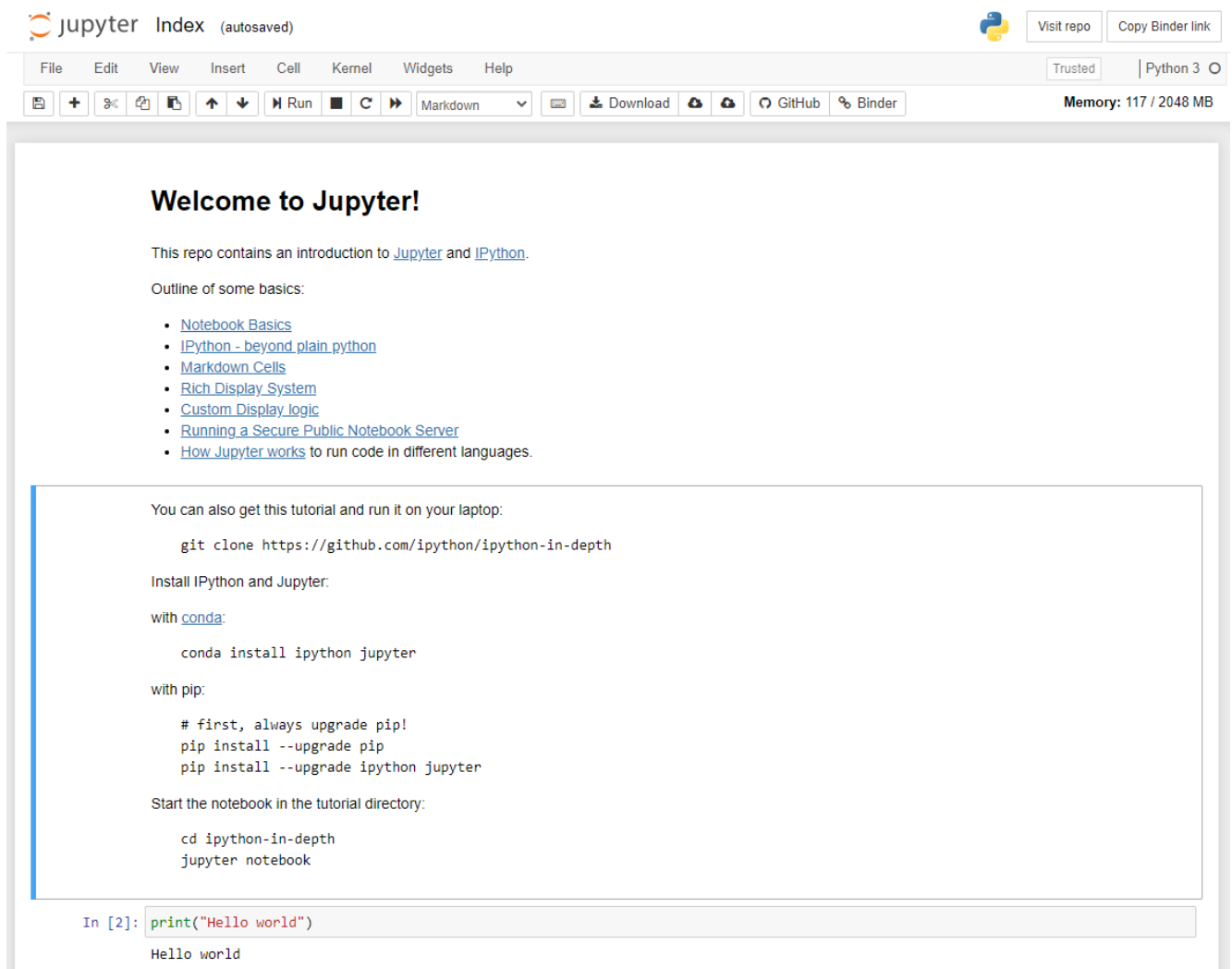


Рисунок 1.1 – Інтерфейс Jupyter Notebook

Документ Jupyter Notebook – це JSON документ, який має систему контролю версій і містить упорядкований список клітинок вводу/виводу, які можуть містити код, текст, математику, графіки та мультимедійні файли (рис. 1.2). Документи Jupyter Notebook, як правило, мають ".ipynb" розширення [6].

```

{
  "cells": [
    {
      "metadata": {
      },
      "cell_type": "markdown",
      "source": "# Welcome to Jupyter!"
    },
    {
      "metadata": {
      },
      "cell_type": "markdown",
      "source": "This repo contains an intr
    },
    {
      "metadata": {
      },
      "cell_type": "markdown",
      "source": "You can also get this tuto
    },
    {
      "metadata": {
        "trusted": true
      },
      "cell_type": "code",
      "source": "print(\"Hello world\")",
      "execution_count": 2,
      "outputs": [
        {
          "output_type": "stream",
          "text": "Hello world\n",
          "name": "stdout"
        }
      ]
    },
    {
      "metadata": {
        "trusted": true
      },
      "cell_type": "code",
      "source": "",
      "execution_count": null,
      "outputs": [
      ]
    }
  ],
  "metadata": {
    "kernelspec": {
      "name": "python3",
      "display_name": "Python 3",
      "language": "python"
    },
    "language_info": {
      "name": "python",

```

Рисунок 1.2 – Приклад вмісту ".ipynb" файла

З точки зору використання потужностей Jupyter Notebook – це сервер-орієнтоване рішення, бо всі обчислювальні операції виконуються на стороні вебсервера. Клієнт отримує лише результат. Дане рішення було прийнято через те, що Jupyter Notebook є досить самостійною, але все ж таки частиною більшого рішення Jupyter Lab, яке спрямоване на використання в кластерних системах з можливістю швидкого та простого масштабування.

1.2.2 Apache Zeppelin

Apache Zeppelin (рис. 1.3) – це багатоцільова інтерактивна платформа інтелектуального аналізу даних, яка пропонує інструменти для:

- завантаження даних (Data Ingestion);
- виявлення даних (Data Discovery);
- аналізу даних (Data Analytics);
- відображення даних (Data Visualization);
- сумісної роботи з даними (Data Collaboration).

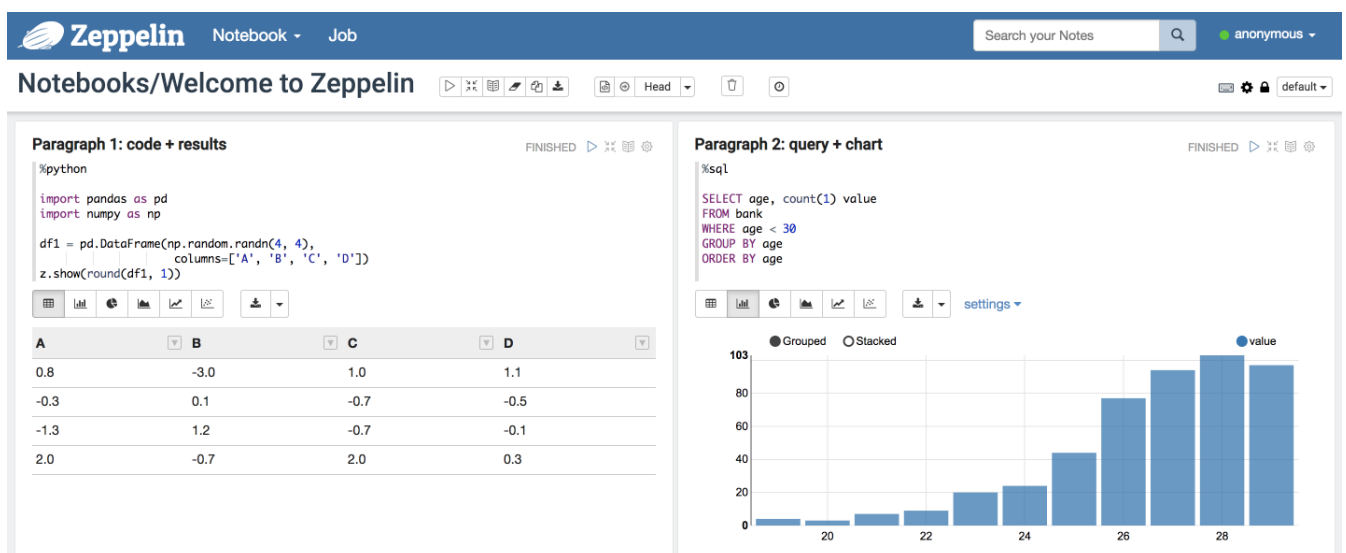


Рисунок 1.3 – Приклад інтерфейсу Apache Zeppelin

Комбінування різних джерел даних – в рамках одного дашборда – одне з його ключових переваг. В рамках стандартної збірки включено значний набір

інтерпретаторів (до NoSQL і реляційних баз даних). Саме завдяки цьому Apache Zeppelin дозволяє завантажувати дані з багатьох різних джерел таких як [7]-[8]:

- Postgres;
- MySQL;
- MariaDB;
- AWS Redshift;
- Apache Hive;
- Apache Phoenix;
- Apache Drill;
- Apache Tajo;
- Apache Geode;
- Elasticsearch;
- Lens;
- Alluxio;
- та багато інших.

З точки зору використання потужностей Apache Zeppelin як і Jupyter Notebook – це сервер-орієнтоване рішення, бо всі обчислювальні операції виконуються на стороні вебсервера. Так було зроблено через те, що Apache Zeppelin одразу створювався як корпоративне рішення для роботи в кластерах.

Документ Apache Zeppelin зберігає, також як і Jupyter Notebook, у форматі JSON файлів, але має свою унікальну структуру (рис. 1.4).

```
{
  "paragraphs": [
    {
      "title": "",
      "text": "%md\n\n# Classification of motor imagery electroencephalogram signals\n\nAuthor: Paul Bustios",
      "dateUpdated": "Aug 21, 2016 12:08:30 AM",
      "config": {
        "colWidth": 12.0,
        "editorMode": "ace/mode/markdown",
        "editorHide": true,
        "graph": {
          "mode": "table",
          "height": 300.0,
          "optionOpen": false,
          "keys": [],
          "values": [],
          "groups": [],
          "scatter": {}
        },
        "enabled": true
      },
      "settings": {
        "params": {},
        "forms": {}
      },
      "apps": [],
      "jobName": "paragraph_1463773880824_1661539059",
      "id": "20160520-165120_1968765561",
      "result": {
        "code": "SUCCESS",
        "type": "HTML",
        "msg": "\u003ch1\u003eClassification of motor imagery electroencephalogram signals\u003c/h1\u003e\n\u003cp\u003eAuthor: Paul Bustios"
      },
      "dateCreated": "May 20, 2016 4:51:20 AM",
      "dateStarted": "Aug 21, 2016 12:08:30 AM",
      "dateFinished": "Aug 21, 2016 12:08:31 AM",
      "status": "FINISHED",
      "progressUpdateIntervalMs": 500
    },
    {
      "title": "Getting the data",
      "text": "%sh\n\nEND\u003d${Number of files to download\u003d8,5|6|7|8|9|10|11|12|13|14}",
      "dateUpdated": "Jun 23, 2016 12:55:33 PM",
      "config": {
        "colWidth": 12.0,
        "editorMode": "ace/mode/sh",
        "editorHide": false,
        "title": true,
        "graph": {
          "mode": "table",
          "height": 86.0,
          "optionOpen": false,
          "keys": [],
          "values": [],
          "groups": [],
          "scatter": {}
        }
      },
      "settings": {
        "params": {},
        "forms": {}
      },
      "apps": [],
      "jobName": "paragraph_1463773880824_1661539059",
      "id": "20160520-165120_1968765561",
      "result": {
        "code": "SUCCESS",
        "type": "HTML",
        "msg": "\u003ch1\u003eClassification of motor imagery electroencephalogram signals\u003c/h1\u003e\n\u003cp\u003eAuthor: Paul Bustios"
      },
      "dateCreated": "May 20, 2016 4:51:20 AM",
      "dateStarted": "Aug 21, 2016 12:08:30 AM",
      "dateFinished": "Aug 21, 2016 12:08:31 AM",
      "status": "FINISHED",
      "progressUpdateIntervalMs": 500
    }
  ]
}
```

Рисунок 1.4 – Приклад вмісту документа Apache Zeppelin

1.2.3 Iodide

Iodide – це інструмент, розроблений для того, щоб зробити комунікацію та обчислення більш доступними для вчених та людей які цікавляться наукою, надаючи їм знайомі інструменти у звичному середовищі, яке було створено під впливом таких інструментів, як Jupyter, R-Studio та MATLAB IDE. Підчас розробки Iodide було витрачено значну кількість зусиль для спрощення механізму поширення та відтворення роботи програмного коду на різних машинах. Це було зроблено за допомогою широкого використання сучасних

вебтехнологій таких як WebAssembly. WebAssembly дозволяє виконувати у веббраузерах код написаний на будь-якій мові програмування.

Iodide має повний доступ до інтерпретатора JavaScript у веббраузері, тому JavaScript, за бажанням, можна використовувати як просто для створення прототипів так і для експериментів у вебпрограмуванні. Iodide – це повністю інтерактивний вебдокумент, здатний запускати власний JavaScript (та інші мови через WebAssembly). Та має повний доступ до повного набору можливостей браузера. І на відміну від PDF-файлу, створеного за допомогою R-Markdown, одним кліком миші можна повернутися до коду, оновивши та змінивши його за бажанням.

Проект Iodide має сервери, які надають можливість збереження та розповсюдження та подальшого змінення розробленої програми. Сервер дозволяє створити профіль для збереження колекції програм, відстежувати зміни [9].

Також Iodide має функцію автозбереження, яка спрацьовує кожні 30 секунд підчас введення тексту програми. Нижче наведено зображення інтерфейсу Iodide (рис. 1.5).

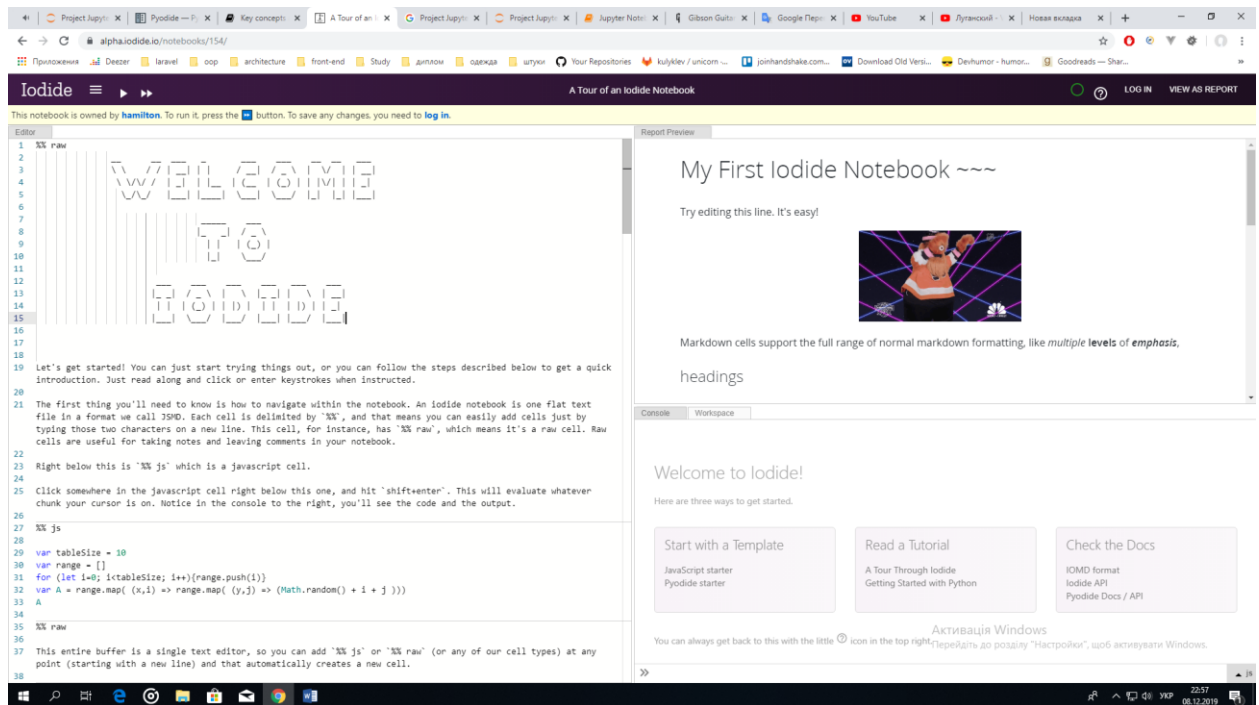


Рисунок 1.5 – Інтерфейс Iodide

Документи Iodide працюють у браузері, що означає, що обчислювальна машина завжди доступна, бо використовуються ресурси машини користувача. Щоразу, коли хтось ділиться своєю роботою, він ділиться інтерактивним звітом у реальному часі з запуском коду. Більше того, оскільки обчислення відбуваються у браузері, немає необхідності робити запит до сервера, щоб він виконав код у захищеному процесі. Це означає, що інтерактивні документи оновлюються в режимі реального часу, відкриваючи можливість безшовних тривимірних візуалізацій, навіть з низькою затримкою та високою частотою кадрів, необхідною для віртуальної реальності (ВР) [10].

1.3 Порівняння існуючих та розроблюваного програмних продуктів

Вище було наведено найбільш популярні на сьогоднішній день інтерактивні платформи для інтелектуального аналізу даних, які дозволяють виконувати обробку даних та ділитися своїми досягненнями з колегами або однодумцями. Але крім усіх своїх переваг, усі вони мають і недоліки.

В табл. 1.1 проведено більш наочне порівняння існуючих програмних продуктів (ПП) та програми, яка буде розроблена в ході виконання дипломної роботи.

Таблиця 1.1 – Порівняння програмних продуктів

Критерії порівняння	Програмні продукти			
	Jupyter Notebook	Apache Zeppelin	Iodide	Програма, що розробляється
Виконання програми на сервері	+	+	-	-
Виконання програми у браузері	-	-	+	+
Збереження програми	+	+	-	+
Просте відтворення коду на інших машинах	-	-	+	+
Завантаження даних з багатьох джерел	-	+	±	±
Простий та зручний інтерфейс	-	-	±	+

1.4 Формальна постановка задачі

Розроблюване ПЗ має відповідати наступним основним вимогам:

- запуск Python скриптів у веббраузері;
- підтримка популярних бібліотек для роботи з даними у Python;
- збереження розроблених програм;
- завантаження даних з файлів;
- відображення результатів виконання програми;
- авторизація для створення Python скриптів;
- запуск програми та відтворення її результатів на різних персональних комп'ютерах(ПК).

До вхідних даних ПЗ відносяться HTTP-запити та посилання на категорії книг в інтернет-магазинах.

До вихідних даних відносяться HTTP-відповіді, зібрані дані про книги та ціни на них.

1.5 Висновки за розділом 1

В першому розділі розглянуто особливості предметної області, проведено аналіз існуючих ПП та виявлено їх переваги і недоліки.

Також, було доведено актуальність теми та сформульовано основні вимоги до розроблюваної програми.

2 МАТЕРІАЛИ І МЕТОДИ

В першому розділі було проаналізовано існуючі рішення та їх методи виконання користувацьких Python скриптів.

Другий розділ буде присвячено детальному ознайомленню та дослідженню безпечних способів виконання користувацьких коду, а саме виконання коду на стороні клієнта та виконання коду на стороні сервера.

2.1 Виконання користувацького коду на стороні сервера

Для створення вебсервісу, який приймає код Python від користувачів і виконує його на сервері, існує ряд підходів, які можна застосувати для реалізації даного рішення. Зокрема, буде зосереджено увагу на тих, які захищають апаратну та програмну частину сервера від коду, який є зловмисним за випадковим збігом обставин чи дизайнерським задумом.

Перш за все, розробка вебсервісу, який приймає код Python від користувачів і запускає його на стороні сервера, сама по собі не є технічно складною задачею. Проблеми полягають у тому, як забезпечити вебсервіс можливістю успішного запуску коду користувача та як захистити вебсервіс від коду користувача.

Перший виклик, як забезпечити сервер можливістю успішного запуску коду користувача, можна перефразувати таким чином, як переконатися у тому, щоб користувачі подавали лише той код, який може бути успішно запущений на сервері. Найпростіший спосіб яким це можна зробити, опублікувавши інформацію про середовище, в якому сервер буде запускати код користувача. Наприклад, надати користувачам версію операційної системи, інтерпретатор та версію Python, доступні бібліотеки та версії. Цей підхід покладає на користувачів всі обов'язки гарантування того, що їх код може працювати у вказаному середовищі.

Це підхід, який застосовують, наприклад, сервіси з неперервної інтеграції, які публікують середовища доступні користувачам (наприклад, Travis CI, Circle CI, Gitlab CI або AppVeyor).

Другий і найважливіший виклик – це захист вебсерверу від коду користувача. Наприклад, у коді може бути помилка, через яку він потрапляє у нескінченний цикл, споживаючи ресурси центрального процесора сервера, на якому він працює; це може статися з випадковою реалізацією нескінченної рекурсії, яка поглинає пам'ять сервера; або, код може бути написаний зловмисником і записувати великі файли сміття, забираючи доступний простір на диску. Як ми можемо запобігти впливу на сервер користувацького коду, на якому він працює, і таким чином забезпечити доступність вебсервіс або забезпечити безперебійне виконання коду від інших користувачів? Існує багато варіантів захисту і низка з них наведена нижче [11]:

- використання `pysandbox`;
- використання пісочниці `PyPy Python`;
- використання можливостей операційної системи (ОС) `Linux`;
- використання інструменту для контейнеризації `Docker`;
- створення мультисерверної архітектури.

2.1.1 Використання `pysandbox`

`Pysandbox` – це реалізація пісочниці в `Python`. Він призначений для запуску ненадійного коду `Python` (у нашому випадку коду, що подається користувачем) у обмеженому середовищі, яке може бути налаштоване розгортачем `pysandbox`. Наприклад, розгортач може заборонити не довіреному коду доступ до файлової системи та стандартних каналів вводу/виводу та запобігти імпортуванню модулів `Python`. `Pysandbox` також дозволяє розгортачу встановлювати верхню межу часу роботи ненадійного коду, використання пам'яті, обсягів вводу/виводу даних та обмежувати глибину рекурсивних викликів функцій. Однак у 2013 році розробник `pysandbox` Віктор Стіннер та

співтовариство дійшли висновку, що "проєкт pysandbox зламаний за дизайном". Тобто не є безпечним. З того часу альтернатива з аналогічним підходом так і не з'явилась, або не набрала популярності і її не знайти.

2.1.2 Використання пісочниці PyPy Python

Реалізація PyPy Python забезпечує альтернативну модель пісочниці Python. Пісочниця PyPy пропонує пісочницю на рівні, порівнянному з тим, який пропонують операційні системи. Довірена програма Python (наприклад, наш вебсервіс або програма, яку вона викликає) створює під-процес, який запускає ненадійний код (наприклад, код, надісланий користувачем), використовуючи ізольовану версію PyPy. Ця версія PyPy серіалізує всі вхідні/вихідні дані до стандартного вхідного/вихідного каналу. Потім довірена програма Python визначає, який доступ до вводу-виводу дозволений чи ні. Також може бути встановлений контроль над обсягом процесорного часу та оперативної пам'яті, який може використовувати ненадійний код.

Нажаль, цей підхід вважається експериментальним самими його розробниками, може не працювати з PyPy3 та PyPy інтерпретатором Python 3.

2.1.3 Використання можливостей операційної системи Linux

Існують різні функції ОС Linux, які можна використовувати для контролю виконання процесів, обмеження файлів та інших ресурсів вводу-виводу, до яких має доступ процес, а також обмеження використання центрального процесора та споживання пам'яті. Дані функції можуть використовуватися для контролю виконання коду, поданого користувачами. Перелік таких функцій:

- chroot – може бути використаний для обмеження частин файлової системи, доступ до яких не має кореневий процес та будь-які підпроцеси, які він

породжує. Процес не може отримати доступ до файлів за межами цієї обмеженої файлової системи, яка називається "в'язницею chroot";

- ulimit – це команда, яка може встановити обмеження на центральний процесор, кількість процесів, кількість відкритих файлів та пам'ять, доступну користувачеві;

- seccomp – це ядро, яке може ізолювати процес від системних ресурсів, дозволяючи йому отримувати доступ лише до дескрипторів відкритих файлів і виходити. Якщо процес намагається здійснити будь-який інший системний виклик (наприклад, відкрити інший файл), він вбивається. seccomp використовується Docker;

- seccomp-bpf надає додаткову гнучкість seccomp. Це дозволяє писати програмні фільтрування, які визначають, які системні виклики повинні бути доступними та яким процесам;

- AppArmor – це модуль безпеки ядра, що дозволяє контролювати доступ до мережевих, сокетних та файлових ресурсів, щоб бути налаштованим та застосованим для конкретних програм;

- SELinux – це модуль безпеки ядра для контролю доступу, який виконує функцію, подібну до AppArmor, хоча і з більш багатою, складною конфігурацією.

2.1.4 Використання інструменту для контейнеризації Docker

Docker використовує функції ізоляції та віртуалізації ядра Linux, щоб дозволити незалежним «контейнерам» працювати на одному сервері Linux. Кожен контейнер Docker пропонує базову версію Linux. Docker можна налаштувати для обмеження ресурсів контейнера, включаючи його центральний процесор та ресурси пам'яті.

Можна створити вебсервер, який буде отримувати код від користувача, запускати контейнер Docker, який запускає код користувача, отримує результати, повертає їх користувачеві та вимикає контейнер. Прикладом цієї

моделі на практиці є сервіс «Remote Interview», який використовує Docker для надання можливості кандидатам на співбесіді запускати свій код. Код користувача працює в контейнері Docker з обмеженими ресурсами.

Деякі розробники висловлювали занепокоєння безпекою Docker. Бо контейнери мають спільне ядро, інші компоненти хост-операційної системи, і вони мають root-доступ. Контейнери менш ізольовані один від одного, ніж віртуальні машини, і якщо в ядрі є уразливість, це може поставити під загрозу безпеку інших контейнерів так само. Docker хороший для досягнення ізоляції, але не стільки з точки зору безпеки.

2.1.5 Створення мультисерверної архітектури

Створення архітектури, згідно з якою вебслужба працює на одному сервері (або на фізичній, або на віртуальній машині), а код кожного користувача виконується на окремому сервері (знову ж таки, фізичному чи віртуальному) значно зменшує або навіть усуває ризик того, що код одного користувача впливає або на роботу вебсерверу, або на код інших користувачів. У цьому випадку вебсервер виконує функції диспетчера, копіюючи код користувача на сервер, контролюючи стан його виконання та повертаючи результати користувачам. Існує багато способів, як це можна реалізувати. Невеликий їх перелік такий:

- безпечна оболонка (від англ. Secure shell, SSH) може викликати команди `bash` на віддалених серверах, а захищена передача файлів (SFTP) або захищене копіювання (SCP) може використовуватися для передачі файлів на сервери та з них;
- фреймворк Paramiko – це реалізація SSH2 на Python з відкритим кодом, що включає модуль SFTP;
- Fabric – це фреймворк Python з відкритим кодом для виконання локальних або віддалених команд оболонки та завантаження / завантаження файлів на / з віддалених хостів;

- продукти з відкритим кодом, включаючи SLURM, Kubernetes, Torque або HTCondor, підтримують планування завдань та управління навантаженням у кластері фізичних або віртуальних машин.

2.2 Виконання користувацького коду на стороні клієнта

Для створення застосунку, який приймає код Python від користувачів і виконує його на стороні клієнта у веббраузері, існує також ряд підходів, які можна застосувати. Найбільша перевага всіх цих підходів на відміну від підходів виконання користувацьких Python скриптів на стороні вебсервера – це відсутність проблем з впровадженням захисту апаратної та програмної частини сервера від коду, який може бути зловмисним, і придбання великих обчислювальних потужностей.

Натомість є низка інших задач, які треба вирішити:

- запуск програмного коду на мові програмування відмінної від JavaScript;
- портування бібліотек;
- малий трансфер даних через мережу.

Для запуску користувацьких Python скриптів у веббраузері існує два способи:

- компіляція Python до JavaScript;
- створення середовища (runtime) Python;

2.2.1 Компіляція Python до JavaScript

Оскільки, всі сучасні веббраузери за замовчуванням гарно підтримують JavaScript, то одним із найпростіших способів змусити працювати код, написаний за допомогою мови програмування Python – це скомпілювати код на Python до JavaScript. Нижче буде описано підхід як це робиться.

Для компіляції Python до JavaScript необхідно створити мініфіковане об'єднання окремих скриптів, які обробляють конкретні завдання, або під час компіляції, або під час виконання.

Кореневий скрипт компілятора повинен створювати 2 змінні у глобальному просторі імен JavaScript: одна змінна має вмістити функцію, що викликається при завантаженні сторінки, а друга має містити об'єкт, що містить усі внутрішні об'єкти, необхідні для запуску скриптів Python.

Для компіляції Python до JavaScript необхідно виконати такі дії:

- токенизатор повинен прочитати токени у вихідному коді на Python та передати їх автомату, який створює абстрактне дерево коду, або викликає `SyntaxError` або `IndentationError`;
- це дерево необхідно трансформувати (вузли додаються або модифікуються) для перекладу деяких окремих операторів Python у ряд операторів JavaScript;
- додаткові вузли додаються для оновлення внутрішнього об'єкта;
- перетворене дерево вже може повертати код на JavaScript.

Об'єкти Python, можна реалізувати як власні об'єкти JavaScript. Рядки Python можна реалізувати як рядки JavaScript. Списки та кортежі Python реалізуються як масиви JavaScript.

Цілі числа Python можна реалізувати як цілі числа JavaScript, якщо вони знаходяться в діапазоні "безпечних цілих чисел" JavaScript, тобто мають значення $[-(2^{53}-1), 2^{53}-1]$; поза цим діапазоном їх необхідно реалізувати за допомогою внутрішнього класу. Змінні типу `float` у Python можна трансформувати як екземпляри класу JavaScript `Number`.

Усі інші класи Python (вбудовані або визначені користувачем) трансформуються як об'єкт JavaScript, що містить атрибути та методи класу.

Функції Python перетворюються у функції JavaScript, але цей процес досить складний, бо є багато відмінностей як у визначенні функції, так і у викликах функцій.

Для визначення функції Python її параметри можна задати різними способами:

- іменованими параметрами;
- зі значеннями за замовчуванням;
- упакованими ітераторами;
- упакованими словниками.

Функцію Python можна викликати з:

- позиційними аргументами: `f(2)`;
- аргументами ключових слів: `f(y = 1)`;
- упакованими словниками: `f(**kw)`.

JavaScript також має різні способи вказання параметрів:

- іменовані параметри;
- обробка аргументів за допомогою членів об'єкта, які можна

використовувати всередині функції, більш-менш схожими на список.

Виклики функцій можуть виконуватися з іменованими аргументами, або за допомогою методів `call()` та `apply()`.

Для викликів функцій аргументи, передані функції Python, перекладаються таким чином:

- позиційні аргументи залишаються незмінними;
- упаковані кортежі розпаковуються та додаються до позиційних аргументів;
- всі аргументи з ключовими словами (включаючи упаковані словники) згруповані в одному аргументі, розміщеному в кінці списку аргументів.

Програма Python розділена на блоки: модулі, функції, класи та вирази. Для кожного блоку необхідно визначати змінну JavaScript, яка буде містити всі імена, зв'язані у блоці.

На основі лексичного аналізу можна дізнатися, в якому блоці доступне (видно) ім'я. Воно перекладається як однойменний атрибут об'єкта.

Для виконання програми необхідно визначити рамки виконання за допомогою стеку. Кожного разу, коли програма вводить новий модуль або нову функцію (включаючи лямбда-функції та вирази), інформація про глобальне та локальне середовище розміщується поверх стека; коли функція або модуль виходить, елемент зверху стека видаляється. У згенерованому JavaScript виконується вхід та вихід до кожної рамки виконання [12].

Описаний підхід цілком придатний для компіляція користувацького коду Python до JavaScript, але має суттєві недоліки, коли мова йде про компіляцію бібліотек. Адже вони дуже часто написані на C, який не можливо в повній мірі скомпілювати до JavaScript.

2.2.2 Створення середовища (runtime) Python

Для вирішення проблеми компіляції бібліотек Python можна скористатися новим сучасним інструментом, які підтримують усі сучасні веббраузери – Web Assembly.

Web Assembly – це технологія, що надає новий тип коду, який можна запускати в сучасних веббраузерах, що забезпечує нові функції і значне підвищення продуктивності. Код для WebAssembly не призначений для написання вручну, скоріше він спроектований для ефективної компіляції з низькорівневих вихідних мов, таких як C, C ++, Rust та інші. Втім він також підходить для високорівневих мов таких як Python.

WebAssembly дозволяє запускати код, написаний на різних мовах у вебзастосунках майже з природною швидкістю. Це має величезне значення для вебплатформи, так як раніше це було неможливо зробити.

Є кілька ключових понять, необхідних для розуміння того, як WebAssembly працює у браузері:

- модуль – являє двійковий файл WebAssembly, який було скомпільовано браузером в виконуваний машинний код. Модуль не має стану і,

таким чином, як Blob, може бути явним чином розділений між windows і workers. У модулі є оголошення імпорту та експорту точно таке ж, як і в модулі ES2015;

- пам'ять – масив ArrayBuffer змінюваного розміру, який містить лінійний масив байтів, зчитувальних і записуються інструкціями низькорівневого доступу до пам'яті в WebAssembly;

- таблиця – типізований масив посилань змінюваного розміру (наприклад, для функцій), які не можуть бути розміщені в пам'яті у вигляді байтів (з міркувань безпеки і портативності);

- примірник – модуль в парі зі своїм станом, який він використовує під час виконання, включаючи пам'ять, таблицю і набір імпортованих значень.

JavaScript API надає розробникам можливість створювати модулі, об'єкти пам'яті, таблиці і екземпляри модулів. Отримавши екземпляр модуля WebAssembly, код JavaScript може синхронно викликати його екпорти, які представляються як звичайні функції JavaScript. Будь-які функції JavaScript також можуть синхронно викликатися кодом WebAssembly шляхом передачі цих функцій в якості імпорту в екземпляр модуля WebAssembly [13].

Таким чином для того щоб запустити код на Python з його бібліотеками за допомогою WebAssembly необхідно зробити декілька кроків:

- завантажити компілятор типу код-у-код – Emscripten;
- завантажити вихідний коду інтерпретатора Python (CPython) та наукових обчислювальних пакетів (NumPy тощо);
- застосувати невеликі зміни, щоб змусити Python працювати в нових умовах;
- скомпілювати все до WebAssembly за допомогою компілятора emscripten.

Emscripten — компілятор типу код-у-код або транскompілятор. На вході він приймає LLVM-байткод, отриманий компіляцією сирцевого коду мовою C або C++. На виході він видає файл з кодом мовою JavaScript, що може бути запуснений у вебпереглядачі [14].

Якщо просто взяти скопійований у WebAssembly код і завантажити його у браузер, для інтерпретатора Python все виглядало б зовсім інакше, ніж при запуску безпосередньо поверх операційної системи. Бо веб-браузери не мають файлової системи. На щастя, emscripten надає віртуальну файлову систему, написану на JavaScript, яку може використовувати інтерпретатор Python. За замовчуванням ці віртуальні "файли" знаходяться в тимчасовій пам'яті на вкладці браузера, і вони зникають при виході з сторінки.

Емулюючи файлову систему та інші особливості стандартного обчислювального середовища, emscripten робить можливим переміщення існуючих проєктів у веббраузер.

2.3 Порівняння методів запуску користувацьких Python-скриптів

В табл. 2.1 наведено порівняння розглянутих методів виконання користувацьких скриптів на мові програмування Python.

Таблиця 2.1 – Порівняння методів виконання користувацьких скриптів на мові програмування Python.

Методи	Критерії			
	Місце виконання	Час виконання	Складність реалізації	Вимоги до ресурсів
Використання pysandbox	Сервер	ЛІТ	+	Середні
Використання пісочниці PyPy Python	Сервер	ЛІТ	++	Середні
Використання можливостей ОС Linux	Сервер	ЛІТ	+++	Середні
Використання Docker	Сервер	ЛІТ	++	Високі
Створення мультисерверної архітектури	Сервер	ЛІТ	+++	Високі
Компіляція Python до JavaScript	Клієнт	АОТ	++	Низькі
Створення runtime Python	Клієнт	ЛІТ	++	Низькі

2.4 Висновки за розділом 2

У другому розділі проаналізовано методи запуску коду написаного за допомогою Python на вебсервері та у веббраузері, розглянуто можливі складнощі з якими можна стикнутися та шляхи їх вирішення.

3 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

Даний розділ присвячений опису та обґрунтуванню програмно-технічних рішень, які були прийняті під час створення ПЗ.

Розроблене ПЗ складається з двох основних частин:

- користувацький інтерфейс;
- інтерфейс прикладного програмування (від англ. Application programming interface, API).

Користувацький інтерфейс відповідає за відображення даних, виконання Python скриптів, створення скриптів для обробки даних.

Модуль API відповідає за надання, обробку, збереження даних, взаємодію з базою даних (БД).

3.1 Опис основних вимог до програми

Головне завдання програмного забезпечення, що розробляється: реалізувати інтерактивну вебплатформу для інтелектуального аналізу даних. Програмна реалізація повинна надавати інструменти для завантаження та аналізу даних за допомогою вебтехнологій.

Серед функціональних вимог до системи можна виділити:

- авторизація користувача;
- завантаження користувацьких вибірок даних;
- виконання користувацьких Python скриптів в ізольованому середовищі;
- створення, оновлення та видалення Python скриптів лише для авторизованих користувачів;
- виведення повідомлення з консолі;
- побудова графіків.

Для забезпечення надійної роботи системи для голосування, всі дані про голоси голосувань зберігаються в блокчейн ланцюгу, та дублюються по всім

вузлам мережі блокчейн. Для забезпечення простого та зручного доступу користувача до функцій голосування головний вузол мережі має централізовану базу даних, в якій зберігаються дані користувача.

3.2 Проектування структури програмної системи

Вибір архітектури розроблюваного ПЗ залежить від місця, де будуть виконуватися Python скрипти. Вони у можуть виконуватися або у клієнта в браузері, або на віддаленому сервері. Кожен з цих варіантів має значну кількість як переваг, так і недоліків. Вибір того чи іншого підходу ґрунтується лише на задачах, які повинна виконувати розроблювана система. Головні чинники, які впливали на обрання технології, описані нижче.

До переваг виконання Python скриптів у браузері користувача можна віднести:

- відносна простота реалізації;
- відсутність потреби у потужному сервері;
- мала кількість даних, яку треба передавати між клієнтом та сервером;
- легко відтворити результати програми на будь-якій машині в будь-якій точці світу з доступом до всесвітньої мережі Інтернет.

Але даний підхід також має свої недоліки:

- великий час завантаження сторінки вебсайту, бо необхідно завантажити багато даних для виконання Python;
- неможливо створити розподілену кластерну систему для виконання надскладних задач.

Усіх цих недоліків позбавлений спосіб виконання Python скриптів на віддаленому сервері. Але цей підхід потребує значно складнішої інфраструктури системи. Бо, з міркувань безпеки, користувацький код не можна просто взяти і запустити на сервері. Необхідно створювати ізольовані процеси, що в свою чергу накладає деякі обмеження. Також час на відображення результату програми

може бути значно більший, бо доведеться робити http запити з великим об'ємом даних. Крім того, у разі великої завантаженості системи, необхідно буде запускати черги на виконання користувацьких програм, що створить додаткову затримку в отриманні результатів. І для всього цього буде необхідний потужний сервер, або кластер серверів, щомісячне утримання якого буде коштувати дуже дорого.

Зваживши всі переваги та недоліки кожного з підходів реалізації поставленої задачі, було прийняте рішення про використання підходу виконання Python скриптів у браузері користувача.

Оскільки, розроблюваний ПП є вебзастосунком, для побудови архітектури було використано архітектурний шаблон модель-вид-контролер(від англ. Model-View-Controller, MVC), а саме ПЗ розділено на дві частини:

- фронтенд (від англ. front-end);
- бекенд (від англ. back-end).

Які в своє чергу будуть взаємодіяти між собою за допомогою API за стандартом передачі репрезентативного стану (від англ. Representational state transfer, REST). Загальна архітектура розроблюваного ПЗ представлена на рис. 3.1.

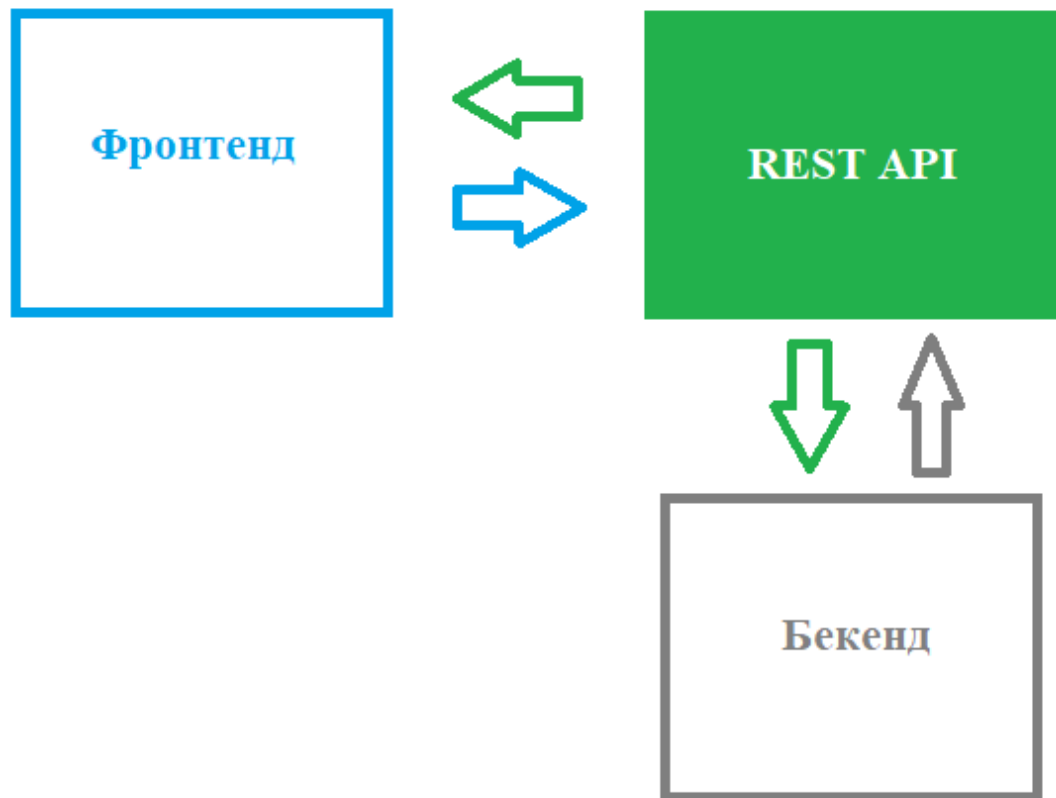


Рисунок 3.1 – Загальна архітектура системи

Вся інфраструктура ПП буде контейнеризована за допомогою Docker та буде розгортатися з використанням Docker-compose.

3.2.1 Структура бекенду

При проектуванні структури майбутнього бекенду платформи було виділено 3 основних програмних модуля, які представлено на рис. 3.2.



Рисунок 3.2 – Основні програмні модулі бекенду

Розробку представлених модулів можна аргументувати тим, що саме розділення основного функціоналу програмного комплексу за таким принципом дозволяю сформувати функціонально завершені блоки програми, які не є тісно пов'язані між собою. Розглянемо докладніше функціонал кожного з модулів.

До модулю обробки Python скриптів відносяться:

- функції збереження вхідних даних у проміжковому форматі для подальшої роботи із ними;
- функції взаємодії з модулем зв'язку із БД.

До модулю модулем зв'язку із БД відносяться:

- функції запису даних до БД;
- функції зчитування даних із БД;
- функції видалення даних із БД.

До модулю керування користувачами відносяться:

- функції авторизації користувачів;
- функції автентифікації користувачів.

3.2.2 Структура фронтенду

Підчас проектуванні структури майбутнього фронтенду платформи було прийнято рішення створювати його у вигляді одно сторінкового застосунку (від англ. single-page application, SPA), який матиме 4 основні програмні модуля, які представлено на рис. 3.3.



Рисунок 3.3 – Основні програмні модулі фронтенду

Завдяки такому розподіленню основного функціоналу програмного комплексу по модулям дозволяє досягти низького рівня зв'язності між компонентами, що спрощує розробку та підтримку ПП. Більш детальний огляд функціонал кожного з модулів наведено нижче.

До модулю виконання Python скриптів відносяться:

- функції запуску та виконання Python скриптів;
- функції відображення результатів виконання Python скриптів.

До модулю взаємодії з бекендом відносяться:

- функції виконання запитів до бекенду;
- функції парсингу даних отриманих з бекенду;
- функції формування запитів до бекенду.

До модулю зв'язку із вебсховищем відносяться:

- функції збереження даних отриманих з бекенду;
- функції запису даних до вебсховища;
- функції зчитування даних із вебсховища;
- функції видалення даних із вебсховища.

До модулю роутингу відносяться:

- функції формування URL-адресів сторінок;
- функції контролю доступу до сторінок;
- функції перемикання сторінок.

3.3 Моделювання логіки програми

Далі на діаграмах діяльності наведено логіку роботи та дії, що виконується в платформі для інтелектуального аналізу даних.

Діаграму діяльності, що демонструє авторизацію користувача наведено на рис. 3.4.

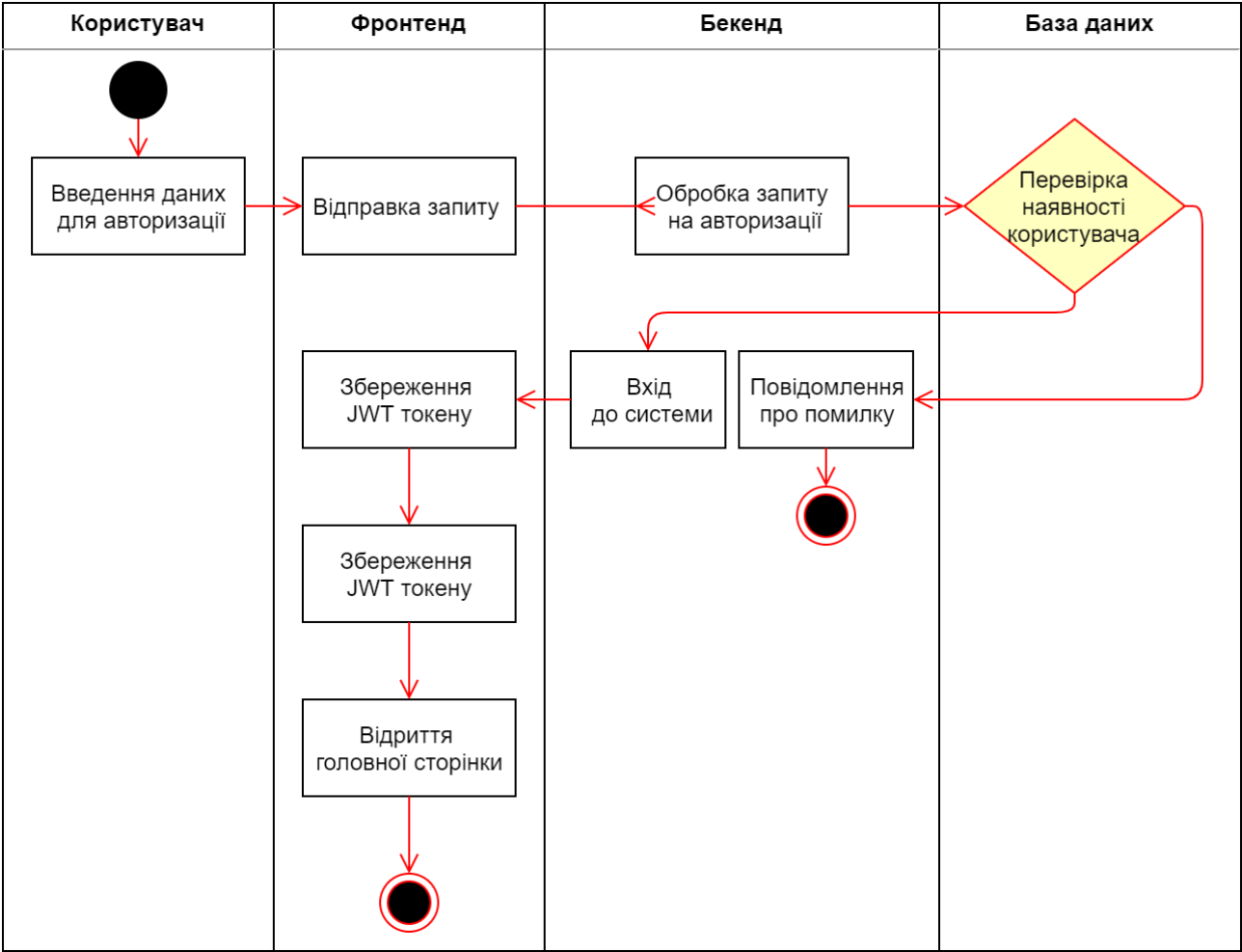


Рисунок 3.4 – Діаграма діяльності, що демонструє авторизацію користувача в нотації UML 2.0

Як видно з рисунку наведеного вище, м робота із вебплатформою починається з авторизації користувача. Після успішної перевірки даних користувач може перейти до повноцінного застосунку.

На рис. 3.5 наведено діаграму діяльності, що демонструє процес створення, використання та збереження нового скрипта для аналізу даних.

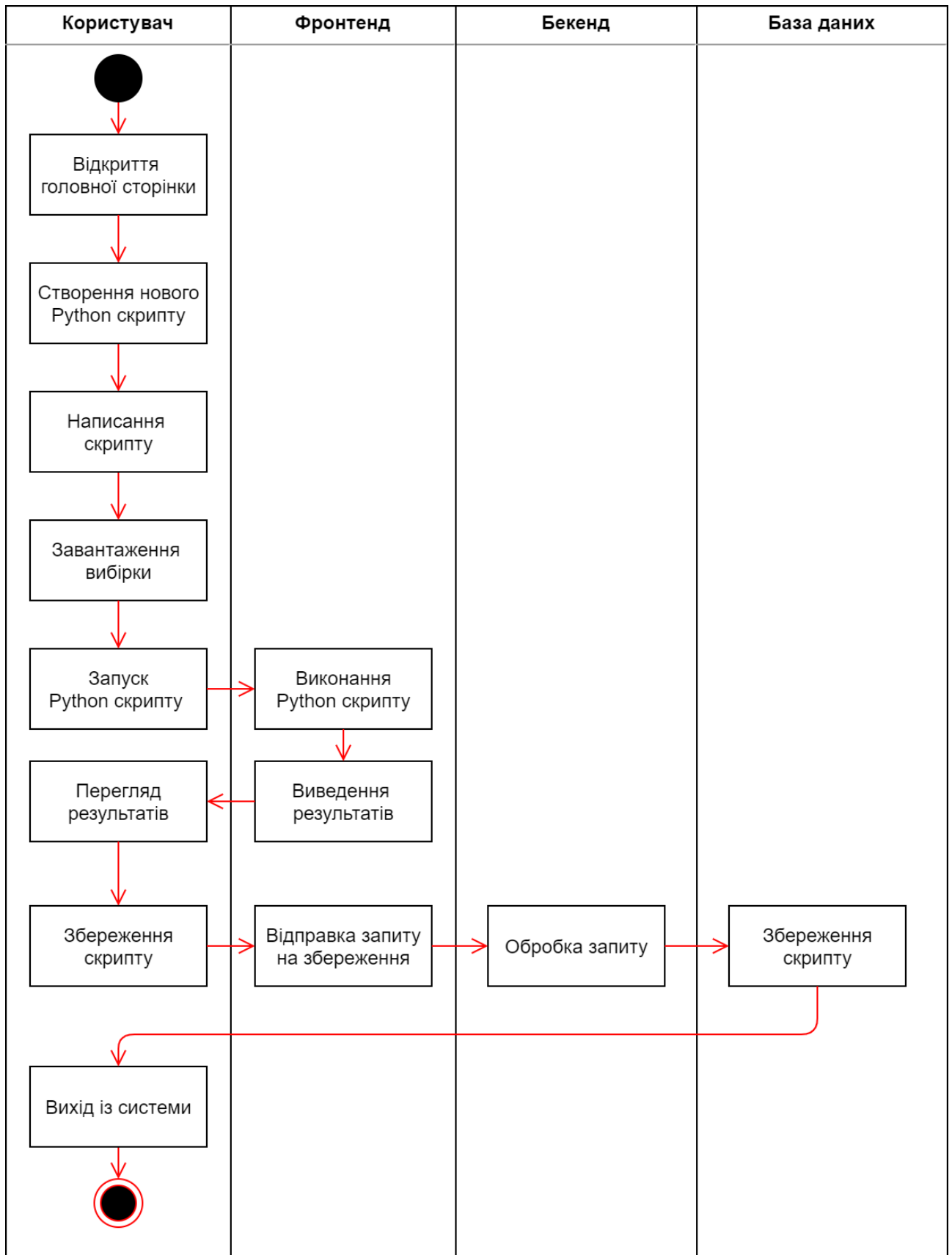


Рисунок 3.5 – Діаграма діяльності, що демонструє створення нового скрипта в нотації UML 2.0

Як видно з рисунку, після відкриття головної сторінки платформи та створення нового скрипта користувачем, йому необхідно написати цей самий скрипт. Після чого необхідно завантажити дані та запустити його, щоб впевнитися у працездатності програми. Затим треба переглянути результат на коректність. Далі виконується збереження та вихід з системи.

На рис. 3.6 наведено діаграму діяльності, що демонструє процес виконання раніше збереженого Python скрипта.

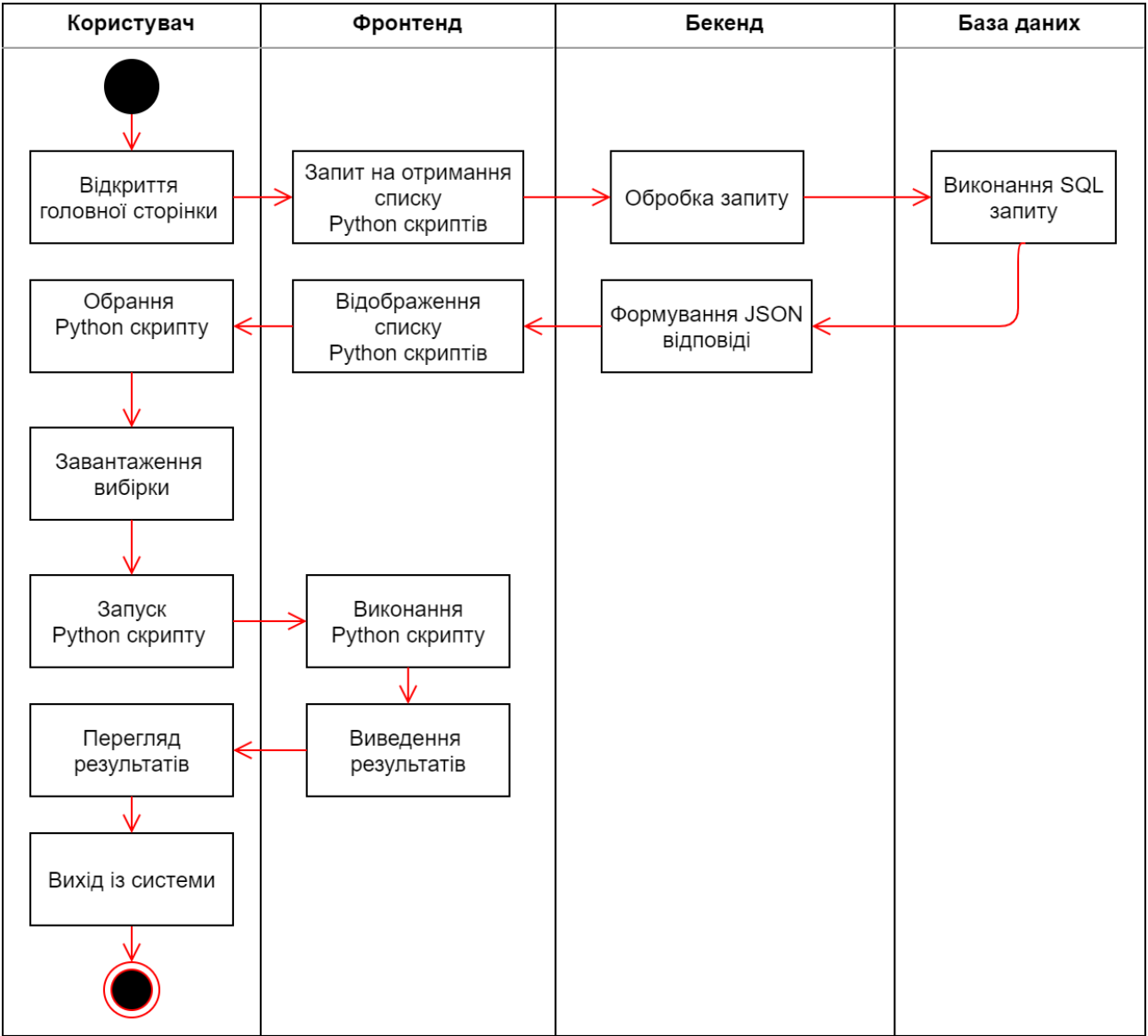


Рисунок 3.6 – Діаграма діяльності, що демонструє використання збереженого скрипта в нотації UML 2.0

Як видно з рисунку наведеного вище, для виконання збереженого скрипта необхідно на головній сторінці обрати скрипт зі списку Python скриптів, які завантажуються одразу після відкриття головної сторінки. Після завантаження даних та запуску даних, фронтенд виконує обраний скрипт та відображає користувачу результати роботи.

3.4 Проектування бази даних

Також для подальшої роботи слід розробити концептуальну модель бази даних, яка буде використовуватися для роботи. Концептуальна модель майбутньої бази даних демонструється на рис. 3.7.

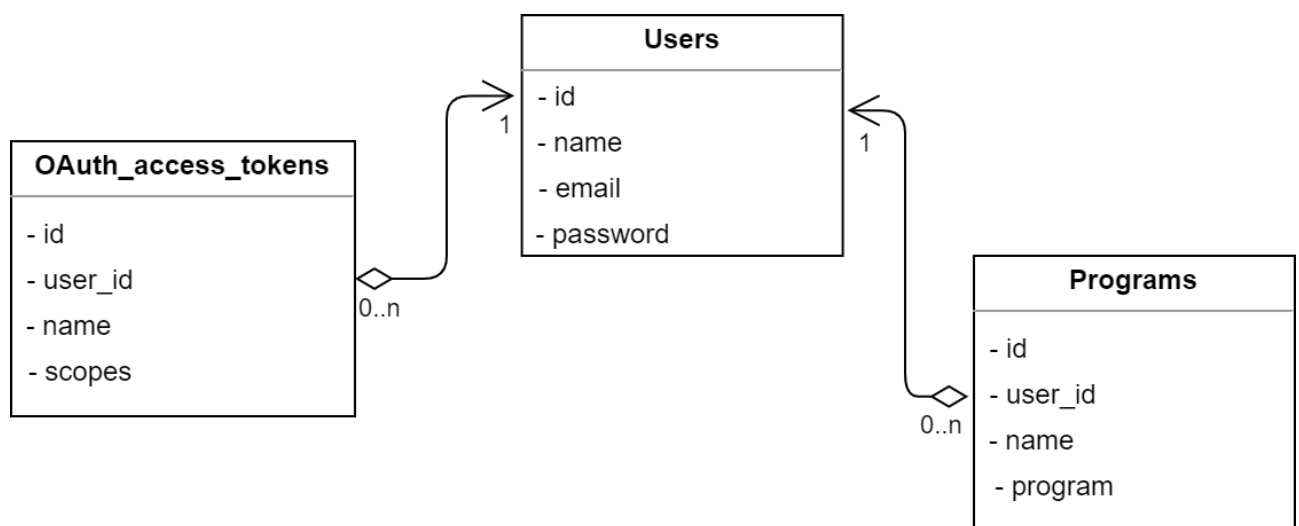


Рисунок 3.7 – Концептуальна модель бази даних

Як видно зі схеми для зберігання даних програми буде достатньо трьох таблиці: User, OAuth_access_tokens, Programs. Розглянемо представлені сутності детальніше:

- Users (id, name, email, password) – сутність, що містить інформацію про користувачів системи. Поля сутності: id – номер користувача (унікальний ідентифікатор); name – ім'я або логін користувача; email – имейл адреса скриньки користувача; password – пароль користувача.

- OAuth_access_tokens (id, user_id, name, scopes) – сутність, що містить інформацію про JWT токени користувачів. Поля сутності: id – номер токenu (унікальний ідентифікатор), user_id – номер пов’язаного користувача, name – ім’я токenu, scopes – область дії токenu.
- Programs (id, user_id, name, program) – сутність, що містить збережені скрипти. Поля сутності: id – номер скрипту (унікальний ідентифікатор), user_id – номер автора, name – назва скрипту, program – код скрипта.

3.5 Вибір мови програмування

Оскільки розроблюване ПЗ складається з двох незалежних частин, вирішують дві різні задачі та працюють у різних середовищах – для кожної частини мова програмування буде обиратися окремо.

3.5.1 Мова програмування Python

В якості мови програмування для написання бекенду (вебсервера) було обрано Python.

Python – інтерпретована мова програмування високого рівня та загального призначення. Створена Гідо ван Россумом і вперше випущена в 1991 році, філософія дизайну Python підкреслює читабельність коду завдяки своєму помітному використанню значних пробілів. Його мовні конструкції та об’єктно-орієнтований підхід мають на меті допомогти програмістам писати чіткий логічний код для малих та великих проєктів [15]. Високорівневі вбудовані структури даних у поєднанні з динамічною типізацією та динамічним зв’язуванням роблять її дуже привабливою для швидкої розробки додатків, а також для використання в якості мови сценаріїв або «клею» для з’єднання існуючих компонентів. Простий, легкий у вивченні синтаксис Python підкреслює читабельність і, отже, зменшує витрати на обслуговування програми. Python підтримує модулі та пакети, що заохочує робити програми модульними та

повторно використовувати код. Інтерпретатор Python та велика стандартна бібліотека доступні у вихідній або двійковій формі безкоштовно для всіх основних платформ і можуть вільно розповсюджуватися.

Часто програмісти обирають Python через підвищену продуктивність, яку він забезпечує. Оскільки не існує кроку компіляції, цикл редагування-тестування-налагодження неймовірно швидкий. Налагодження програм Python дуже просто: помилка або неправильний ввід ніколи не спричинить помилку сегментації. Натомість, коли інтерпретатор виявляє помилку, виникає виняток. Коли програма не вловлює винятків, інтерпретатор друкує трасування стека. Налагоджувач рівня джерела дозволяє перевіряти локальні та глобальні змінні, оцінювати довільні вирази, встановлювати точки зупинки, переходити через код за рядком за один раз тощо. Налагоджувач написаний на самому Python, що свідчить про інтроспективну силу Python. З іншого боку, часто найшвидшим способом налагодження програми є додавання кількох операторів друку до джерела: швидкий цикл редагування-тестування-налагодження робить цей простий підхід дуже ефективним [16].

Інтерпретатори Python доступні для багатьох операційних систем. Світова спільнота програмістів розробляє та підтримує реалізацію CPython, яка є безкоштовною та з відкритим сирцевим кодом. Некомерційна організація, Фонд програмного забезпечення Python, управляє та направляє ресурси для розробки Python та CPython.

На відміну від багатьох інших мов, Python не обов'язково звільняє пам'ять. Натомість він має виділений розподільник об'єктів для об'єктів розміром менше 512 байт, який зберігає деякі фрагменти вже виділеної пам'яті для подальшого використання в майбутньому. Обсяг пам'яті, який зберігає Python, залежить від моделей використання. У деяких випадках вся виділена пам'ять може бути звільнена лише тоді, коли процес Python завершується.

Якщо тривалий процес Python з часом забирає більше пам'яті, це не обов'язково означає, що є витоки пам'яті [17].

У табл. 3.1 наведено порівняння обраної мови програмування із іншими можливими альтернативами.

Таблиця 3.1 – Порівняння мов програмування для вебсервера

Критерії	Мови програмування			
	Python	PHP	JavaScript	C#
Динамічна типізація	+	+	+	+
Явна типізація	±	±	-	+
Неявна типізація	+	+	+	±
Багатоплатформність	+	+	+	-
Підтримка інтерфейсів	+	+	±	+
Декларація типів та повертаємих значень	+	+	-	+

Як видно з таблиці мова програмування Python має усі необхідні функції для сучасної веброзробки. Іншим переконливим аргументом на користь використання Python є те, що розроблюється платформа з інтелектуального аналізу даних для Python. Тому з огляду на можливе майбутнє розширення та вдосконалення системи доцільно зробити саме такий вибір.

3.5.2 Вибір мови програмування для розробки фронтенду

Для розробки фронтед частини було обрано мову програмування JavaScript, бо вона на сьогоднішній день є майже єдиним рішенням за допомогою якого можна розробляти застосунки, які можуть працювати у веббраузері.

JavaScript – це легка, інтерпретована, об'єктно-орієнтована мова з першокласними функціями і найвідоміша як мова сценаріїв вебсторінок, але вона використовується і в багатьох середовищах, що не належать до браузера. Вона прототипоорієнтована, багатопарадигмна мова сценаріїв, яка є динамічною та підтримує об'єктно-орієнтований, імперативний та функціональний стилі програмування [18].

Єдиною альтернативою може бути TypeScript, але він по суті є обгорткою над JavaScript, яка просто додає деякий «синтаксичний цукор». Також є мова WebAssembly, але воно не є альтернативою JavaScript, а скоріше доповненням.

3.6 Вибір фреймворку для розробки

Через те, що розроблюване ПЗ складається з двох незалежних частин, вирішують дві різні задачі та працюють у різних середовищах – для кожної частини фреймворк буде обиратися окремо.

3.6.1 Обрання Python фреймворку для реалізації бекенду

В якості Python фреймворку було обрано FastAPI.

FastAPI – це сучасний, швидкий, високопродуктивний вебфреймворк для побудови API з Python 3.6+ на основі стандартних підказок типу Python.

Основні особливості:

- швидкий: Дуже висока продуктивність, нарівні з NodeJS та Go (завдяки Starlette та Pydantic). Один з найшвидших доступних фреймворків Python;
- швидкий у розробці: збільшує швидкість розробки функцій приблизно на 200% до 300%;
- зменшує кількість помилок: зменшує до 40% помилок, спричинених людиною (розробником);
- інтуїтивно зрозумілий: відмінна підтримка редактора, всюди працює автодоповнення, витрачається менше часу на налагодження;
- простий: побудований так, щоб бути простим у використанні та вивчені;
- стислий: мінімізує копіювання коду;
- надійний: постачає готовий до реального використання код. З автоматичною інтерактивною документацією;

– заснований на стандартах: базується на (і повністю сумісний) відкритих стандартах для API: OpenAPI (раніше відомий як Swagger) та схема JSON [19].

Порівняльна характеристика найпоширеніших фреймворків мови Python наведена у табл. 3.2.

Таблиця 3.2 – Порівняння розглянутих Python фреймворків

Критерії	Фреймворки		
	Fastapi	Django	flask
ORM за замовчуванням	SqlAlchemy	Django ORM	немає
Перший випуск	2019	2005	2010
Мікрофреймворк	+	-	+
Зосереджений на створення API	+	±	+
Активне використання можливостей Python 3.6+	+	-	-
Вбудована асинхронність	+	-	+

В якості фрейворку для розробки бекенкду було обрано Fastapi, тому що він використовує як перевірені часом та розповсюджені рішення такі як SqlAlchemy в якості ORM, так і нові можливості, які надає сучасний Python.

3.6.2 Обрання JavaScript фреймворку для реалізації фронкенду

Сьогодні існує величезна кількість фреймворків для JavaScript та JavaScript-подібних мов програмування, які значно спрощують створення користувацького інтерфейсу для вебсайту.

Оскільки, існує відома проблема з запуском Python скриптів у браузері – великий час завантаження. Тому, було прийняте рішення розроблювати single-page application(SPA), яке допоможе в значній мірі вирішити цю проблему. Бо довге завантаження буде тільки під час першого відкриття сайту.

SPA – це вебзастосунок чи вебсайт, який вміщується на одній сторінці з метою забезпечити користувачу досвід близький до користування настільною програмою. В односторінковому застосунку весь необхідний код - HTML, JavaScript, та CSS - завантажується разом зі сторінкою, або динамічно довантажується за потребою, зазвичай у відповідь на дії користувача. Сторінка не оновлюється і не перенаправляє користувача до іншої сторінки у процесі роботи з нею. Взаємодія з односторінковим застосунком часто включає в себе динамічний зв'язок з вебсервером [20].

Найбільш розповсюджені фреймворки для створення SPA застосунків – це Vue.js, Angular та React.

В табл. 3.3 наведено порівняння фронтенд фреймворків.

Таблиця 3.3 – Порівняння фронтенд фреймворків

	Vue.js	Angular	React
Приблизний розмір фреймворку	80КБ	500КБ	100КБ
Кількість зірочок на github	122 тис.	43 тис.	71 тис.
Підтримка	Створено колишнім співробітником Google	Розроблено в Google	Підтримується Facebook
Написано на	JavaScript	TypeScript	JavaScript
Ідеально підходить для	Веброзробка та односторінкові програми	Масштабні, багатофункціональні програми	Сучасна веброзробка та нативні програми для iOS та Android

З огляду на те, що необхідно розроблювати SPA застосунок та є вимога, щоб об'єм даних, які будуть завантажуватися, був по можливості якомога меншим – було обрано Vue.js.

3.7 Обрання системи керування базою даних

Дані які зберігаються в БД мають бути однозначні, достовірні, несуперечливі. Для створення структури БД, яка відповідає вказаним вимогам, найкраще підходить реляційна БД.

Реляційна база даних – це набір формально описаних таблиць, з яких дані можуть бути доступні або повторно зібрані різними способами без необхідності реорганізації таблиць БД [21]. Для взаємодії з реляційною БД використовують structured query language (SQL).

SQL – стандартна мова для управління реляційними базами даних і маніпулювання даними. SQL використовується для додавання, оновлення, отримання та видалення даних [22].

Тому в якості сховища даних було обрано PostgreSQL, бо це реляційна система керування реляційними базами даних (СКБД), яка має гарну підтримку стандартів та більш ніж достатній набір функцій необхідних для керування БД у розроблюваному застосунку.

PostgreSQL, також відомий як Postgres, це безкоштовна та відкрита СКБД, що характеризується розширюваністю та відповідністю технічним стандартам. Дана СКБД призначена для роботи з різними навантаженнями, від окремих машин до сховищ даних або вебсервісів з багатьма одночасними користувачами. Це СКБД може працювати на сервері під управлінням операційної системи macOS, Linux, FreeBSD, OpenBSD та Windows [23].

PostgreSQL пропонує транзакції з властивостями атомарності, узгодженості, ізолюваності, довговічності (від англ. atomicity, consistency, isolation, durability, ACID), автоматично оновлюються представлення даних, матеріалізовані подання, тригери, зовнішні ключі та збережені процедури [24].

PostgreSQL має безліч функцій, спрямованих на те, щоб допомогти розробникам створювати додатки, адміністраторам захищати цілісність даних та створювати відмовостійкі середовища, а також допомагати керувати даними незалежно від того, наскільки великим чи малим є набір даних. На додаток до

того, що PostgreSQL є безкоштовним та з відкритим вихідним кодом, він дуже розширюваний. Наприклад, можна визначати власні типи даних, створювати власні функції, навіть писати код з різних мов програмування, не перекомпілюючи базу даних.

PostgreSQL намагається відповідати стандарту SQL, коли така відповідність не суперечить традиційним особливостям або не може призвести до неправильних архітектурних рішень. Багато функцій, які вимагає стандарт SQL, підтримуються, хоча іноді мають дещо різний синтаксис або функції. З часом можна очікувати подальших зрушень до більшої відповідності. На момент випуску версії 13 у вересні 2020 року PostgreSQL відповідає принаймні 170 із 179 обов'язкових функцій для SQL: 2016 Core conformance. На даний момент це найвищий показник серед реляційних база даних на ринку.

Нижче наведено неповний перелік різних функцій PostgreSQL:

а) типи даних:

- 1) примітиви: ціле, числове, рядок, булеве значення;
- 2) структуровані: дата/час, масив, діапазон, UUID;
- 3) документ: JSON/JSONB, XML, ключ-значення (Hstore);
- 4) геометричні: точка, лінія, коло, багатокутник;
- 5) користувацькі: комбіновані, спеціальні типи;

б) цілісність даних:

- 1) унікальний індекс, не пустий індекс;
- 2) первинні ключі;
- 3) іноземні ключі;
- 4) обмеження виключення;
- 5) явні замки, консультативні замки;

в) паралельність та продуктивність:

- 1) індексація: B-tree, Multicolumn, Expressions, Partial;
- 2) розширене індексування: GiST, SP-Gist, KNN Gist, GIN, BRIN, індекси покриття, фільтри Bloom;

- 3) складний планувальник/оптимізатор запитів, сканування лише за індексом, статистика по багатьом колонкам;
- 4) транзакції, вкладені транзакції (через точки збереження);
- 5) багатoversійний контроль паралельності (від англ. Multiversion Concurrency Control, MVCC);
- 6) паралелізація запитів читання та побудова індексів B-tree;
- 7) усі рівні ізоляції транзакцій, визначені стандартом SQL, включаючи серіалізацію;
- 8) JIT-компіляція виразів;
- д) надійність, відновлення після помилок:
 - 1) запис перед логуванням (від англ. Write-ahead Logging, WAL);
 - 2) реплікація: асинхронна, синхронна, логічна;
 - 3) точкове відновлення у часі (від англ. Point-in-time-recovery, PITR), активні режими очікування;
 - 4) табличні простори;
- е) безпека:
 - 1) аутентифікація: GSSAPI, SSPI, LDAP, SCRAM-SHA-256, сертифікати тощо;
 - 2) надійна система контролю доступу;
 - 3) захист на рівні стовпців і рядків;
 - 4) багатофакторна автентифікація за допомогою сертифікатів та додаткового методу;
- ж) розширюваність:
 - 1) збережені функції та процедури;
 - 2) підтримка процедурних мов: PL / PGSQL, Perl, Python і багато інших;
 - 3) обгортки для доступу до зовнішніх даних: підключаються до інших баз даних або потоків за допомогою стандартного інтерфейсу SQL;
 - 4) налаштований інтерфейс зберігання для таблиць;

5) багато розширень, що надають додаткові функціональні можливості, включаючи PostGIS;

з) інтернаціоналізація, пошук тексту:

1) підтримка міжнародних наборів символів;

2) повнотекстовий пошук.

Порівняльна характеристика найпоширеніших реляційних СКБД наведена у табл. 3.4.

Таблиця 3.4 – Порівняння реляційних СКБД

Критерії	СКБД		
	PostgreSQL	MySQL	MS SQL Server
Мова програмування реалізації	C	C++	C/C++
ACID	+	+	+
Повнотекстовий пошук	+	±	+
Підтримка каскадного видалення таблиць	+	-	-
Аналітичні функції	+	-	±

3.8 Вибір середовища розробки

В якості середовища для розробки було використано Pycharm, бо даний інструмент допомагає в розробці застосунків з використанням мови Python, JavaScript та багато інших.

PyCharm – інтегроване середовища розробки для мови програмування Python, розроблене компанією JetBrains.

Редактор фактично "отримує" код і глибоко розуміє його структуру, підтримуючи всі можливості мови для сучасних і застарілих проєктів. Він забезпечує автозавершення коду, рефакторинг, запобігання помилок та багато іншого.

Також, підтримується крос-технологічна розробка. Крім основної мови програмування, підтримуються:

- JavaScript;
- CoffeeScript;
- TypeScript;
- Cython;
- SQL;
- HTML / CSS;
- Node.js;
- Emmet.

Всі перелічені вище технології також мають підтримку рефакторинга, налагодження, юніт-тестування та іншого.

На додаток до підтримки часто використовуваних вебтехнологій, PyCharm також забезпечує першокласну підтримку вебплатформи Python, такої як Django. Можна використовувати IDE, щоб скористатися пропозиціями щодо доповнення коду для тегів Django, фільтрів, параметрів і змінних шаблону. Крім того, можна збирати додаткову інформацію про теги та фільтри, посиляючись на коротку документацію. IDE Python навіть допомагає веброзробникам налагоджувати шаблони Django, формувати код, перевіряти код та працювати з декількома консолями. У той же час PyCharm також підтримує широко використовувані вебфреймворки Python, такі як Pyramid та Web2Py. Він забезпечує заповнення коду та параметри навігації, характерні для Pyramid. Так само, це дозволяє веброзробникам скористатися можливостями заповнення коду та навігації під час роботи з Web2Py та багатьма іншими засобами написаними на Python [25].

PyCharm дозволяє підключатися до різних машин та віддалено створювати програмні додатки. Для цього можна скористатися вбудованою консоллю SSH, наданою IDE, для підключення до машин та віддаленого виконання різних завдань розробки за допомогою SSH. Можна навіть запускати, налагоджувати та профілювати програми Python у віддаленому середовищі, замінюючи локальний інтерпретатор на віддалений інтерпретатор. Крім того, PyCharm дозволяє програмістам створювати відтворювані середовища розробки

за допомогою надійного інструменту, такого як Vagrant, та спрощувати розробку розподілених програм через Docker.

Крім того, середовища розробки також мають вбудовані інструменти розробника. Вони допомагають виконувати багато рутинних завдань безпосередньо з середовища розробки, завдяки інтеграції з системами керування версіями, підтримці віддаленого розгортання, баз даних/SQL, інструментів командного рядка, Docker, Composer, REST Client та багатьох інших інструментів [26].

Порівняльна таблиця середовищ розробки наведена у табл. 3.5.

Таблиця 3.5 – Порівняння середовищ розробки

Критерії	Середовища розробки		
	Pycharm	VS Code	Komodo
Інтегроване середовище розробки	+	-	+
Безкоштовна версія для студентів	+	+	+
Кросплатформеність	+	+	+
Робота з БД	+	±	-
Підтримка Docker	+	±	-
Інструменти розробника	+	±	±

3.9 Вимоги до технічних засобів

Для нормальної розробки програми, повинні виконуватися наступні умови до апаратної частини:

- наявність IBM-сумісної персональної електронної обчислювальної машини (ПЕОМ), на якій буде працювати додаток;
- процесор із тактовою частотою 2,0 ГГц та більше;
- оперативна пам'ять 4 ГБ та більше;
- вільний простір на жорсткому диску не менше, ніж 7 ГБ.

3.10 Висновки за розділом 3

Під час роботи над розділом було спроектовано архітектуру майбутнього ПП. Були спроектовані модулі майбутньої програми. Було проаналізовано найбільш популярні мови програмування та після ретельного порівняння перевага була віддана мовам програмування Python для бекенд частини та JavaScript для фронтенд частини застосунку. В якості фреймворків було обрано FastAPI для Python та Vue.js для JavaScript. Також проаналізувавши найбільш популярні середовища розробки, було обрано середовище PyCharm.

4 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

Для забезпечення роботи інтерактивної платформи для інтелектуального аналізу даних було створено два модулі:

- бекенд, який відповідає за зберігання даних;
- фронтенд, який надає графічний інтерфейс у веббраузері та виконує Python скрипти.

Розглянемо розроблені модулі детальніше.

4.1 Структура бекенду

Загальна структура ПЗ бекенду (вебсерверу) має наступний вигляд. Вебсервер відповідає за обробку запитів, надає API для роботи зі збереженим програмним кодом, роботу з БД, зберігання розроблених програм, відповідає за авторизацію та контроль доступу до створення програм.

Обробка запитів відбувається наступним чином. Користувач зі свого веббраузера робить запит, який отримує вебсервер NGINX. Сервер, під час обробки запиту, надає SSL-сертифікат для безпечної передачі даних, приймає рішення, що необхідно віддати клієнту. Якщо клієнту необхідна статична інформація (картинки, JavaScript, CSS файли та інше), то вебсервер звертається до файлової системи за цими даними. Якщо клієнту необхідна динамічна інформація, то в цьому разі запит передається до Python вебзастосунку.

Схема обробки запитів наведена на рис. 4.1.

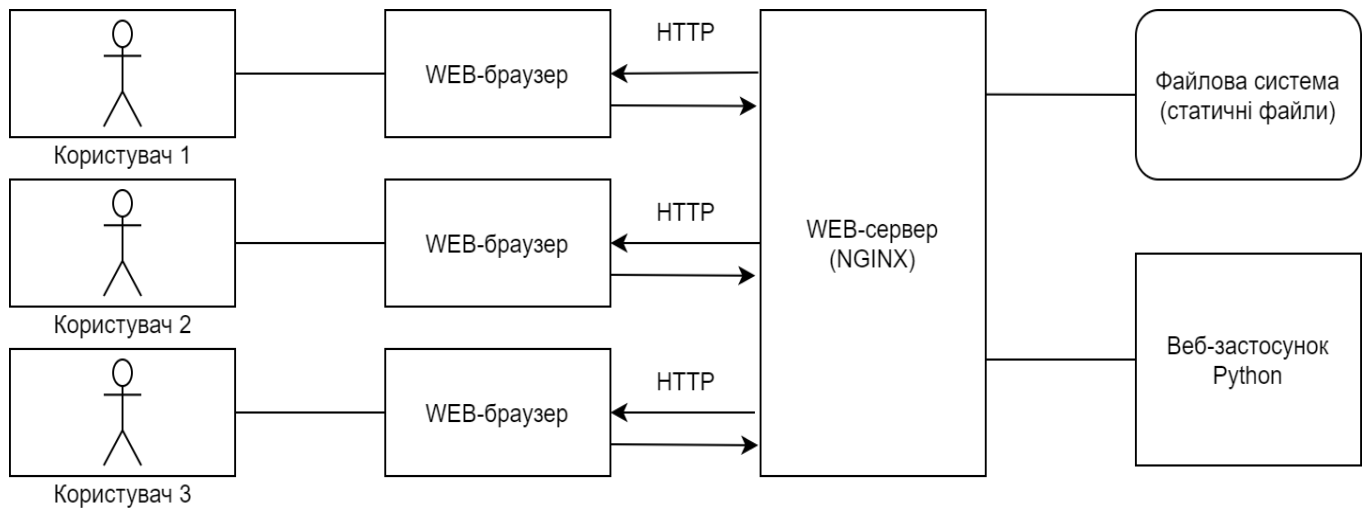


Рисунок 4.1 – Схема обробки запитів

В свою чергу Python вебзастосунок робить наступні операції:

- визначає, чи авторизований користувач з використанням стандарту JWT;
- визначає, яка вхідна точка повинна опрацювати запит;
- визначає, чи має користувач право доступу до вхідної точки;
- виконує всю бізнес логіку;
- працює зі сторонніми сервісами такими як БД.

Схема роботи Python вебзастосунка зображена на рис. 4.2.

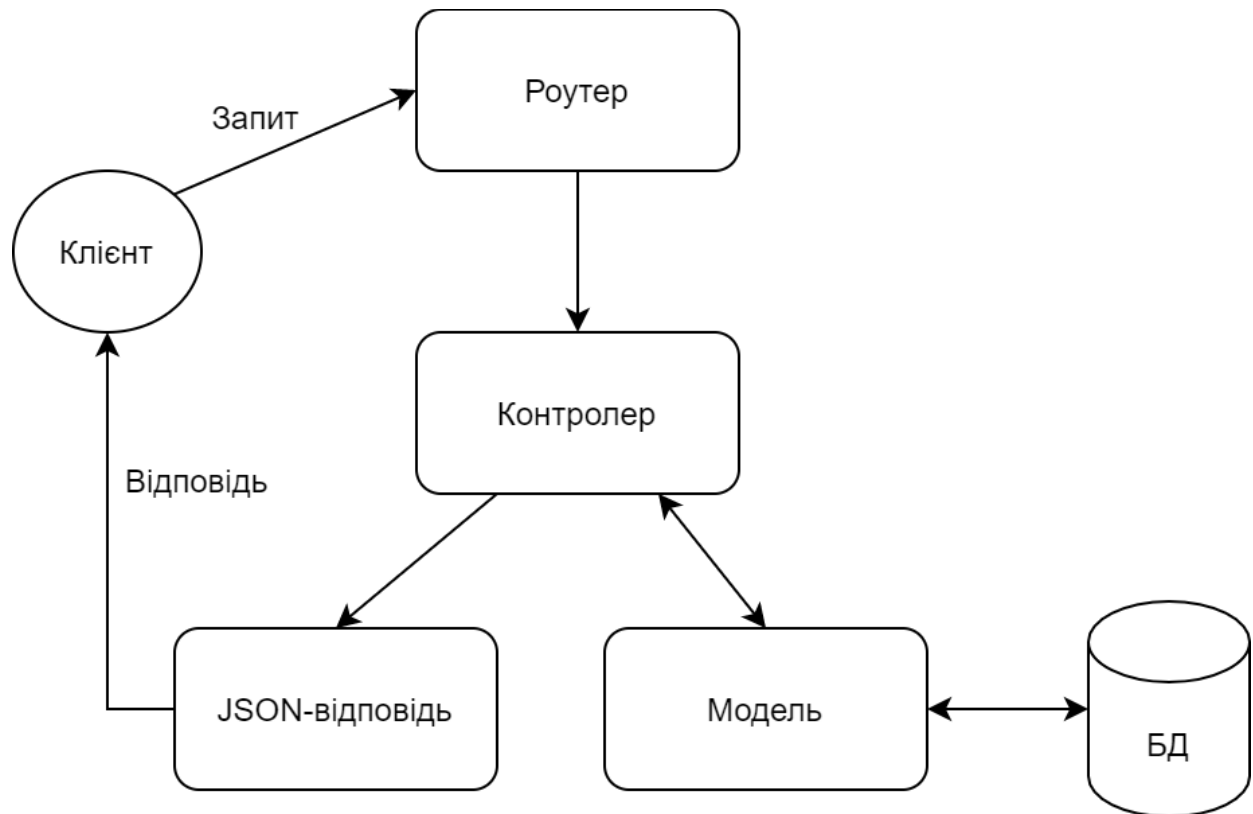


Рисунок 4.2 – Схема роботи Python вебзастосунка

Авторизація користувачів реалізована з використанням стандарту JWT.

4.1.1 Авторизація з використанням стандарту JWT

JSON web token (JWT) – це відкритий стандарт (RFC 7519), який визначає компактний та автономний спосіб безпечної передачі інформації між сторонами такої як об'єкт JSON. Цю інформацію можна перевірити та довіряти їй, оскільки вона має цифровий підпис. JWT можна підписати, використовуючи секрет (з алгоритмом HMAC) або пару публічних/приватних ключів, використовуючи RSA або ECDSA.

Підписані токени можуть перевірити цілісність вимог, що містяться в ньому, тоді як зашифровані токени приховують ці вимоги від інших сторін. Коли токени підписуються за допомогою пар публічного/приватного ключів, підпис також засвідчує, що лише сторона, що тримає приватний ключ, та, що підписала мають доступ до інформації.

У своєму компактному вигляді вебмаркери JSON складаються з трьох частин, розділених крапками, а саме:

- заголовок;
- вмісту;
- підпис.

Заголовок, як правило, складається з двох частин: типу токена, яким є JWT, та використовуваного алгоритму підписання.

Друга частина токена – це вміст токена, який містить твердження. Твердження – містить інформацію про сутність (як правило, про користувача) та додаткові дані. Існує три типи тверджень:

- зарезервовані – це набір заздалегідь визначених тверджень, які не є обов'язковими, але рекомендуються, щоб забезпечити набір корисних, взаємодіючих тверджень;
- публічні – можуть бути визначені за бажанням користувачів JWT. Але, щоб уникнути колізій, їх слід визначити у реєстрі вебтокенів IANA JSON або визначити як URI, що містить простір імен, стійкий до колізій;
- приватні – користувацькі твердження, створені для обміну інформацією між сторонами, які домовляються про їх використання, і не є ні зареєстрованими, ні публічними твердженнями.

Підпис використовується для перевірки того, що повідомлення не було змінено в після відправки, і, у випадку токенів, підписаних приватним ключем, він також може підтвердити, що відправник JWT є тим, ким він називається.

Для створення підпису потрібно взяти закодований заголовок, закодований вміст, секрет, алгоритм, зазначений у заголовку, і все це підписати [27].

Для використання JWT стандарту було використано Python бібліотеку `python-jwt`, яка реалізує даний стандарт та надає зручний та простий інтерфейс для роботи з ним.

4.2 Структура фронтенду

Фронтенд є ключовою частиною розроблюваного проєкту, бо він виконує ліву частину роботи з реалізації поставлених задач та містить основну частину бізнес логіки.

Фронтенд реалізує наступні функції:

- виконання Python скриптів;
- надання редактора для написання коду скриптів;
- відображення результатів виконання Python скриптів;
- надання зручного інтерфейсу;
- взаємодія з API надаваним бекендом.

Для реалізації виконання Python скриптів було використано пакет, розроблений компанією Mozilla, Pyodide. Редактор було зроблено за допомогою пакета CodeMirror. Для створення інтерфейсу та відображення результатів виконання Python скриптів було застосовано пакет Highcharts, Bootstrap та стандартні вебтехнології (HTML, CSS/SCSS, JS).

Для розробки фронтенду було використано фреймворк Vue.js, який деякою мірою наперед визначає майбутню архітектуру застосунку. Завдяки цьому розробка застосунку значно пришвидшується, бо тривіальні задачі типові для кожного проєкту заздалегідь вирішені.

Vue використовує Vuex сховище, яке є центральним елементом системи. «Сховище» – це контейнер, в якому зберігається стан застосунку. Дві особливості відрізняють сховище Vuex від простого глобального об'єкта:

- сховище Vuex реактивно. Коли компоненти Vue покладаються на його стан, то вони будуть реактивно і ефективно оновлюватися, якщо стан сховища змінюється;
- не можна безпосередньо змінювати стан сховища. Єдиний спосіб зробити зміни – явно викликати мутацію. Це гарантує, що будь-яка зміна стану залишає слід і дозволяє використовувати інструментарій розробника, щоб краще розуміти хід виконання програми.

З огляду на все вище написане архітектура фронтенд застосунку має вигляд зображений на рис. 4.3.

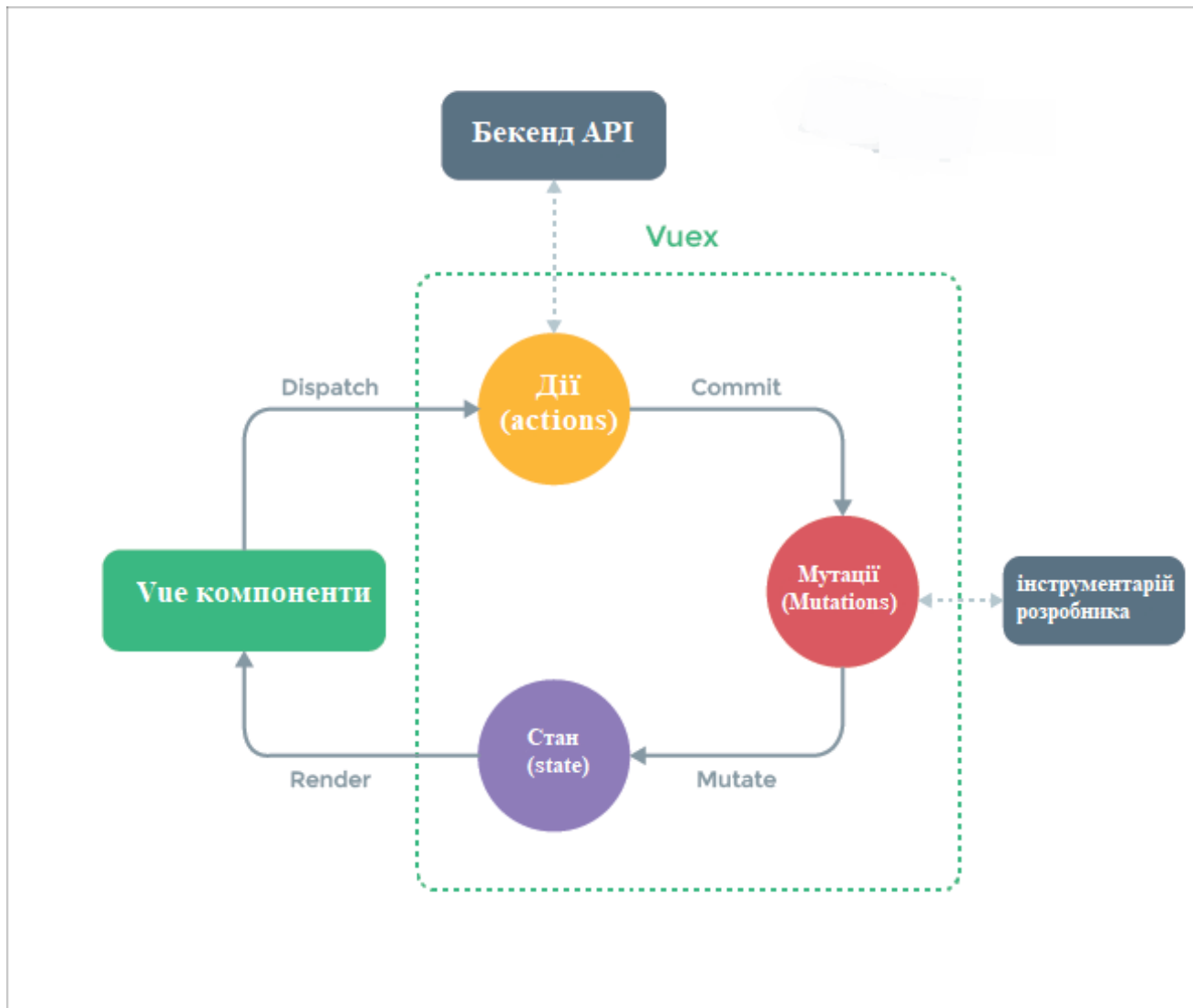


Рисунок 4.3 – Архітектура фронтенду.

Стан – це єдине дерево стану у Vuex – коли один об'єкт містить весь глобальне стан програми і служить «єдиним джерелом істини». Це також означає, що в застосунку буде тільки одне таке сховище. Єдине дерево стану дозволяє легко знайти потрібну його частину або робити знімки поточного стану програми з метою налагодження.

Мутації – єдиний спосіб зміни стану сховища у Vuex. Мутації у Vuex дуже схожі на події у JavaScript.

Дії – схожі на мутації з декількома відмінностями:

- замість того, щоб безпосередньо змінювати стан, дії ініціюють мутації;

- дії можуть використовуватися для асинхронних операцій.

Компоненти – це екземпляри Vue зі своїм ім'ям, які можна перевикористовувати. Компоненти зазвичай використовуються як HTML тег всередині кореневого примірника Vue.

Інтерфейс розроблюваного застосунку було поділено на багато малих компонентів, якими зручно керувати. Та з яких просто складати і формувати сторінки.

Список розроблених компонентів:

- nav-header – навігаційний блок, який дозволяє швидко переходити на різні сторінки (рис. 4.4);

- code-mirror-control-panel – блок з кнопками для запуску, збереження, видалення розроблених програм та налаштування редактора (рис. 4.5);

- code-mirror-editor – компонент з редактором (рис. 4.6);

- code-result-output – компонент, який відображає результат виконання програм (рис. 4.7);

- saved-programs-list – список збережених програм (рис. 4.8).

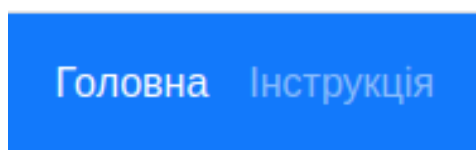


Рисунок 4.4 – Компонент nav-header.

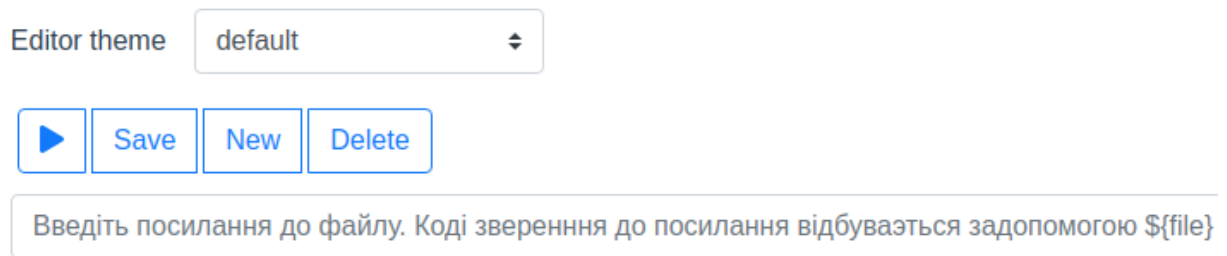
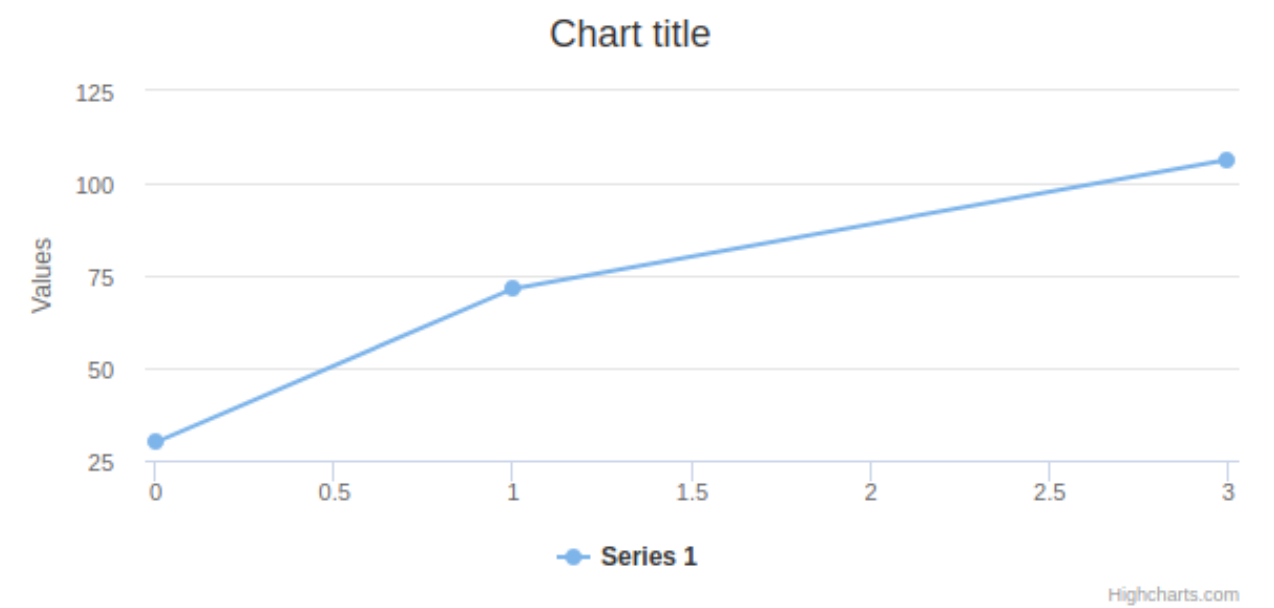


Рисунок 4.5 – Компонент code-mirror-control-panel

```
1 import pyodide
2 import pandas as pd
3
4 data_url = "http://www.sentiweb.fr/datasets/incidence-PAY-3.csv"
5
6 raw_data = pd.read_csv(pyodide.open_url(data_url), skiprows=1)
7
8 print(raw_data)
```

Рисунок 4.6 – Компонент code-mirror-editor

Data visualisation



Python console

```
undefined
Python initialization complete
```

Рисунок 4.7 – Компонент code-result-output

Список збережених програм

File download

File download from browser. Not good idea

Building chart

Рисунок 4.8 – Компонент saved-programs-list

4.2.1 Pyodide

На жаль, "мова, яку ми всі маємо" в браузері, JavaScript, не має зрілого набору бібліотек для роботи з даними, і в ній відсутня низка функцій, корисних для чисельних обчислень, таких як перевантаження оператора. Крім того, переважна більшість людей, які працюють з даними не використовують JavaScript у своїй роботі через перелічені вище причину та «особливу» роботу JavaScript з числами та округленнями.

Натомість Python позбавлений всіх вище перелічених недоліків але його не можливо просто взяти й запустити у браузері. Для вирішення цієї проблеми було використано пакет Pyodide.

Pyodide – це експериментальний проєкт Mozilla для створення повноцінного стека Python для роботи з даними, який повністю працює в браузері. Pyodide це інструмент призначений для експериментів з науковими даними, заснований на сучасних вебтехнологіях. Зокрема, він призначений для обчислення даних в браузері, а не на віддаленому сервері.

Pyodide дає повний, стандартний інтерпретатор Python, який працює повністю у веббраузері, з повним доступом до веб-API браузера.

Pyodide складається з трьох компонентів:

- вихідного коду основного інтерпретатора Python (CPython) та пакетів для наукових обчислень (NumPy тощо)
- застосування невеликих набору змін, щоб змусити завантажені компоненти працювати в умовах веббраузера;
- компіляції інтерпретатора Python та пакетів до WebAssembly за допомогою компілятора emscripten.

Якщо просто взяти цю вебзбірку і завантажили її в браузер, все буде зовсім іншим, порівняно з тим, що відбувається безпосередньо в операційній системі. Наприклад, веббраузер не мають файлової системи. Але emscripten забезпечує віртуальну файлову систему, написану на JavaScript, якою може користуватися інтерпретатор Python.

Емулюючи файлову систему та інші особливості стандартного обчислювального середовища, `emscripten` робить можливим переміщення існуючих проєктів у веббраузер із зовсім невеликими змінами. Тобто розробка в цьому середовищі мало чим відрізняється від звичної у редакторах на ПК.

Щоб завантажити `Pyodide` у браузер, потрібно завантажити:

- скомпільований інтерпретатор Python як `WebAssembly`;
- кілька JS файлів, які надає `emscripten`, що забезпечує емуляцію системи;
- упаковану файлову систему, що містить усі файли, які знадобляться інтерпретатору Python, особливо, стандартній бібліотеці Python.

Ці файли можуть бути досить великими: сам Python важить 21 Мб, `NumPy` – 7 Мб тощо. Але ці пакунки потрібно завантажити лише один раз, після чого вони зберігаються в кеші браузера.

Використовуючи всі ці фрагменти в тандемі, інтерпретатор Python може отримати доступ до файлів у своїй стандартній бібліотеці, запустити та зробити запуск коду користувача.

4.2.2 CodeMirror

Розробляти програми у звичайному текстовому полі зовсім не зручно. Тому для відтворення досвіду розробки в інтегрованому середовищі розробки було створена редактор коду за допомогою пакету `CodeMirror`.

`CodeMirror` – це редактор, який використовується в інструментах розробника в Firefox, Chrome і Safari, в Light Table, Adobe Brackets, Bitbucket та багатьох інших проєктах, що засвідчує його якість.

`CodeMirror` – компонент редактора коду, який можна вбудовувати у вебсторінки. Основна бібліотека містить лише компонент редактора, відсутні супровідні кнопки, автоматичне доповнення або інші функції IDE.

`CodeMirror` працює більше ніж з 100 мовами програмування, з окрема з Python. `CodeMirror` допомагає формувати код та підсвічує синтаксичні

конструкції, що значно полегшує розробку. Також має безліч тем, які допомагають обрати необхідну колірну гаму редактора для зручної роботи [28].

4.2.3 Bootstrap

Для організації та стандартизації коду, який відповідальний за побудову та відображення інтерфейсу (HTML, CSS), було використано вебфреймворк Bootstrap.

Фреймворк Bootstrap побудовано на HTML, CSS та JavaScript для полегшення адаптивних сайтів, мобільних сайтів та додатків.

Мета використання адаптивного вебдизайну – практичне відображення інформації та зручна навігація на всіх пристроях із доступом до інтернету. За технологією адаптивного вебдизайну не потрібно створювати окремі версії вебсайту. Один сайт може працювати на всьому спектрі пристроїв. Адаптивний дизайн дозволяє вебсторінці чи застосунку визначати розмір екрана відвідувача та автоматично адаптувати відображення.

Адаптивний дизайн складається з трьох основних принципів:

- підходи до побудови – розробка починається або з мобільного інтерфейсу, а далі відбувається адаптація інтерфейсу для інших розширень, або, навпаки, розробка починається від вигляду на екранах стаціонарних ПК і закінчується інтерфейсом на мобільних телефонах;
- використання відносних одиниць вимірювання – оскільки екрани різних пристроїв мають різну щільність пікселів, необхідно використовувати гнучкі одиниці вимірювання, що працюють всюди однаково;
- контрольні точки – дозволяють змінювати розташування блоків на сторінці тільки в разі використання екрану з певними розмірами, тобто розмітка вебсторінки змінюється в залежності від розміру екрану.

В ході виконання роботи розробка інтерфейсу починалася від вигляду на екранах стаціонарних ПК і закінчилася інтерфейсом для мобільних пристроїв,

відносних одиниці вимірювання та контрольні точки були використані, які надає фреймворк Bootstrap. Більшість екранів мають 2-3 контрольні точки.

4.2.4 Highcharts

Було використано JS бібліотеку для побудови графіків Highcharts, оскільки виникли деякі труднощі з відображенням графіків, які можуть будувати Python скрипти у розроблюваному рішенні. Великі графіки погано розміщуються на надаваному полі та не мають ніякої інтерактивності.

Бібліотека Highcharts – написана на чистому JavaScript. Вона пропонує простий спосіб додавання інтерактивних діаграм на вебсайт або вебдодаток. В даний час Highcharts підтримує більше десяти видів графіків.

Дана бібліотека працює у всіх сучасних браузерях для мобільних пристроїв та настільних ПК, включаючи iPhone/iPad та Internet Explorer версії 6. На iOS та Android підтримка мультитач забезпечує найкращий користувацький досвід. В сучасних браузерях використовується SVG для візуалізації графіки. У застарілому Internet Explorer графіка малюється за допомогою VML.

Highcharts базується виключно на нативних технологіях браузера і не потребує плагінів на стороні клієнта, таких як Flash або Java.

Приклад коду який необхідно вставити в Python скрипт для побудови графіку за допомогою Highcharts наведено на рис. 4.9.

```
listArr = []  
listArr.append({  
    "type": "line",  
    "name": "Predicted",  
    "data": prediction,  
})  
listArr.append({  
    "type": "scatter",  
    "name": "Data",  
    "data": scatter,  
})  
  
listArr
```

Рисунок 4.9 – Архітектура фронтенду

listArr – це список Python словників, де кожний словник має містити поля згідно з документації Highcharts.

4.3 Вхідні та вихідні дані бекенду

До вхідних даних вебсервера відноситься HTTP запити від користувачів до надаваного API.

Список та опис HTTP запитів, які може приймати API наведений у табл. 4.1.

Таблиця 4.1 – Опис HTTP запитів до API

Метод	URI	Опис
GET HEAD	api/programs	Повертає список збережених програм
POST	api/programs	Зберігає нову програму
PUT PATCH	api/programs/{program}	Оновлює існуючу програму
DELETE	api/programs/{program}	Видаляє збережану програму
POST	api/register	Реєструє користувача
POST	api/login	Авторизує користувача

Всі вихідні дані вебсервера – HTTP відповіді організовані у JSON форматі. Приклад відповідей, які повертають запити, наведені в додатку Б.

4.4 Вхідні та вихідні дані фронтенду

До вхідних даних фронтенду відносяться вихідні дані бекенду та текст програм, які створюють користувачі у редакторі.

До вихідних даних фронтенду відносяться вхідні дані бекенду та результати виконання Python скриптів.

4.5 Організація даних

Для зберігання даних було використано реляційну БД на основі СКБД PostgreSQL. Для створення схеми БД були застосовані міграції, які надаються об'єктно-реляційним інструментарієм SQLAlchemy у фреймворку FastApi. За допомогою цього інструментарію створюються моделі та взаємовідносини між ними, потім на основі розроблених моделей автоматично генеруються міграції, які вже безпосередньо відповідають за створення схеми БД. Завдяки цьому підходу можливо достатньо легко розроблювати схему БД та в разі помилок безпечно відновити стабільну версію.

Підчас проектування схеми БД за можливістю та необхідністю були дотримані принципи нормалізації БД. В результаті нормалізації БД була отримана третя нормальна форма, що є цілком достатньо для вирішення поставлених задач у розроблюваному ПЗ.

ER-діаграму розробленої БД зображено на рис. 4.10.

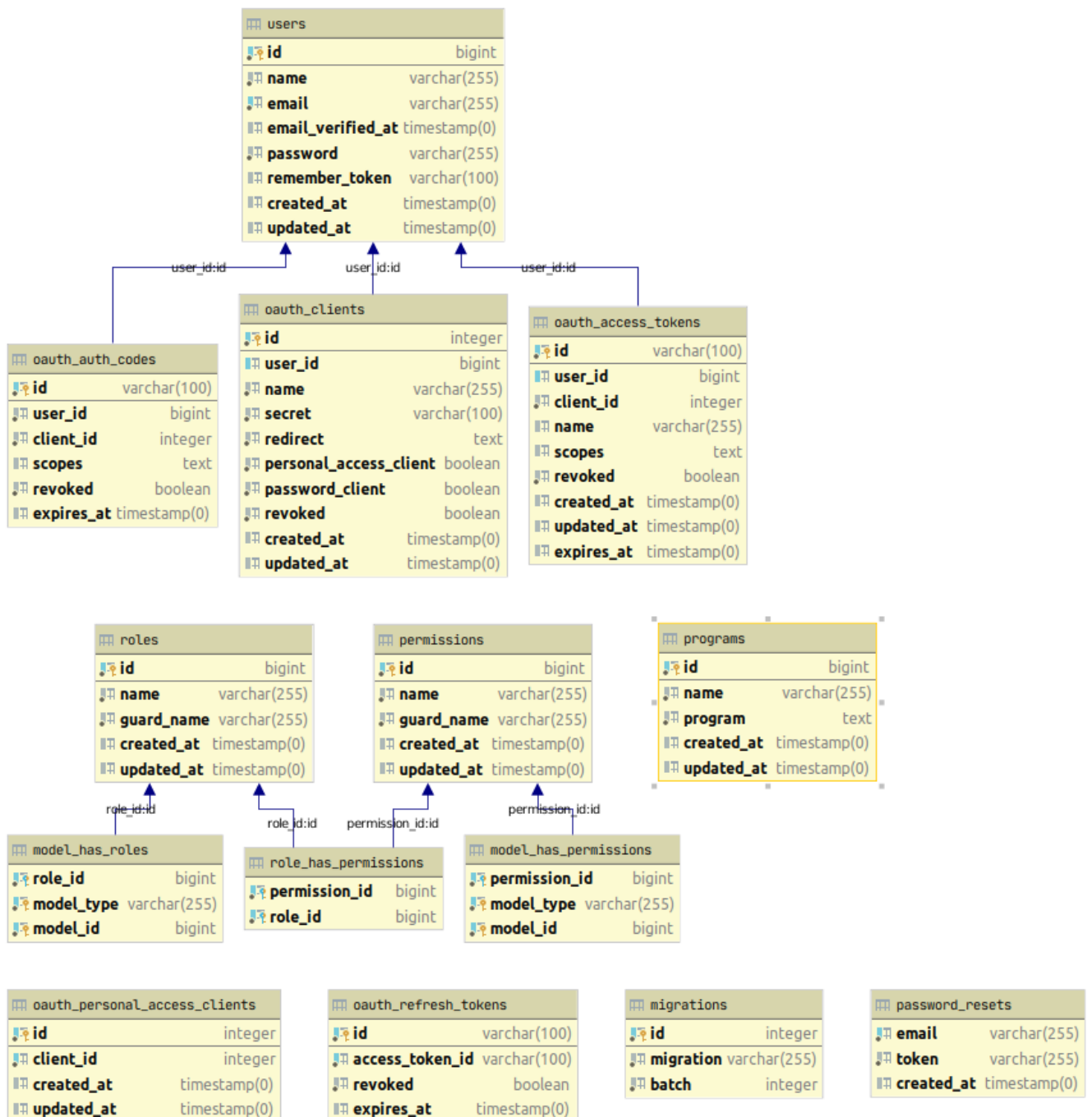


Рисунок 4.10 – ER-діаграма розробленої БД

4.6 Висновки за розділом 4

Розроблено інтерактивну платформу для інтелектуального аналізу даних, яка повністю відповідає поставленим вимогам. Описано рішення, які були прийняті під час розробки кожного з модулів системи. Створено REST API та вебінтерфейс для роботи з системою, яким зручно користуватися.

5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

Цей розділ присвячено опису розробленої інтерактивної платформи для інтелектуального аналізу даних було створено два модулі. Буде описано наведено інструкцію користування, а також буде проведено тестування та оцінка отриманих результатів.

5.1 Опис застосування

Розроблене ПЗ виконує Python скрипти, зберігає їх та надає можливість для їх розповсюдження між користувачами. Дане ПЗ може використовуватися для навчання студентів інтелектуальному аналізу даних в університетах та інших навчальних закладах, в приватних підприємствах з відділом по аналізу даних.

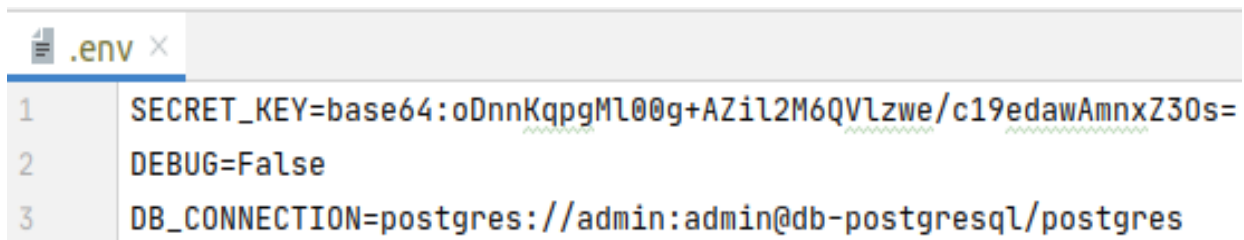
5.2 Інструкція по експлуатації програми

Для коректного запуску програми необхідно, щоб на машині був встановлена система керування версіями GIT, Docker Engine не нижче версії 19.03.13, або відповідна версія Docker Desktop для запуску контейнерів та Docker Compose для запуску мультиконтейнерного середовища.

Далі необхідно за допомогою GIT завантажити репозиторії проєкта та сконфігурувати його. Для цього необхідно знайти «.env.example» файл, який розташований в репозиторії проєкту (back-end), скопіювати його, замінивши ім'я на «.env». Приклад «.env» файлу зображено на рис. 5.1.

Після цього необхідно відкрити у термінал в репозиторії проєкту «back-end» та виконати в ньому команду: «docker-compose up -d». Після виконання цієї команди запускаються усі сервіси у режимі «detach» і ПЗ будо готове до використання.

Режимі «detach» означає, що контейнери будуть запущені у фоновому режимі і закриття терміналу не вплине на роботу застосунку.



```

1 SECRET_KEY=base64:oDnnKqpgMl00g+AZil2M6QVLzwe/c19edawAmnxZ30s=
2 DEBUG=False
3 DB_CONNECTION=postgres://admin:admin@db-postgresql/postgres

```

Рисунок 5.1 – Приклад .env файлу

Для перевірки працездатності ПЗ можна виконати будь-який запит до API, або виконати команду «docker container ls». Дана команда відобразить список запущених сервісів. Результат успішного виконання команди, наведеної вище, зображено на рис. 5.2.

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
e8e6a92c6057	someproject	"nginx -g 'daemon of...'"	8 seconds ago	Up 7 seconds	0.0.0.0:80->80/tcp	devops_nginx_1
bb5f31c1798f	someproject	"/usr/local/bin/wait..."	9 seconds ago	Up 7 seconds	9000/tcp	devops_app-python_1
904ba3ead2f9	someproject	"docker-entrypoint.s..."	9 seconds ago	Up 8 seconds	0.0.0.0:5432->5432/tcp	devops_db-postgresql_1

Рисунок 5.2 – Приклад успішно запущених сервісів

Для забезпечення роботи користувача з програмою необхідно мати обчислювальну систему з встановленим будь-яким сучасним веббраузером.

5.3 Дослідження розробленої програми

Для дослідження розробленого ПЗ було розроблено декілька Python скриптів та проведено серія їх запусків..

Дослідження проводилося на персональному комп'ютері із наступними характеристиками:

- процесор Intel Core i5-3470 з тактовою частотою 3.2 ГГц;
- 16 Гб ОЗУ;

– SSD накопичувач на 250 Гб.

– В ході дослідження розробленої програми було перевірені наступні функції:

- авторизація;
- побудова графіків за допомогою засобів Python;
- побудова графіків за допомогою засобів JavaScript;
- перевірка можливостей інтелектуального аналізу даних;
- перевірка можливостей не авторизованих користувачів.

Для перевірки авторизації було створено нового користувача, виконано вхід до системи та перевірено інтерфейс на наявність усіх необхідних можливостей для авторизованого користувача (рис. 5.3).

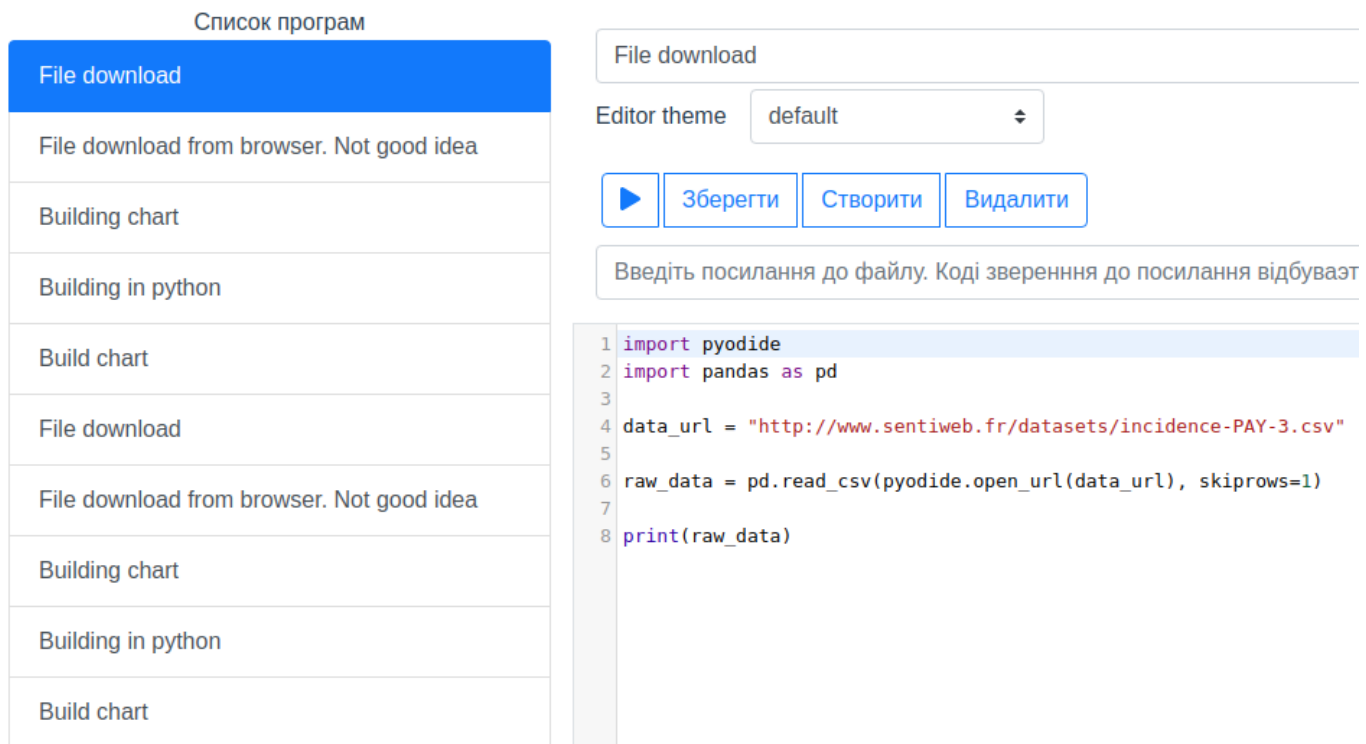
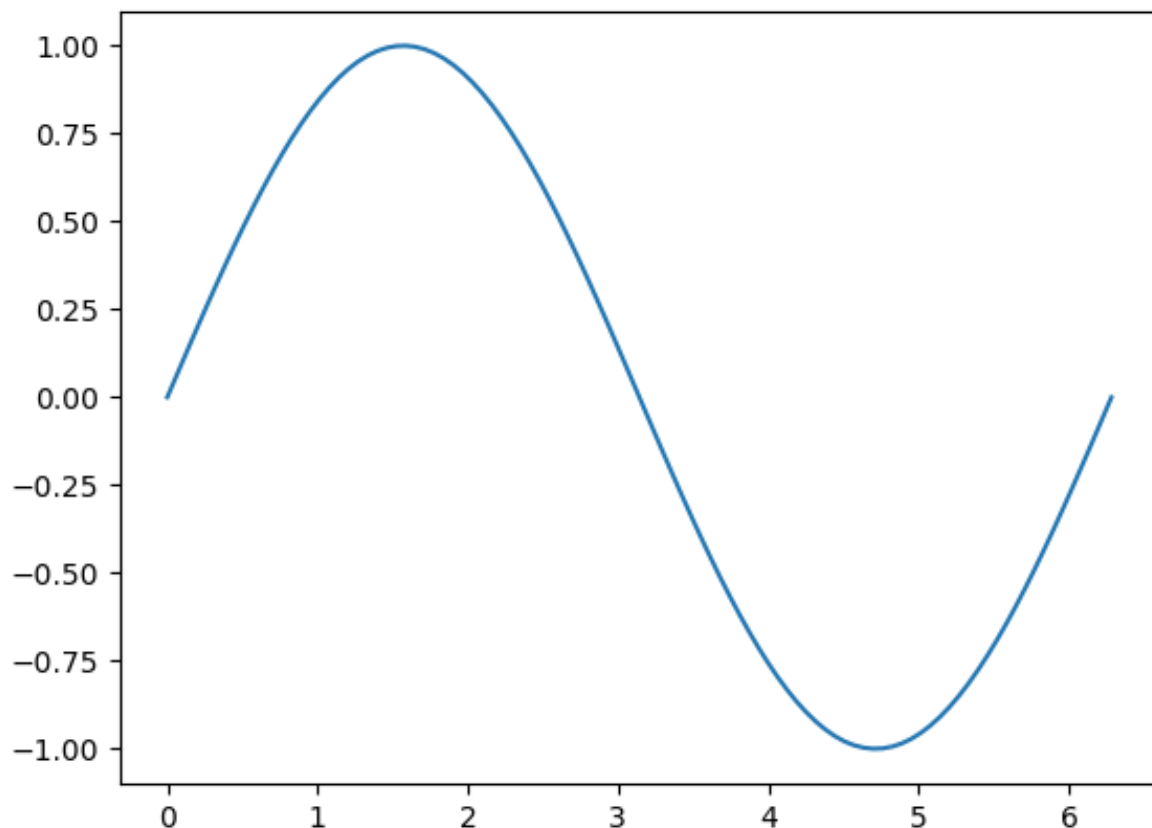


Рисунок 5.3 – Інтерфейс авторизованого користувача

Для перевірки побудови графіків за допомогою засобів Python було створено відповідний скрипт. Скрипт наведений в додатку В.1. Результат виконання цього скрипту наведений на рис. 5.4.

Візуалізація даних



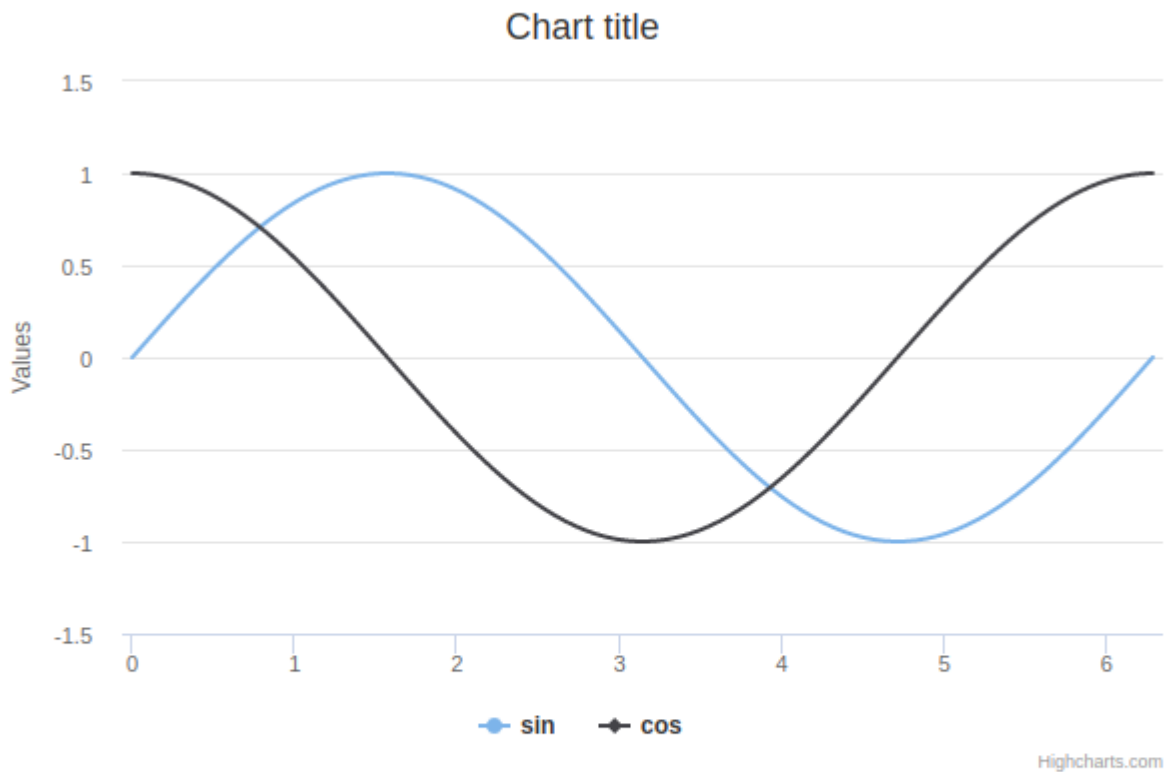
Консоль Python

```
matplotlib already loaded from default channel  
pandas already loaded from default channel  
numpy already loaded from default channel
```

Рисунок 5.4 – Побудова графіку за допомогою засобів Python

Для перевірки побудови графіків за допомогою засобів JavaScript було створено відповідний скрипт в якому будується графік синуса та косинуса. Скрипт наведений в додатку В.2. Результат виконання цього скрипту наведений на рис. 5.5.

Візуалізація даних



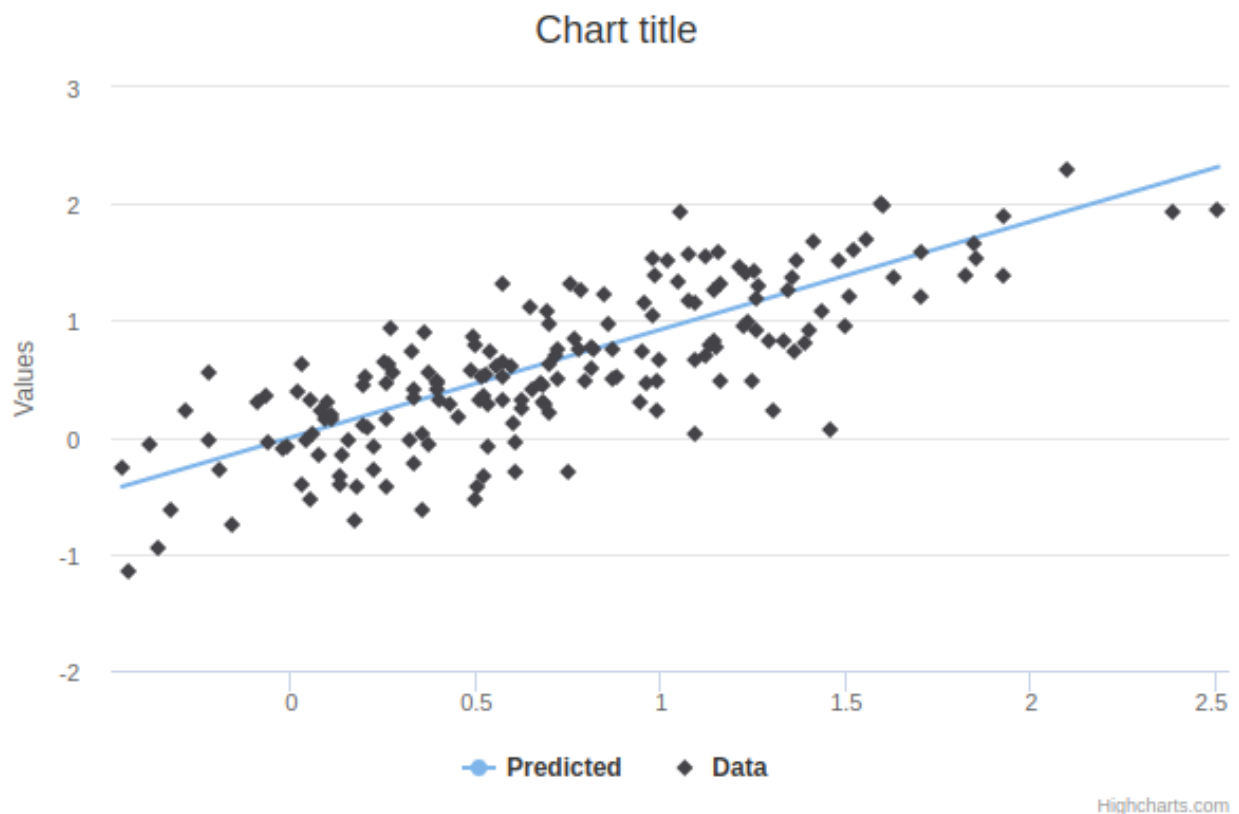
Консоль Python

```
matplotlib already loaded from default channel
pandas already loaded from default channel
numpy already loaded from default channel
```

Рисунок 5.5 – Побудова графіку за допомогою засобів JavaScript

Для перевірки можливостей ПЗ в якості інструмента для інтелектуального аналізу даних було створено скрипт, який будує лінійну регресію. Код скрипту наведений в додатку В.3. Результат виконання цього скрипту наведений на рис. 5.6.

Візуалізація даних



Консоль Python

```

kiwisolver to be loaded from default channel
cyclar to be loaded from default channel
pandas to be loaded from default channel
Loading matplotlib, pytz, python-dateutil, pyparsing, numpy, kiwisolver, cyclar, pandas
Loading matplotlib from https://pyodide-cdn2.iodide.io/v0.15.0/full/matplotlib.js
Loading pytz from https://pyodide-cdn2.iodide.io/v0.15.0/full/pytz.js
Loading python-dateutil from https://pyodide-cdn2.iodide.io/v0.15.0/full/python-dateutil.js
Loading pyparsing from https://pyodide-cdn2.iodide.io/v0.15.0/full/pyparsing.js
Loading numpy from https://pyodide-cdn2.iodide.io/v0.15.0/full/numpy.js
Loading kiwisolver from https://pyodide-cdn2.iodide.io/v0.15.0/full/kiwisolver.js
Loading cyclar from https://pyodide-cdn2.iodide.io/v0.15.0/full/cyclar.js
Loading pandas from https://pyodide-cdn2.iodide.io/v0.15.0/full/pandas.js
Loaded matplotlib, pytz, python-dateutil, pyparsing, numpy, kiwisolver, cyclar, pandas
https://raw.githubusercontent.com/Baakchsu/LinearRegression/master/weight-height.csv

```

Рисунок 5.6 – Результат побудови лінійної регресії

Для перевірки доступу до системи неавторизованого користувача, було відкрито застосунок у веббраузері в режимі інкогніто. В ході перевірки було затверджено, що не авторизований користувач може лише використовувати існуючі скрипти, модифікувати їх для власного використання, але не може створювати нові та видаляти існуючі (рис. 5.7). Що повністю задовольняє поставленим вимогам до ПЗ.

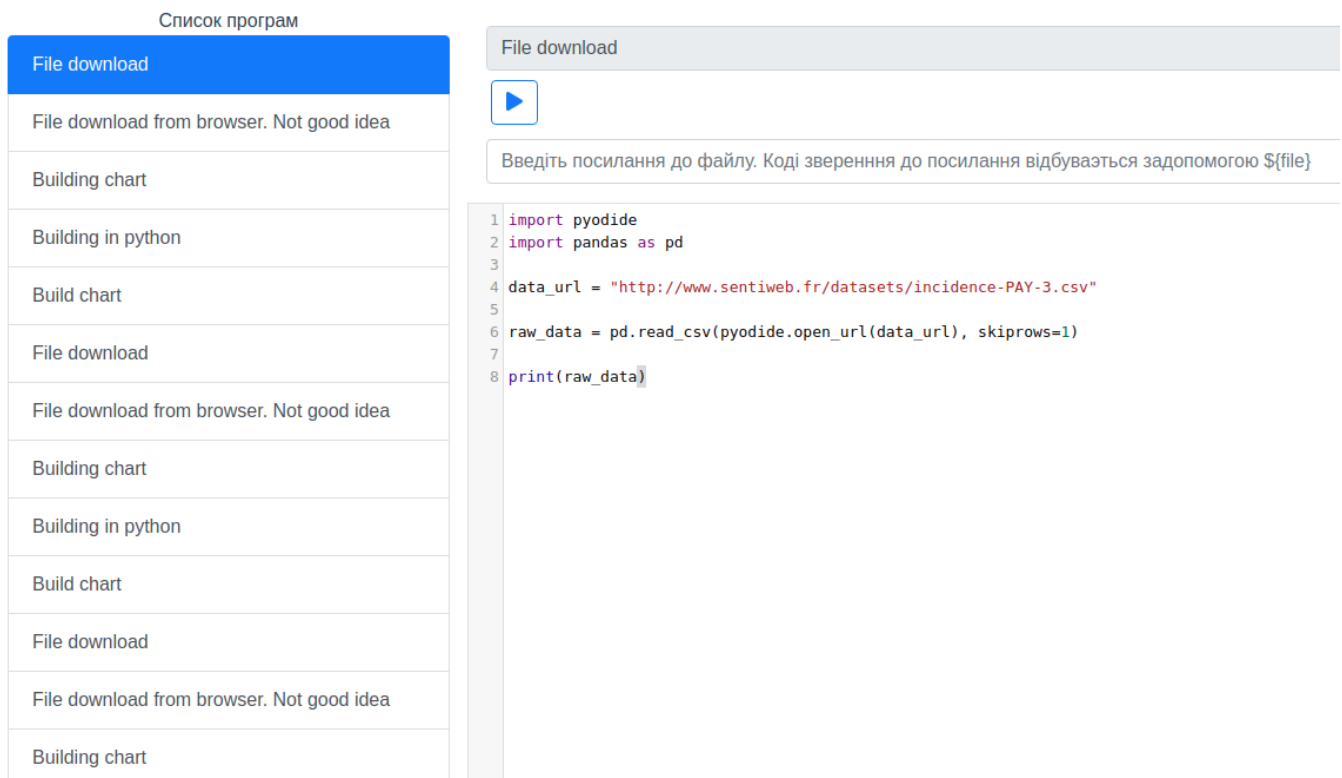


Рисунок 5.7 – Інтерфейс не авторизованого користувача.

5.4 Висновки за розділом 4

В розділі 4 було описано програму. Надано інструкцію з користування та проведено експериментальне дослідження і тестування ПЗ. В ході якого було розроблено Python-скрипти для побудови графіків різними способами та побудови лінійної регресії.

6 ОРГАНІЗАЦІЙНО-ЕКОНОМІЧНА ЧАСТИНА

Даний розділ присвячено для проведення економічного обґрунтування створення ПЗ, оцінка затрат на його створення, розрахунок економічної ефективності та терміну окупності [29].

6.1 Концепція економічного обґрунтування

Метою даної дипломної роботи є дослідження та використання засобів запуску користувацьких скриптів на мові програмування Python для створення інтерактивної платформи для інтелектуального аналізу даних. Основним завдання є створення зручної у використанні онлайн платформи для аналізу даних.

Оскільки аналітика даних зараз є невід’ємною частиною ділового життя, все більше компаній розробляють стратегії для ефективного управління нею та максимального використання. У сучасному стрімкому діловому світі для успішного управління бізнесом необхідні зручні засоби аналізу даних. Ми ніколи не отримували такої кількості інформації, як сьогодні, і не збирали стільки даних.

Переглядаючи історичні тенденції, відстежуючи поточну ефективність та роблячи прогнози на майбутнє, тепер компанії можуть використовувати даних для прийняття більш точних рішень у режимі реального часу, що допоможе впорядкувати ділові операції, покращити робочі процеси та збільшити прибуток.

Практична цінність роботи полягає в тому, що використання онлайн платформи для інтелектуального аналізу даних дає можливість для опрацювання даних та відтворення результатів у різних користувачів з найменшими зусиллями.

Застосування розроблюваного програмного продукту зменшує час необхідний для демонстрації ідей та розробок колегам та іншим зацікавленим сторонам. Що в свою чергу дозволяє швидше приймати якісні рішення у сучасному часі високої конкуренції практично у всіх галузях ринкової

економіки, і зменшити витрати на пошук цих рішень. Як результат підприємства збільшують свої прибутки та зменшують витрати.

Розроблювана платформа значною мірою покладається на обчислювані потужності користувачів, що дозволяє відмовитися від створення та підтримку потужного та коштовного дата-центру. Також такий підхід значно скорочує витрати на розробку, бо технічні задачі вирішуються простіше і кількість цих задач дещо менша.

Якщо розглядати конкурентів, то слід відзначити найбільш популярний конкурента на сьогоднішній день – проєкт Jupyter. Це сучасне комплексне рішення, але воно вимагає або безпосередньо встановлення цього рішення на ПК, або створення потужного дата-центру, який буде виконувати всі обчислення для кожного користувача. Таким чином при виході на ринок прямих конкурентів на сьогоднішній день не існує.

6.2 Планування розробки програмного продукту

Для розрахунку витрат необхідно визначити етапи робіт та їх тривалість. У розробці беруть участь програміст протягом 3 місяців та консультант протягом 0,15 місяця. Розробка ПЗ починається другого вересня та повинна бути закінчена до двадцять сьомого листопада 2020 року.

Тривалість робіт визначають за формулою (6.1):

$$T_{\text{ц}} = \frac{Q}{R} \quad (6.1)$$

де $T_{\text{ц}}$ – тривалість циклу, днів;

Q – трудомісткість, людино-днів;

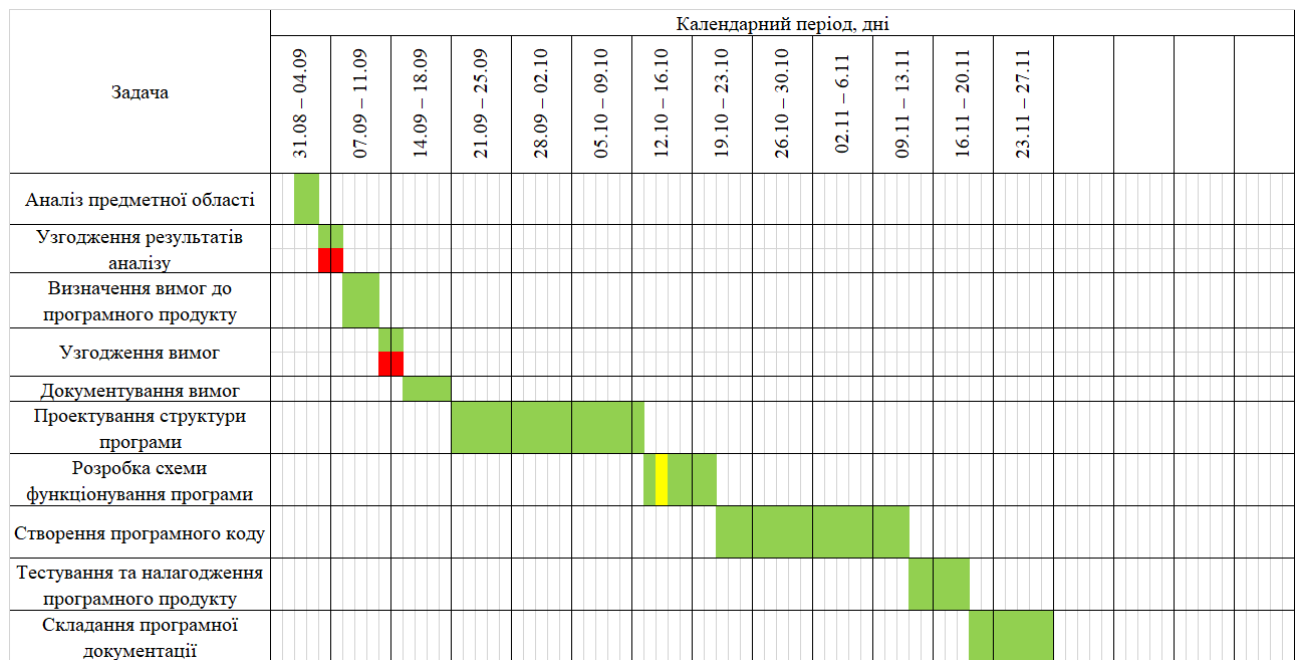
R – кількість виконавців, людей.

У табл. 6.1 наведена трудомісткість проектування системи.

Таблиця 6.1 – Характеристика робіт з розробки програми

Найменування роботи	Трудомісткість		Виконавці	Тривалість, дні
	люд.-дні	% до підсумку		
Аналіз предметної області	2	3,08%	Програміст	2
Узгодження результатів аналізу	4	6,15%	Програміст Консультант	2
Визначення вимог до програмного продукту	3	4,62%	Програміст	3
Узгодження вимог	2	3,08%	Програміст Консультант	2
Документування вимог	4	6,15%	Програміст	4
Проектування структури програми	16	24,62%	Програміст	16
Розробка схеми функціонування програми	6	9,23%	Програміст	6
Створення програмного коду	16	24,62%	Програміст	16
Тестування та налагодження програмного продукту	5	7,69%	Програміст	5
Складання програмної документації	7	10,77%	Програміст	7
Підсумок	65	100		63

За даними табл. 6.1 складається зведений стрічковий графік розробки програмного продукту, який представляє собою таблицю, в першому стовпці якої розміщені в порядку збільшення строків початку виконання всі види роботи, а навпаки – календарний період їх виконання. Даний графік наведений на рис. 6.1.



Легенда: зелений – робота програміста, червоний – робота консультанта, жовтий – святкові дні.

Рисунок 6.1 – Зведений стрічковий графік планування розробки

6.3 Розрахунок основної заробітної плати

Витрати за цією статтею складаються з планового фонду зарплати всіх категорій працівників, зайнятих у розробці програми. Розрахунок зарплати ведеться на підставі даних про трудомісткість, поданих у табл. 6.2.

Таблиця 6.2 – Розрахунок основної заробітної плати

Посада виконавця	Чисельність, чол.	Місячний оклад, грн	Кількість місяців роботи	Сума ЗП, грн
Програміст	1	10000	3	30000
Консультант	1	15000	0,15	2250
Підсумок	2			32225

6.4 Розрахунок додаткової заробітної плати

Додаткову заробітну плату приймають рівною 10% від основної заробітної плати працівників і розраховують за формулою (6.2):

$$ЗП_{\text{дод}} = ЗП_{\text{осн}} \cdot 0,1. \quad (6.2)$$

У нашому випадку отримаємо:

$$ЗП_{\text{дод}} = 32250 \cdot 0,1 = 3225 \text{ грн.}$$

6.5 Єдиний соціальний внесок

Єдиний соціальний внесок становить 22% і береться від основної та додаткової заробітної плати, формула (6.3).

$$ВТ_{\text{соц}} = (ЗП_{\text{осн}} + ЗП_{\text{дод}}) \cdot 0,22. \quad (6.3)$$

Далі отримаємо:

$$ВТ_{\text{соц}} = (32225 + 3225) \cdot 0,22 = 7779 \text{ грн.}$$

6.6 Визначення витрат на матеріали

Використовується 3 найменування матеріалів: диск DVD+RW – 15 грн; картридж – 100 грн, і папір – 100 грн (1 упаковка).

Витрати на матеріали вираховуються за наступною формулою (6.4):

$$ВТ_{\text{мат}} = \sum_{i=1}^n (Ц_i \cdot N_i \cdot (1 + K_{\text{т.з}}) - Ц_{io} \cdot N_{io}), \quad (6.4)$$

де $ВТ_{\text{мат}}$ – це витрати на матеріали, покупні напівфабрикати та комплектуючі вироби, грн.;

$K_{\text{т.з}}$ – коефіцієнт, що враховує транспортно-заготівельні витрати;

$Ц_i$ – ціна i -го найменування матеріалу, напівфабрикату і комплектуючого, грн;

N_i – потреба в i -му матеріалі, напівфабрикаті і комплектуючому;

Ц_{io} – ціна зворотних відходів i -го найменування матеріалу, грн;

N_{io} – кількість зворотних відходів i -го найменування;

n – кількість найменувань матеріалів, напівфабрикатів і комплектуючих.

$$\text{Ц}_{io} = 0; N_{io} = 0; K_{\text{т.з}} = 0,05.$$

$$\text{ВТ}_{\text{мат}} = (1 + 0,05) \cdot (15 + 100 + 100) = 225,75 \text{ грн.}$$

Разом, витрати на матеріали становлять 225,75 грн.

6.7 Витрати на спеціальне обладнання

Основною робочою машиною є персональний комп'ютер в наступній конфігурації: Intel Core i5-3470, Asus P8H77-M, 16 ГБ RAM, 1 ТБ HDD, Nvidia GeForce GTX 1070, Dell P2419H, Dell P2418D. Виходячи з даних, наданих виробником обладнання, а так само знятими системними показниками можна припустити, що середнє енергоспоживання персонального комп'ютера при подібній проектній роботі складає 500 Ватт. Початкова ціна робочої машини – 25000 грн.

Амортизаційні відрахування визначають за формулою (6.5) [30]:

$$A = \Phi_6 \cdot \frac{H_a}{100}, \quad (6.5)$$

де Φ_6 – балансова вартість обчислювальної техніки, грн;

H_a – норма амортизаційних відрахувань на повне відновлення обчислювальної техніки, %.

Норма амортизації для ПК становить 25%.

Балансова вартість обчислювальної техніки становить:

$$\Phi_6 = 7000 \text{ грн.}$$

Тоді отримаємо амортизаційні відрахування:

$$A = 7000 \cdot 0,25 = 1750 \text{ грн.}$$

Статтю «Експлуатація обладнання» розраховують підсумовуванням витрат на електроенергію та допоміжні матеріали, формула (6.6):

$$C_e = N_n \cdot \Phi_{\text{еф}} \cdot K_{\text{зч}} \cdot K_{\text{зп}} \cdot C_{\text{ел.е}}, \quad (6.6)$$

де N_n – номінальна потужність ЕОМ, кВт;

$\Phi_{\text{еф}}$ – річний ефективний фонд часу роботи ЕОМ, машино-год;

$K_{\text{зч}}$ – середній коефіцієнт завантаження по часу;

$K_{\text{зп}}$ – коефіцієнт завантаження по потужності;

$C_{\text{ел.е}}$ – ціна одного кВт·год електроенергії, грн/(кВт·год).

Номінальна потужність робочої машини – 0,2 кВт. Річний ефективний фонд часу роботи ЕОМ становить 1800 годин. Середні коефіцієнти завантаження за часом і по потужності рівні відповідно 0,9 і 0,6. Ціна однієї кіловат-години електроенергії становить 2,11 грн.

Отримуємо ціну експлуатації обладнання:

$$C_e = 0,5 \cdot 1800 \cdot 0,9 \cdot 0,6 \cdot 2,80 = 1360,80 \text{ грн.}$$

Зарплата обслуговуючого персоналу розраховується за формулою (6.7):

$$ЗП_{\text{обсл}} = \Phi ЗП_p \cdot (1 + K_{\text{відр}}) \cdot \frac{t_{\text{обсл}}}{\Phi_{\text{еф.обсл}}}, \quad (6.7)$$

де $\Phi ЗП_r$ – річний фонд заробітної плати (основної та додаткової) обслуговуючих робітників, грн;

$K_{\text{відр}}$ – коефіцієнт, що враховує відрахування на соціальне страхування й в інші фонди;

$t_{\text{обсл}}$ – час протягом року, необхідне на технічне обслуговування ЕОМ, год/рік;

$\Phi_{\text{еф.обсл}}$ – річний ефективний фонд часу обслуговуючого персоналу, год/рік.

Місячна заробітна плата обслуговуючого персоналу становить 5000 грн., а річний фонд заробітної плати відповідно дорівнює 60000 грн. Річний ефективний фонд робочого часу обслуговуючого ПК працівника дорівнює 1750 год/рік. На обслуговування одного ПК витрачається по 1 годині на місяць, що в рік становить 12 годин.

Отримуємо:

$$ЗП_{\text{обсл}} = 60000 \cdot (1 + 0,22) \cdot \frac{12}{1750} = 501,94 \text{ грн.}$$

Стаття «Поточний ремонт обладнання» приймається рівною 3% від балансової вартості обладнання і становить 210 грн.

Стаття «Інші витрати» приймається рівною 5% від суми всіх попередніх статей витрат на утримання і експлуатацію обладнання. Сума всіх попередніх статей дорівнює 3822,74 грн., 5% від суми становлять 191.14 грн.

Розраховані статті витрат на утримання та експлуатацію обладнання внесені до табл. 6.3.

Таблиця 6.3 – Кошторис витрат на утримання та експлуатацію обладнання

Найменування статей витрат	Сума, грн.
Амортизація обладнання	1750
Експлуатація обладнання (крім витрат на поточний ремонт)	1360,80
Заробітна плата основна та додаткова обслуговуючих робітників з відрахуваннями на соціальні заходи	501,94
Поточний ремонт обладнання	210
Інші витрати	191,14
Разом	4013,88

Витрати на оплату машинного часу ЕОМ для написання і налагодження даної програми визначаються за формулою (6.8):

$$C_{\text{мо}} = V_{\text{екс}} \cdot t_{\text{мо}}, \quad (6.8)$$

де $C_{\text{мо}}$ – витрати на оплату машинного часу, грн;

$V_{\text{екс}}$ – експлуатаційні витрати на одну годину машинного часу цієї цифрової ЕОМ, грн/машино-год;

$t_{\text{мо}}$ – машинний час цифровий ЕОМ для написання і налагодження даного програмного продукту, машино-год.

Експлуатаційні витрати на одну годину машинного часу використовуваної ЕОМ розраховують діленням суми витрат за кошторисом «Витрати на утримання та експлуатацію устаткування (ЕОМ)» (табл. 6.4) на річний ефективний фонд часу роботи ЕОМ. Річний ефективний фонд часу роботи ЕОМ дорівнює 1800 годин. В результаті експлуатаційні витрати на одну годину машинного часу рівні:

$$V_{\text{екс}} = \frac{4013,88}{1800} = 2,23 \text{ грн / машино-год.}$$

ЕОМ експлуатується 63 дні в одну зміну, що становить в сумі 504 годин. Таким чином, витрати на оплату машинного часу складуть [31]

$$C_{\text{МО}} = 2,23 \cdot 504 = 1123,92 \text{ грн.}$$

6.8 Інші прямі витрати

До інших прямих витрат включаються витрати на використання при розробці системи комерційне ПЗ:

- пайове ПЗ, що використовується постійно при роботі ПК (Ubuntu 20.04 LTS) – 0 грн. без ПДВ;
- цільове ПЗ, що для даної конкретної задачі (PyCharm) – 5600 грн. без ПДВ.

Розрахунки виконані з використанням формул (6.9) та (6.10).

$$S_{\text{пай.ПО}} = \frac{C_{\text{ПОUbuntu}} \cdot T_{\text{КТС}}}{\Phi_{\text{еф.КТС}} \cdot T_{\text{с.ПО}}}, \quad (6.9)$$

$$S_{\text{цел.ПО}} = C_{\text{ПОPyCharm}}, \quad (6.10)$$

де $S_{\text{пай.ПО}}$ – витрати на пайове ПЗ при розробці ПЗ з розрахунків, грн;

$S_{\text{цел.ПО}}$ – витрати на цільове ПЗ, що купується виключно для розробки ПЗ з розрахунків, грн;

$C_{\text{ПОUbuntu}}$ – ціна за Ubuntu (без ПДВ), грн;

$C_{\text{ПОPyCharm}}$ – ціна за PyCharm (без ПДВ), грн;

$T_{\text{КТС}}$ – машинний час КТС, необхідний користувачеві для розробки системи, машино-год/рік;

$\Phi_{\text{еф.КТС}}$ – річний ефективний фонд часу роботи КТС, машино-год/рік;

$T_{\text{с.ПО}}$ – термін служби пайового ПЗ, років.

$$S_{\text{пай.ПО}} = \frac{0 \cdot 504}{1800 \cdot 5} = 0$$

$$S_{\text{цел.ПО}} = 5620 + 0 = 5620 \text{ грн.}$$

$$S_{\Sigma} = 0 + 5620 = 5620 \text{ грн.}$$

6.9 Розрахунок накладних витрат

До накладних витрат відносяться витрати на загальне управління і загальногосподарські потреби (заробітна плата апарату управління, канцелярські витрати і т. д.), утримання та експлуатацію будівель.

Накладні витрати включаються у вартість розробки програми непрямым шляхом – у відсотках до основної заробітної плати розробників. В даному випадку накладні витрати складають 40% до основної заробітної плати розробників, що становить 12890 грн.

Результати визначення витрат на розробку програми у вигляді калькуляції кошторисної вартості робіт наведені в табл. 6.4.

Таблиця 6.4 – Калькуляція кошторисної вартості робіт з розробки програми

Найменування статей витрат	Сума, грн.	Питома вага до підсумку, %
Основна заробітна плата	32225	49,15
Додаткова заробітна плата	3225	4,91
Відрахування на соцстрах та ін.	7779	11,89
Матеріали та комплектуючі	225,75	0,50
Витрати на спец. обладнання	1123,92	1,00
Інші прямі витрати	5620	12,88
Накладні витрати	12890	19,66
Разом	63088,67	100
ПДВ	12617,73	
Разом (з ПДВ)	75706,40	

Таким чином, вартість проєкту оцінюється в 75706,40 грн.

6.10 Розрахунок економічної ефективності програмного продукту

Ефективність прикладних НДР визначається, як зіставлення річного економічного ефекту від застосування результатів розрахункових досліджень в умовах виробництва і використання капіталовкладень для здійснення втілення їх у виробництво, формула (6.11).

$$E_{\text{фак}} = \frac{\text{Економ. ефект.}}{\text{Капіталовклад.}} \quad (6.11)$$

Зіставляючи фактичну ефективність з нормативною ($E_n = 0,1 \dots 0,15$) робиться висновок про доцільність досліджень. Якщо $E_{\text{фак}} \geq E_n$, то дослідження визнаються економічно ефективними. У зворотному випадку дослідження не слід виконувати, тому що економічна вигода від таких досліджень мала. Економічну ефективність досліджень можливо оцінити за терміном окупності витрат на їх проведення та втілення, формула (6.12).

$$T_{\text{ок}}^{\text{фак}} = \frac{\text{Капіталовклад}}{\text{Економ. ефект.}} \quad (6.12)$$

Фактичний термін окупності капіталовкладень зіставляється з нормативним терміном окупності капіталовкладень, прийнятий для промисловості (приблизно 6 років), та на основі порівняння робиться висновок про доцільність досліджень.

Для теоретичних досліджень в більшості випадків важко або навіть неможливо розрахувати економічний коефіцієнт, тому доцільно визначити їх техніко-економічну ефективність з урахуванням наступних показників:

- важливість дослідження;
- складності розробки;

- результативності та можливості використання.

Важливість теоретичного дослідження оцінюють за його призначенням:

- вирішення проблемних питань;
- задоволення вимог спеціальної техніки;
- пошук принципово нових конструкцій і технологічних рішень та

т. п.

Складність виконання роботи визначають порівнянням отриманих результатів даного дослідження з результатами відомих аналогічних досліджень з обліком грошових і трудових витрат на їх проведення.

Результативність НДР можна визначити за повнотою рішень поставленого завдання: отриманий результат відповідає плановому, задовільний (часткове рішення) або негативний.

Аналіз залежності між цими показниками та витратами на їх досягнення дає можливість кількісної оцінки техніко-економічної ефективності теоретичних НДР по формулі (6.13):

$$K_{\text{НДР}} = \frac{J^n \cdot R \cdot T}{B_{\text{НДР}} \cdot t_{\text{НДР}}}, \quad (6.13)$$

де $K_{\text{НДР}}$ – рівень ефективності дослідження (коефіцієнт техніко-економічної ефективності НДР);

J – важливість роботи;

n – показник використання результатів НДР;

R – результативність роботи;

T – технічна складність виконання НДР;

$B_{\text{НДР}}$ – витрати на проведення НДР;

$t_{\text{НДР}}$ – час проведення НДР.

При $K_{\text{НДР}} \geq 1$ НДР вважається ефективною.

$$K_{\text{НДР}} = \frac{2,5^2 \cdot 1,1 \cdot 2,6}{75,7 \cdot 0,22} = 1,07$$

Після розрахунку отримуємо рівень ефективності дослідження вищий за 1, тому можна зробити висновок, що дана дослідницька робота є ефективною.

7 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА У НАДЗВИЧАЙНИХ СИТУАЦІЯХ

7.1 Аналіз потенційних небезпек

З сучасним розвитком науки та техніки всі сфери виробництва, змінюють предмети праці, технологію, якість, методи зберігання та обробки інформації. Разом з тим йде удосконалення безпечності та універсальності місця праці. Але на сьогоднішній день у багатьох сферах виробництва місце праці працівника залишається небезпечним та шкідливим для його здоров'я.

При роботі, в офісному та серверному приміщенні людину оточують: комп'ютери, системи зв'язку та інші засоби, що використовують досягнення радіоелектроніки, є генераторами цілої низки електромагнітних випромінювань. На сьогодні вплив на організм людини електромагнітного випромінювання ще не зовсім вивчений.

Методологічною основою охорони праці є науковий аналіз умов організації праці з точки зору її безпеки, функціонування системи взаємодії людини з середовищем, аналізу фізичних, хімічних, біологічних, психологічних та соціальних факторів безпеки виробничого процесу та особливостей впливу їх на організм людини [31].

Додержання правил техніки безпеки і виробничої санітарії залежить не тільки від виконання власником або уповноваженим ним органом своїх обов'язків, а й від того, наскільки кожний працівник знає і виконує їх під час роботи.

Програмний комплекс, що розробляється для проведення інтерактивного інтелектуального аналізу даних, передбачає роботу в обчислювальному центрі невеликої команди програмістів, що складається з 4 чоловік. Розміри приміщення $10 \cdot 5 \cdot 3.5\text{м}$.

Площа приміщення становить $10 \cdot 5 = 50\text{м}^2$, об'єм – $10 \cdot 5 \cdot 3.5 = 175\text{м}^3$.

Небезпечні та шкідливі виробничі фактори за природою виникнення поділяються на такі групи:

- фізичні (підвищена і знижена температура повітря, надмірна загазованість та запиленість повітря та ін.);
- біологічні (підвищений вміст у повітрі мікроорганізмів та ін.), хімічні (вміст у робочій зоні шкідливих речовин та ін.);
- психофізіологічні (нервово-емоційні перевантаження, розумове напруження та ін.).

Також потенційна небезпеки може бути пов'язана з порушеннями правил пожежної безпеки, наприклад: проблеми в організації заходів з пожежної безпеки, що можуть призвести до швидкого розповсюдження пожежі.

У даному розділі розглянуто наступні питання:

- мікроклімат та склад повітря робочої зони;
- виробниче освітлення;
- шум;
- виробничі випромінювання;
- безпека щодо організації робочих місць;
- безпечність технологічного обладнання та процесу;
- електробезпека;
- пожежна безпека.

7.2 Заходи із забезпечення безпеки

При роботі в обчислювальному центрі головним небезпечним фактором є електрика.

Приміщення обчислювального центру, за ступенем небезпеки ураження електричним струмом, відноситься до приміщень без підвищеної небезпеки

Питання правильного електроживлення при об'єднанні комп'ютерів в локальні мережі мають важливе значення як для забезпечення електробезпеки користувачів, так і для безаварійної і безперебійної роботи обладнання.

Заходи щодо забезпечення електробезпеки розроблено відповідно до стандартів [31].

Обладнання, що використовується в приміщенні обчислювального центру є споживачем електроенергії, що живиться від змінного струму 220 В, частотою струму 50 Гц від мережі з заземленою нейтраллю, та відноситься до електроустановок до 1000В закритого виконання.

Комп'ютери мають власні блоки живлення, які перетворюють змінний струм мережі у постійний (5 В, 12 В), комп'ютери використовують блоки живлення потужністю 500 – 750 Вт.

Заради зменшення ймовірності ураження людини електричним струмом живлення всіх електричних пристроїв здійснюється спеціальним кабелем, корпуси всіх електричних пристроїв виготовляються з неструмопровідних матеріалів.

Для забезпечення безпечності у приміщенні, всі кабелі сховані в спеціальні коробки, штепсельні розетки розташовані на висоті одного та 0,2 метри від підлоги, вимикачі на стінах знаходяться на висоті 1,2 метра від підлоги, електропроводка трифазна, чотирьох жильна, з подвійною ізоляцією, електрична мережа вмикається і вимикається за допомогою пускової апаратури (рубильником). Відповідно до міжнародного стандарту СЕ зазначені міри електробезпеки відповідають вимогам, пропонованим до побутових приладів.

Відповідно до стандартів [32] в приміщенні розроблено заходи щодо забезпечення електробезпеки.

Також для запобігання ураження людини електричним струмом в приміщенні використовуються:

- захисне заземлення;
- окремі розетки по 220 В для кожного робочого місця з обчислювальним пристроєм;
- триполюсні вилки та розетки, для підключення обчислювальних пристроїв до струму, у яких відповідний полюс з'єднаний з шиною заземлення;
- мережевий фільтр до якого підключений кожен блок живлення обчислювального пристрою або периферійного пристрою. Конденсатори цього

фільтра призначені для шунтування високочастотних перешкод живлячої мережі на землю через дріт захисного заземлення;

- окремий провід на загальний контор при з'єднанні пристроїв для ліквідування проблеми різниці потенціалів;
- ізолююча – лінолеум підлога.

Опір захисного заземлення в обчислювальному центрі становить 3 Ом (за стандартом – не більше 4 Ом) згідно з "Правилами улаштування електроустановок" (ПУЕ) також в приміщенні встановлено автоматичні вимикачі, пристрій заземлення контуру згідно стандартів [33].

Втома та синдром тунельний синдром також є розповсюдженим небезпечним фактором для здоров'я люди при роботі в обчислювальному центрі.

Для профілактики втоми робітників застосовуються специфічні методи, до яких можна віднести засоби відновлення функціонального стану зорового та опорно-рухового апарату, зменшення гіподинамії, підсилення мозкового кровообігу, оптимізацію розумової діяльності.

Для профілактики втоми та запобігання статистичного навантаження робітників встановлені такі правила організації робочого місця:

- оптимальна висота клавіатури від підлоги – 65-75 см;
- можливість регулювання висоти і нахилу клавіатури (відстань від поверхні стола до середини клавіатури – не більше 30 мм, кут підйому клавіатури – від 2 град до 15 град);
- наявність у клавіатури підставки для рук та килимка для миші з захистом від тунельного синдрому;
- наявність стільця або крісла з підлокітниками;
- рекомендовано використовувати перерви в роботі 5 хв. через кожні 40 хвилин та 15 хв. перерву кожні 2 години.

7.3 Заходи з забезпечення виробничої санітарії та гігієни праці

Заходи щодо забезпечення виробничої санітарії та гігієни праці для обчислювального центру обладнаного ПК з ВДТ розроблені відповідно до вимог державних санітарних норм та правил [34]-[36]. Розроблені заходи щодо забезпечення виробничої санітарії та гігієни праці для обчислювального центру обладнаного обчислювальними пристроями з ВДТ.

Так як роботи програмістів в обчислювальному центрі виконуються сидячи і не потребують фізичного напруження, то вони відносять до типу операторської роботи, даний тип роботи відносить до категорії 1а. Робоче місце програміста є постійним, робота проводиться у холодний та теплий періоди року тому рекомендується підтримувати оптимальні показники мікроклімату.

Для забезпечення оптимального рівня параметрів повітряного виробничого середовища використовуємо державний стандарт [37] та ГОСТ [38] (табл. 7.1).

Таблиця 7.1 – Норми мікроклімату в приміщенні

Період року	Категорія робіт	Температура повітря, °С	Відносна вологість повітря, %	Швидкість руху повітря, м/с
Холодний	Легка	22–24	40–60	0,1
Теплий	Легка	23–25	40–60	0,1-0,2

Оптимальні рівні позитивних (n+) і негативних (n-) іонів у повітрі приміщення з ПК нормовані згідно ГН 2152-80 «Санітарно-гігієнічні норми допустимих рівнів іонізації повітря виробничих та громадських приміщень» і становлять: $n+=1500-30000$ (шт. на 1см^3); $n- = 3000-5000$ (шт. на 1см^3). Підтримка оптимального рівня легких позитивних і негативних аероіонів у повітрі на робочих місцях забезпечено за допомогою біполярних коронних аероіонізаторів.

В обчислювальному центрі передбачено: устрій системи водяного опалення приміщення для забезпечення необхідної температури повітря в холодний період року відповідно до будівельних норм [39].

Проведені дослідження мікрокліматичних умов на комп'ютеризованих робочих місцях показали, що зимою значення відносної вологості повітря часто можуть нижчими за встановлені норми і становити в середньому 30–40%. Це приводить не лише до надмірного висихання слизових оболонок очей, носа, горла, а й до нагромадження зарядів статичної електрики, що утворюються в процесі роботи комп'ютера. Для усунення таких проблем в приміщенні передбачена наявність 4 зволожувачів повітря, які допомагають підняти вологість до рекомендованих 60% для усунення таких проблем. Кількість зволожувачів зумовлена тим, що один такий апарат розраховано на приміщення площею до 15 м².

У теплий період року для підтримки нормальної температури повітря в приміщенні використовуються три кондиціонери типу «спліт-система». Два кондиціонера розташовані у протилежних частинах та один по центру обчислювального центру дозволяють підтримувати температуру повітря не вище 24°C. Кількість кондиціонерів зумовлена тим, що один такий апарат розраховано на кондиціювання приміщення площею до 18 м².

Під впливом шуму знижується концентрація уваги, порушуються фізіологічні функції, з'являється стомленість у зв'язку з підвищеними енергетичними витратами і нервово-психічною напругою, погіршується мовна комутація. Рівні звуку, рівні звукового тиску в октавних смугах частот, та еквівалентні рівні звуку на робочих місцях у приміщення нормуються згідно державних санітарних правил [34]. Оптимальний рів шуму в приміщення забезпечується за допомогою:

- використання блоків живлення ПК з вентиляторами на гумових підвісках;
- використання більш сучасного обладнання;

- розташування принтерів та різноманітного устаткування колективного користування на значній відстані від більшості робочих місць працівників;
- відмова від жорстких дисків на користь твердотільних накопичувачів;
- використання пасивних систем охолодження ПК.

Вимоги до освітлення регламентовані державним будівельним нормативом (ДБН) [40], згідно з якими освітлення у приміщеннях з ЕОМ суміщене, при якому недостатнє за нормами природне освітлення доповнене штучним. Природне освітлення бічне, одностороннє.

Згідно з вимогами регламентованими в ДБН [40] в обчислювальному центрі коефіцієнт природної освітленості (КПО) має становити в середньому 1,1% (1,2% – у сонячний день та 1% – у хмарну погоду). Вікна в приміщенні обладнанні сонцезахисними металевими жалюзі з системою регульована для забезпечення відносної постійності природного освітлення. коефіцієнт відбиття жалюзі становить 0,7. Робочі місця робітників розташовано так, щоб у поле зору робітника не потрапляли світлі поверхні світильників та вікон.

Найбільш прийнятними для приміщень, де працюють ПК, є люмінесцентні лампи ЛБ (білого світла) і ЛТБ (тепло-білого світла) потужністю 20, 40 або 80 Вт. Для освітлення приміщення розміром $10 \cdot 5 \cdot 3,5\text{ м}$ освітленістю $E_p = 400\text{ лк}$, коефіцієнтом відбиття стелі 70% і стін 50% розрахуємо кількість ламп необхідних для освітлення використовуючи люмінесцентні лампи типу ЛБ в світильниках ЛПО.

Далі наведено формулу (7.1), для знаходження індекс приміщення:

$$i = \frac{A \cdot B}{h \cdot (A + B)} = \frac{10 \cdot 5}{3,5 \cdot (10 + 5)} = 0,9, \quad (7.1)$$

де A – в довжина приміщення;

B – ширина приміщення;

H – висота приміщення.

Приймаємо коефіцієнт запасу $k = 1,6$ і коефіцієнт нерівномірності освітлення за формулою (7.2):

$$z = \frac{B_{cp}}{B_{min}} = 1.1 \quad (7.2)$$

При індексі $i = 0,9$ отримаємо коефіцієнт використання світлового потоку $\eta = 35\%$.

В два ряди ($N_p = 2$) розміщуємо світильники.

Використовуючи вище зазначені параметри розраховуємо необхідний світловий потік ламп в кожному ряду за формулою (7.3):

$$\Phi_p = \frac{E_n \cdot S \cdot z \cdot k}{N_p \cdot \eta} = \frac{400 \cdot 50 \cdot 1,1 \cdot 1,6}{2 \cdot 0,35} = 50286 \text{ лм} \quad (7.3)$$

Далі розраховуємо кількість необхідних світильників за формулою (7.4), якщо в світильники встановити по дві лампи ЛБ ($n = 2$) потужністю 65 Вт і світловим потоком $\Phi_{\text{л}} = 4500 \text{ лм}$:

$$N_p = \frac{\Phi_p}{n \cdot \Phi_{\text{л}}} = \frac{50286}{2 \cdot 4500} = 6 \quad (7.4)$$

Для захисту від прямих сонячних променів, які створюють прямі та відбиті відблиски на поверхні екранів і клавіатури, використовуються сонцезахисні пристрої, на вікнах встановлені жалюзі або штори.

Організація та обладнання робочого місця в робочу приміщені має відповідати ергономічним вимогам з урахуванням особливостей трудовою дальності.

В приміщенні обчислювального центру одночасно працюють 4 чоловіка, кожний робітник має свій:

- обчислювальний пристрій, пристрою вводу (клавіатура, миш та інше), виводу (дисплей монітору, принтер, звукова навушники та інше) та набір периферійних пристроїв (принтер, сканер та інше);

- стіл;
- ортопедичний стілець;
- тумбу.

Також кожний робітник може скористатися шафою для одягу, шафою з приладами, двома тумбами призначеними для використання всіма робітниками.

Робоче місце працівника має площу 12,5 м², а об'єм – по 43,75 м³.

Приміщення має природне та штучне освітлення. Робочі місця розташовано так, щоб природне світло падало збоку. Робочі місця та взаємне розташування всіх його елементів відповідати антропометричним, фізичним і психологічним вимогам.

Також всі робочі місця організовані за такими правилами організації робочого місця:

- всім робітникам надані ергономічні та зручних миші і клавіатури;
- висота столу забезпечує розташування клавіатури від підлоги – 65-75 см, що позитивно впливає на працездатність робітника;
- кожна клавіатура має можливість міняти висоти і нахилу клавіатури;
- кожна клавіатура має підставку для рук;
- всі килимки для миші мають захист від тунельного синдрому (спеціальний виступ забезпечує правильне положення кисті);
- екрани монітору знаходяться на оптимальній відстані 700 мм від працівника;
- кожний стілець має підлокітник.

Загальний план обчислювального центру та робочих місць наведено на рисунку 7.1.

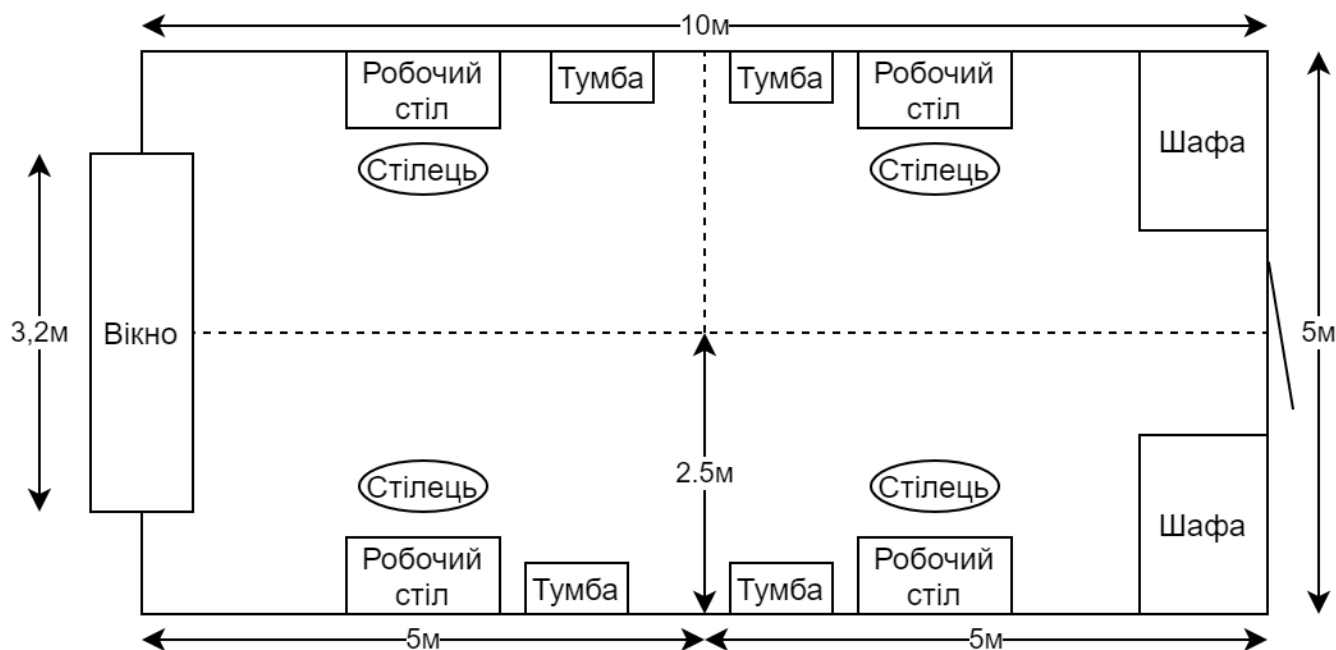


Рисунок 7.1 – Загальна схема обчислювального центру

7.4 Заходи безпеки у надзвичайних ситуаціях

7.4.1 Заходи з пожежної безпеки

Пожежі в обчислювальному центрі становлять особливу небезпеку, тому що пов'язані з великими матеріальними втратами. Так як в приміщенні обчислювального центру є всі три основні чинники, необхідні для виникнення пожежі, а саме: горючі речовини, окислення і джерела запалювання.

До ймовірних причин виникнення пожежі можна віднести несправність електрообладнання (кабелів, розеток), короткі замикання внаслідок виходу з ладу чи експлуатації несправного електроустаткування, порушення правил протипожежної безпеки тощо.

Для забезпечення пожежної безпеки проводилась пожежна профілактика, яка включає в себе комплекс організаційних і технічних заходів, спрямованих на забезпечення безпеки людей, на запобігання пожежі, обмеження її поширення, а також на створення умов для успішного гасіння пожежі.

Для відводу теплоти від ПК у обчислювальному центрі постійно діє потужна система кондиціонування. Тому кисень, як окислювач процесів горіння, є в будь-якій точці обчислювального центру.

Обчислювальний центр, згідно національного стандарту України [41] відноситься до категорії «П-Па», а клас можливої пожежі, згідно ДБН, визначається, як «А».

Для ліквідації пожежі у початковій стадії її розвитку силами персоналу об'єктів застосовуються первинні засоби пожежогасіння. До засобів гасіння пожежі, призначених для локалізації невеликих загорянь, відносяться пожежні стовбури, внутрішні пожежні водопроводи, вогнегасники, сухий пісок, азбестові ковдри і т. п.

Так як приміщення (дослідницької лабораторії, конструкторського бюро, тощо) що обладнане ПК з ВДТ має площу 50 м², тому відповідно до вимог [42] для гасіння електроустановок, що знаходяться під напругою, передбачені вуглекислотні вогнегасники типу ВВК-3,5 у кількості 3 штук (з розрахунку один вогнегасник с величиною заряду вогнегасної речовини 3 кг. і більше, на 20 м² площі приміщення). Відстань між вогнегасниками та місцями можливих загорянь не перевищує 10 м.

Усі працівники обчислювального центру повинні проходити протипожежний інструктаж під час прийняття на роботу і в процесі праці, а також проходити перевірку знань з питань пожежної безпеки. Евакуаційні виходи і проходи утримуються постійно вільними, легкодоступними та не захарашуються.

Відповідальний за протипожежний стан приміщень обчислювального центру повинен періодично (не рідше одного разу на місяць) перевіряти працездатність пристрою, який забезпечує автоматичне вимкнення системи вентиляції у разі пожежі, а також вогнедимозатримувальних пристроїв. Під час ремонту слід не допускати руйнування або порушення цілісності негорючих діафрагм у фальш-підлоговому просторі.

Фільтри припливно-витяжної вентиляції слід чистити згідно з затвердженим графіком, але не рідше двох разів на рік. Роботи з ремонту вузлів (блоків) ЕОМ, як правило, проводити не дозволяється. Їх необхідно проводити в окремому приміщенні (майстерні). У разі необхідності проведення ремонту або технічного обслуговування ЕОМ безпосередньо в залі допускається мати не більше 0,5 л легкозаймистих рідин у небиткій, щільно закритій тарі. Для промивання деталей необхідно застосовувати, як правило, негорючі миючі препарати. Чарунки та інші знімні пристрої допускається промивати горючими рідинами тільки в спеціальному приміщенні, обладнаному припливновитяжною вентиляцією. Відповідальний за пожежну безпеку приміщень обчислювального центру зобов'язаний негайно сповістити технічну службу у разі виявлення незадовільного технічного стану димових та теплових сповіщувачів автоматичної пожежної сигналізації, а також про несправності автоматичної установки об'ємного (газового) пожежогасіння.

Забороняється працівникам проводити ремонт вузлів ЕОМ. Ремонтні роботи необхідно проводити в окремому, спеціалізованому для ремонту приміщенні.

Для своєчасного попередження пожеж та підвищення оперативності реагування при їх виникненні у приміщенні використовується такий комплекс заходів та правил:

- обов'язковий інструктаж персоналу з питань охорони праці;
- зокрема, правила пожежної безпеки у приміщеннях з ЕОМ;
- наявність системи автоматичної пожежної сигналізації з димовими пожежними оповіщувачами;
- наявність шляхів евакуації при виникненні пожежі;
- заборона залишати без нагляду електричну апаратуру, що використовується;
- заборона використання відкритого вогню у приміщенні;
- заборона зберігати постійно в залах ЕОМ носії інформації, запасні блоки та деталі в кількості більшій, ніж необхідно для поточного використання;

- заборона користуватися груповими розетками на горючій панелі;
- заборона використовувати килими та доріжки із синтетичних матеріалів;
- заборона ставити на вікна глухі ґрати (в разі потреби встановлюються ґрати, що розкриваються зсередини);
- заборона застосовувати електронагрівальні побутові прилади;
- розміщення схеми евакуації людей при пожежі і ознайомлення з нею персоналу.

Усі працівники обчислювального центру повинні вміти привести в дію (в разі потреби) первинні засоби пожежогасіння. При виявленні ознак пожежі працівник повинен:

- негайно повідомити про це службу порятунку за номером телефону – 101 (112);
- вжити заходів щодо евакуації людей та матеріальних цінностей;
- вжити заходів до вжити заходів гасіння пожежі з використанням наявних вогнегасників та інших засобів пожежогасіння.

Керівник підрозділу при виникненні пожежі зобов'язаний:

- упевнитися що викликані рятувальні служби;
- відключити подачу електроенергії в приміщення;
- проінформувати про пожежу керівництво;
- перевірити чи людей оповіщено про пожежу;
- організувати евакуацію працівників (за умови загрози життю людей)
- забезпечити дотримання техніки безпеки працівниками, які беруть участь у гасінні пожежі;
- організувати зустріч підрозділів Державної пожежної охорони;
- надати допомогу підрозділам Державної пожежної охорони під час локалізації та ліквідації пожежі;
- забезпечити безперешкодний доступ до місця виникнення пожежі пожежно-рятувальним підрозділам.

7.4.2 Заходи з цивільного захисту. Єдина державна система цивільного захисту, її складові та режими функціонування

Єдина державна система цивільного захисту – сукупність органів управління, сил і засобів центральних та місцевих органів виконавчої влади, Ради міністрів Автономної Республіки Крим, виконавчих органів рад, підприємств, установ та організацій, які забезпечують реалізацію державної політики у сфері цивільного захисту.

Основними завданнями єдиної державної системи цивільного захисту є:

- забезпечення готовності міністерств та інших центральних та місцевих органів, виконавчої влади, органів місцевого самоврядування, підпорядкованих їм сил і засобів до дій, спрямованих на запобігання і реагування на надзвичайні ситуації;
- забезпечення реалізації заходів щодо запобігання виникненню надзвичайних ситуацій;
- навчання населення щодо поведінки та дій у разі виникнення надзвичайної ситуації;
- виконання державних цільових програм, спрямованих на запобігання надзвичайним ситуаціям, забезпечення сталого функціонування підприємств, установ та організацій, зменшення можливих матеріальних втрат;
- опрацювання інформації про надзвичайні ситуації, видання інформаційних матеріалів з питань захисту населення і територій від наслідків надзвичайних ситуацій;
- прогнозування і оцінка соціально-економічних наслідків надзвичайних ситуацій, визначення на основі прогнозу потреби в силах, засобах, матеріальних та фінансових ресурсах;
- створення, раціональне збереження і використання резерву матеріальних та фінансових ресурсів, необхідних для запобігання і реагування на надзвичайні ситуації;

- оповіщення населення про загрозу та виникнення надзвичайних ситуацій, своєчасне та достовірне інформування про фактичну обстановку і вжиті заходи;
- захист населення у разі виникнення надзвичайних ситуацій;
- проведення рятувальних та інших невідкладних робіт щодо ліквідації наслідків надзвичайних ситуацій, організація життєзабезпечення постраждалого населення;
- пом'якшення можливих наслідків надзвичайних ситуацій у разі їх виникнення;
- здійснення заходів щодо соціального захисту постраждалого населення;
- реалізація визначених законом прав у сфері захисту населення від наслідків надзвичайних ситуацій, в тому числі осіб (чи їх сімей), що брали безпосередню участь у ліквідації цих ситуацій;
- інші завдання, визначені законом.

Єдина державна система залежно від масштабів і особливостей надзвичайної ситуації, що прогнозується або виникла, функціонує у режимах:

- повсякденного функціонування;
- підвищеної готовності;
- надзвичайної ситуації;
- надзвичайного стану.

Режим повсякденного функціонування єдиної державної системи цивільного захисту встановлюється за умов нормальної виробничо-промислової, радіаційної, хімічної, сейсмічної, гідрогеологічної, гідрометеорологічної, техногенної та пожежної обстановки та за відсутності епідемій, епізоотій, епіфітотій.

Режим підвищеної готовності впроваджується у разі загрози виникнення надзвичайної ситуації за рішенням відповідно Кабінету Міністрів України, Ради міністрів Автономної Республіки Крим, обласних, Київської чи Севастопольської міських державних адміністрацій для єдиної державної

системи цивільного захисту у повному обсязі або частково для окремих її територіальних підсистем тимчасово встановлюється режим підвищеної готовності.

Режим надзвичайної ситуації встановлюється у разі виникнення надзвичайної ситуації за рішенням відповідно Кабінету Міністрів України, Ради міністрів Автономної Республіки Крим, обласних, Київської чи Севастопольської міських державних адміністрацій для єдиної державної системи цивільного захисту у повному обсязі або частково для окремих її територіальних підсистем тимчасово встановлюється режим надзвичайної ситуації.

Режим надзвичайного стану для єдиної державної системи цивільного захисту у повному обсязі або частково для окремих її територіальних підсистем тимчасово встановлюється у межах території, на якій введено правовий режим надзвичайного стану відповідно до Закону України "Про правовий режим надзвичайного стану".

ВИСНОВКИ

Під час виконання дипломної кваліфікаційної роботи магістра, було розроблено програмне забезпечення для інтерактивної платформи для інтелектуального аналізу даних.

В ході виконання роботи було вирішено наступні задачі:

- ознайомлення і аналіз предметної області;
- розгляд методів виконання користувацьких Python скриптів;
- розгляд мов програмування Python та JavaScript;
- розгляд середовища розробки PyCahrn;
- розроблення інтерактивної платформи для інтелектуального аналізу даних.

Розроблений програмний комплекс реалізує функції:

- авторизація користувачів;
- завантаження користувацьких вибірок даних;
- виконання Python скриптів на стороні клієнта;
- відображення результатів виконання користувацьких Python скриптів;
- збереження, оновлення та видалення розроблених Python скриптів.

Запропоноване рішення для аналізу даних дозволяє організаціям з обмеженими фінансами реалізувати швидке внутрішнє розповсюдження рішень. Також, розроблений програмний комплекс може використовуватися як відкрита площадка для реалізацій методів аналізу даних, бо не потребує значних потужностей від вебсервера. Для більш заощадливого використання ресурсів сервера виконання всього Python коду відбувається на стороні клієнтів.

Розроблений програмний комплекс має практичну спрямованість та готовий до використання.

Під час виконання роботи та дослідження розробленого програмного забезпечення з'явилася ідея, щодо покращення системи, а саме збільшити швидкість виконання Python скриптів за допомогою завантаження середовища

Python у кеш веббраузера підчас першого відкривання будь-якої сторінки ПП, що зменшить час очікування на виконання скрипту та покращить користувацький досвід.

Нові можливості зможуть розширити функціонал програмного комплексу та підвищити його продуктивність

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Computer Sketchpad [Електрон. ресурс]. – Режим доступу: https://web.archive.org/web/20160305171314/http://www.wgbh.org/article/?item_id=3360989
1. JSFiddle [Electronic resource]. – Access mode: <https://docs.jsfiddle.net/getting-started>
2. CodePen [Electronic resource]. – Access mode: <https://codepen.io/about/>
3. Jupyter - About Us [Electronic resource]. – Access mode: <https://jupyter.org/about>
4. Jupyter Notebook: An Introduction [Electronic resource]. – Access mode: <https://realpython.com/jupyter-notebook-introduction/>
5. How to Use Jupyter Notebook [Electronic resource]. – Access mode: <https://www.dataquest.io/blog/jupyter-notebook-tutorial/>
6. Apache Zeppelin [Electronic resource]. – Access mode: <http://zeppelin.apache.org/#multi-purpose-notebook>
7. Чем Apache Zeppelin лучше Jupyter Notebook для интерактивной аналитики Big Data [Электрон. ресурс]. – Режим доступа: <https://www.bigdataschool.ru/blog/apache-zeppelin-vs-jupyter-notebook.html>
8. Key concepts and getting started [Electronic resource]. – Access mode: https://iodide-project.github.io/docs/key_concepts/
9. Iodide: an experimental tool for scientific communication and exploration on the web [Electronic resource]. – Access mode: <https://hacks.mozilla.org/2019/03/iodide-an-experimental-tool-for-scientific-communicatioidide-for-scientific-communication-exploration-on-the-web/>
10. Executing Python code submitted via a web service [Electronic resource]. – Access mode: <https://www.software.ac.uk/blog/2017-11-23-executing-python-code-submitted-web-service>
11. How Brython works [Electronic resource]. – Access mode: <https://github.com/tmikov/jscomp>

12. WebAssembly Concepts [Electronic resource]. – Access mode: <https://developer.mozilla.org/ru/docs/WebAssembly/Concepts>
13. About Emscripten [Electronic resource]. – Access mode: https://emscripten.org/docs/introducing_emscripten/about_emscripten.html
14. Beginner's Guide to Python [Electronic resource]. – Access mode: <https://wiki.python.org/moin/BeginnersGuide>
15. What is Python? Executive Summary [Electronic resource]. – Access mode: <https://www.python.org/doc/essays/blurb/>
16. Garbage collection in Python: things you need to know [Electronic resource]. – Access mode: <https://rushter.com/blog/python-garbage-collector/>
17. FastAPI [Electronic resource]. – Access mode: <https://fastapi.tiangolo.com/>
18. About JavaScript [Electronic resource]. – Access mode: https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript
19. Односторінковий застосунок [Електрон. ресурс]. – Режим доступу: <https://dou.ua/forums/topic/25444/>
20. Relational database [Electronic resource]. – Access mode: <https://searchdatamanagement.techtarget.com/definition/relational-database>
21. Structured Query Language (SQL) [Electronic resource]. – Access mode: <https://www.techopedia.com/definition/1245/structured-query-language-sql>
22. What Is PostgreSQL? [Electronic resource]. – Access mode: <https://www.postgresql.org/docs/current/intro-what-is.html>
23. History [Electronic resource]. – Access mode: <https://web.archive.org/web/20170326020245/https://www.postgresql.org/about/history/#>
24. What is PyCharm IDE? [Electronic resource]. – Access mode: <https://medium.com/@mindfiresolutions.usa/what-is-pycharm-ide-cc0735784f64>
25. Pycharm - Introduction [Electronic resource]. – Access mode: https://www.tutorialspoint.com/pycharm/pycharm_introduction.htm

26. Introduction to JSON Web Tokens [Electronic resource]. – Access mode: <https://jwt.io/introduction/>

27. Description [Electronic resource]. – Access mode: <https://codemirror.net/#description>

28. Остапенко, В.В. Методичні вказівки по економічному обґрунтуванню дипломних проєктів для студентів спеціальностей 7.080403 «Програмне забезпечення автоматизованих систем», 7.080402 «Інформаційні технології проєктування» і 7.091501 «Комп'ютерні та інтелектуальні системи і мережі» всіх форм навчання [Текст] / В.В. Остапенко, Е.М. Касьян, Є.М. Сердюк. – Запоріжжя : ЗНТУ, 2015. – 22 с

29. Методичні вказівки по економічному обґрунтуванню дипломних проєктів для студентів спеціальності 8.090801 «Мікроелектроніка та напівпровідникові прилади» [Текст] / Уклали В.О. Варинська, Т.А. Антоненко, – Запоріжжя: ЗНТУ, 2002 – 26 с.

30. Коробко О.В. Методичні вказівки до виконання розділу «Охорона праці та безпека у надзвичайних ситуаціях» у дипломних проєктах (роботах) студентів–магістрів спеціальностей 121 Інженерія програмного забезпечення (спеціалізація: інженерія програмного забезпечення), 121 Інженерія програмного забезпечення (спеціалізація: програмне забезпечення систем), 122 Комп'ютерні науки (спеціалізація: інформаційні технології проєктування), 122 Комп'ютерні науки (спеціалізація: системи штучного інтелекту), 123 Комп'ютерна інженерія (спеціалізація: комп'ютерні системи та мережі), 123 Комп'ютерна інженерія (спеціалізація: спеціалізовані комп'ютерні системи), 124 Системний аналіз (спеціалізація: інтелектуальні технології та прийняття рішень в складних системах) усіх форм навчання [Текст] / Укл.: О.В. Коробко – Запоріжжя: ЗНТУ, 2018 – 27с.

31. Електробезпека в будівлях і спорудах. Вимоги до захисних заходів від ураження електричним струмом: ДСТУ Б В.2.5-82:2016. – [Чинний від 2017-04-01]. – К.: ДП «УкрНДНЦ», 2016. – 109 с. – (Державний Стандарт України)

32. Правила безпечної експлуатації електроустановок: НПАОП 40.1-1.01-97. – [Чинний від 1997-10-06]. – К.: Держнаглядохоронпраці, 1997. – 97 с. – (Нормативно-правовий акт охорони праці)

33. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин: ДСанПіН 3.3.2.007-98. – Чинний від 1998-12-10. – К. : МОЗ України, 1998. – 110 с. – (Державні санітарні правила та норми)

34. Гігієнічна класифікація праці за показниками шкідливості та небезпечності факторів виробничого середовища, важкості та напруженості трудового процесу : Державні санітарні норми та правила від 30.05.2014 р. № z0472-14. Редакція від: 30.05.2014

35. «Правила охорони праці під час експлуатації електронно-обчислювальних машин»: НПАОП 0.00-1.28-10 Наказ Держгірпромнагляду від 26.03.2010 р. № 65

36. Санітарні норми мікроклімату виробничих приміщень: ДСН 3.3.6.042-99. – [Чинний від 1999-12-01]. – К. : МОЗ України, 1999. – 106 с. – (Державні санітарні норми)

37. Загальні санітарно-гігієнічні вимоги до повітря робочої зони: ГОСТ 12.1.005-88. - [Чинний від 1989-01-01]. – Міністерство охорони здоров'я СРСР, 1988. – 101 с. – (Міжнародний стандарт)

38. Опалення, вентиляція та кондиціонування: ДБН В.2.5-67:2013. – [Чинний від 2014-01-01]. – К.: Мінрегіонбуд України, 2013. – 149 с. – (Державні будівельні норми України)

39. Природне і штучне освітлення: ДБН В.2.5-28-2018. – [Чинний з 2019-03-01]. – К. : Мінрегіон України, 2018. – 133 с. – (Державні будівельні норми України)

40. Визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою: ДСТУ Б В.1.1-36:2016. – [Чинний від 2017-01-01]. – К.: Мінрегіонбуд України, 2016. – 66 с. – (Національний стандарт України)

41. Пожежна техніка. Технічне обслуговування вогнегасників. Загальні технічні вимоги: ДСТУ 4297:2004. – [Чинний від 2004-10-01]. – УкрНДІПБ МНС України, 2004. – 74 с. – (Національний стандарт України)

Додаток А
Технічне завдання

A.1 Вступ

A.1.1 Назва програми

Назва програми: «Інтерактивна платформа для інтелектуального аналізу даних».

A.1.2 Призначення та область застосування

Програма призначена для аналізу даних за допомогою мови програмування Python з використанням вебтехнологій.

Програма може застосовуватися для аналізу даних в дослідницьких центрах або для вивчення аналізу даних.

A.2 Підстави для розробки

Підставою для розробки є завдання на дипломний проєкт на тему «Дослідження, розробка та програмна реалізація інтерактивної платформи для інтелектуального аналізу даних», затверджене наказом №301 в Національному університеті «Запорізька політехніка» від 3 листопада 2020.

A.3 Призначення розробки

Проєкт призначений для виконання аналізу даних з використанням вебтехнологій у веббраузері, що дозволяє миттєво приступати до роботи без необхідності налаштовувати робоче середовище.

A.4 Вимоги до програми

A.4.1 Вимоги до функціональних характеристик

Програма повинна забезпечувати виконання наступних функцій:

- авторизація користувачів;
- завантаження користувацьких вибірок даних;
- виконання Python скриптів на стороні клієнта;
- відображення результатів виконання користувацьких Python скриптів;
- збереження, оновлення та видалення розроблених Python скриптів.

A.4.2 Вимоги до надійності

Програма може базуватися на персональному, або на сервері. Під час роботи програми не повинно виникати збоїв, усі помилки, які можуть виникнути, повинні бути зрозуміло описані користувачу та не призводити до передчасного завершення роботи.

У разі виникнення збою в роботі апаратури, відновлення нормальної роботи вебресурсу повинно проводитися після: перезавантаження системи або перезапуску сервера, а також повторного виконання дій, втрачених до останнього збереження інформації.

До бази даних, яку використовує застосунок, висуваються наступні вимоги:

- надійне збереження даних;
- дані не повинні бути суперечливими;
- схема бази даних повинна мати третю нормальну форму.

A.4.3 Вимоги до складу та параметрів технічних засобів

Для нормального функціонування програми апаратна частина повинна відповідати наступним вимогам:

- наявність персонального комп'ютера з архітектурою процесора x86, або сервера, який підтримує роботу з Docker;
- процесор із тактовою частотою 2 ГГц та більше;
- оперативна пам'ять 4 Гб та більше;
- вільний простір на жорсткому диску не менше 7 Гб.

Також, для нормального функціонування програмна частина повинна відповідати наступним умовам:

- встановлений Docker версії 19.04;
- встановлена система контролю версій GIT.

A.4.5 Вимоги до транспортування та збереження

Транспортування програмного забезпечення повинно відбуватися через мережу Інтернет за допомогою системи контролю версій GIT.

Зберігатися розроблене програмне забезпечення повинно в безкоштовному GIT репозиторії.

A.5 Стадії та етапи розробки

A.5.1 Стадії розробки

Розробка програмного забезпечення повинна складатися з наступних стадій:

- аналіз предметної області;
- розробка технічного завдання;
- проектування архітектури системи;
- впровадження.

A.5.2 Етапи розробки

На стадії розробки технічного завдання повинен бути виконаний етап розробки, погодження та затвердження справжнього технічного завдання.

На стадії робочого проектування повинні бути виконані перераховані нижче етапи робіт:

- розробка системи;
- розробка програмної документації;
- випробування системи.

На стадії впровадження повинен бути виконаний етап розробки, підготовки та передачі системи.

A.5.3 Зміст робіт по етапах

На стадії аналізу предметної області повинно бути виконаний етап вивчення існуючих аналогів, виявлено недоліки існуючих систем та виділено нішу, яку займатиме розроблюване програмне забезпечення.

На стадії розробки технічного завдання повинен бути виконаний етап розробки та затвердження технічне завдання, на основі якого буде виконуватися розробка.

На стадії проектування повинен бути виконаний етап розробки та затвердження архітектури системи.

На стадії впровадження повинен бути виконаний етап розробки, тестування та передачі програми.

A.6 Порядок контролю та приймання

Розроблене програмне забезпечення повинно відповідати всім функціональним та технічним вимогам, пройти тести на значній кількості даних, пройти ручне тестування.

Додаток Б
Приклади відповідей вебсервера

```

1  {
2    "token_type": "Bearer",
3    "expires_in": 1296000,
4    "access_token": "eyJ0eXAiOiI0IiwianRpIjo
      eyJhdWQiOiI0IiwianRpIjo
      ZtMVLxwK3hBioJZMGb6YVYB
      5X-hfAiAUYhV0sZUV9e2EzT
      do_aXFDnqcQXT2yJDahqKK3
5    "refresh_token":
      "def5020045e76362948384
      3629685566ce8a45ac068db
      96f84bbc82cd1a3d36745aa
6  }

```

Рисунок Б.1 – Приклад відповіді на запит «POST api/register»

```

1  {
2    "token_type": "Bearer",
3    "expires_in": 1296000,
4    "access_token": "eyJ0eXAiOiI0IiwianRpIjo
      eyJhdWQiOiI0IiwianRpIjo
      SGW2xs6XAFbWAH2TXP1vqYQl
      E0rLbiM1JfWFG91EBV8BN3xKl
      aFLzSdKA0QHef7HJG3Jval1I
5    "refresh_token":
      "def50200c4a51b9e190bd99
      11537078ac60b306b1d5d7bc
      e4be4ac1d925586fa51a9802
6  }

```

Рисунок Б.2 – Приклад відповіді на запит «POST api/login»


```
1 {  
2   "id": 11,  
3   "name": "New program",  
4   "program": "sdfsdf ccc",  
5   "created_at": "2020-12-07 18:58:37",  
6   "updated_at": "2020-12-07 18:59:36"  
7 }
```

Рисунок Б.5 – Приклад відповіді на запит «PUT api/programs»

Додаток В
Розроблені Python-скрипти

```

Program Diplom;
// Програма для друку привітання
//
begin
    writeln('Hello'); // Виведення тексту на екран
end.

```

Д.1 Код скрипту для побудови графіку за допомогою засобів Python

```

import matplotlib.pyplot as plt
import numpy as np
import io, base64

x = np.linspace(0, 2.0 * np.pi, 100)
y = np.sin(x)

fig, ax = plt.subplots()
ax.plot(x, y)

buf = io.BytesIO()
fig.savefig(buf, format='png')
buf.seek(0)
img_str = base64.b64encode(buf.read()).decode('UTF-8')

```

Д.2 Код скрипту для побудови графіку за допомогою засобів JavaScript

```

import numpy as np
x = np.linspace(0, 2.0 * np.pi, 100)
y = np.sin(x)

x1 = np.linspace(0, 2.0 * np.pi, 100)
y1 = np.cos(x1)

tmp = np.vstack((x, y)).T

```

```
tmp1 = np.vstack((x1, y1)).T
```

```
listArr = []
listArr.append({
    "type": "line",
    "name": "sin",
    "data": tmp,
})
listArr.append({
    "type": "line",
    "name": "cos",
    "data": tmp1,
})
```

```
listArr
```

Д.3 Код скрипту для побудови лінійної регресії

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import pyodide

# data_url =
"https://raw.githubusercontent.com/Baakchsu/LinearRegression/master/weight-
height.csv"
data_url = "${file}"

print(data_url)

df = pd.read_csv(pyodide.open_url(data_url))

class LinearRegression:
    def fit(self,X,Y):
        X=np.array(X).reshape(-1,1)
        Y=np.array(Y).reshape(-1,1)

        x_shape = X.shape

        num_var = x_shape[1]
        weight_matrix = np.random.normal(0,1,(num_var,1))
        intercept = np.random.rand(1)
        for i in range(50):
```

```

        dcostdm
np.sum(np.multiply(((np.matmul(X,weight_matrix)+intercept)-Y),X))*2/x_shape[0]
        dcostdc      =      np.sum(((np.matmul(X,weight_matrix)+intercept)-
Y))*2/x_shape[0]
        weight_matrix -= 0.1*dcostdm
        intercept -= 0.1*dcostdc
    return weight_matrix,intercept
#print(df.drop(['Gender'],axis=1))
reg = LinearRegression()
x = (df['Weight']-df['Weight'].mean())/df['Weight'].std()
y = (df["Height"]-df["Height"].mean())/df["Height"].std()

params = reg.fit(x,y)
pred = np.matmul(np.array(x[:180]).reshape(-1,1),params[0])+params[1]

# print(pd.Series(pred.reshape(-1)))
# print(x[:180])

scatter = np.vstack((x[:180],y[:180])).T
prediction = np.vstack((x[:180], pd.Series(pred.reshape(-1)))).T

listArr = []
listArr.append({
    "type": "line",
    "name": "Predicted",
    "data": prediction,
})
listArr.append({
    "type": "scatter",
    "name": "Data",
    "data": scatter,
})

listArr

```

Додаток Д
Код програми

Д.1 Код файла front-end/src/main.js

```

import Vue from 'vue';
import BootstrapVue from 'bootstrap-vue';
import { library } from '@fortawesome/fontawesome-svg-core';
import { faPlay } from '@fortawesome/free-solid-svg-icons';
import { FontAwesomeIcon } from '@fortawesome/vue-fontawesome';
import './formValidation/vee-validate';
import vueScrollto from 'vue-scrollto';
import App from './App.vue';
import router from './router';
import store from './store';
import '@assets/styles/main.scss';
import RepositoryFactory from './apiAccess/repositoryFactory';

const AuthRepository = RepositoryFactory.get('auth');

library.add(faPlay);

Vue.component('font-awesome-icon', FontAwesomeIcon);

Vue.config.productionTip = false;
Vue.use(BootstrapVue);
Vue.use(vueScrollto);

new Vue({
  router,
  store,
  render: h => h(App),
  created() {
    const token = localStorage.getItem('user-token');
    this.$store.dispatch('loadSavedPrograms');

    if (token) {
      AuthRepository.setAuthToken(token);
    }
  },
}).$mount('#app');
```

Д.2 Код файла front-end/src/App.vue

```

<template>
  <div id="app">
    <nav-header class="mb-4"></nav-header>
    <router-view/>
  </div>
</template>

<script>
import NavHeader from './components/nav-header.vue';

export default {
  name: 'app',
  components: {
    NavHeader,
  },
};
</script>

<style lang="scss">
#app {
  font-family: 'Avenir', Helvetica, Arial, sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  text-align: center;
  color: #2c3e50;
}

#nav {
  padding: 30px;

  a {
    font-weight: bold;
    color: #2c3e50;

    &.router-link-exact-active {
      color: #42b983;
    }
  }
}
</style>

```

Д.3 Код файла front-end/src/apiAccess/repository.js

```
import axios from 'axios';

const baseDomain = 'http://someproject.local';
const baseUrl = `${baseDomain}/api`;

export default axios.create({
  baseUrl,
  headers: {
    Accept: 'application/json',
    // "Authorization":
  },
});
```

Д.4 Код файла front-end/src/apiAccess/repositoryFactory.js

```
import AuthRepository from './authRepository';
import ProgramRepository from './programsRepository';

const repositories = {
  auth: AuthRepository,
  program: ProgramRepository,
};

const RepositoryFactory = {
  get: name => repositories[name],
};

export default RepositoryFactory;
```

Д.5 Код файла front-end/src/apiAccess/programsRepository.js

```
import Repository from './repository';

// const resource = '/lala';

export default {
  get() {
    return Repository.get('programs');
```

```

},

save(newProgram) {
  const formData = new FormData();

  formData.set('name', newProgram.name);
  formData.set('program', newProgram.program);

  return Repository.post('/programs', formData);
},

update(updatedProgram) {
  const programId = updatedProgram.id;
  const formData = new FormData();

  formData.set('name', updatedProgram.name);
  formData.set('program', updatedProgram.program);
  formData.set('_method', 'PUT');

  return Repository.post(`/programs/${programId}`, formData);
},

delete(selectedProgram) {
  const programId = selectedProgram.id;

  return Repository.delete(`/programs/${programId}`);
},
};

```

Д.6 Код файла `front-end/src/apiAccess/authRepository.js`

```

import Repository from './repository';

export default {
  register(payload) {
    const formData = new FormData();

    formData.set('name', payload.name);
    formData.set('email', payload.email);
    formData.set('password', payload.password);
    formData.set('password_confirmation', payload.passwordConfirmation);
  }
};

```

```

    return Repository.post('/register', formData);
  },

  login(payload) {
    const formData = new FormData();

    formData.set('email', payload.email);
    formData.set('password', payload.password);

    return Repository.post('/login', formData);
  },

  setAuthToken(token) {
    Repository.defaults.headers.common.AUTHORIZATION = `Bearer
    ${token}`;
  },
};

```

Д.7 Код файлу `front-end/src/components/workspace/code-mirror-control-panel.vue`

```

<template>
  <b-container fluid class="text-left m-3">
    <loading :active.sync="isLoading" :can-cancel="false" :is-full-
page="true"/>

    <b-row class="mb-1">
      <b-form-input v-model="programName"
:readonly="this.$store.getters.user.token == null" placeholder="Enter your program
name"/>
    </b-row>

    <b-row v-if="this.$store.getters.user.token">
      <b-form-group label-for="editorTheme"
label="Editor theme" label-cols="4" >
        <b-form-select v-model="selectedCodemirrorTheme"
id="editorTheme"
:options="codemirrorThemes"/>
      </b-form-group>
    </b-row>

    <b-row>

```

```

        <b-button-group class="m-1">
            <b-button                variant="outline-primary"                class="mr-1"
@click="runProgram()">
                <font-awesome-icon icon="play" />
            </b-button>

            <b-button v-if="this.$store.getters.user.token" variant="outline-primary"
class="mr-1" @click="saveProgram()">
                Save
            </b-button>

            <b-button v-if="this.$store.getters.user.token" variant="outline-primary"
class="mr-1" @click="newProgram()">
                New
            </b-button>

            <b-button v-if="this.$store.getters.user.token" variant="outline-primary"
class="mr-1" @click="deleteProgram()">
                Delete
            </b-button>
        </b-button-group>
    </b-row>

    <b-row class="mt-2">
        <b-form-input v-model="dataFileUrl" placeholder="Введіть посилання
до файлу. Коді зверення до посилання відбувається за допомогою {{file}}"/>
    </b-row>

    <b-row>
        <b-alert
            :show="dismissCountDownSuccess"
            dismissible
            fade
            variant="success"
            @dismissed="dismissCountDownSuccess=0"
        >
            <p>{{ successAlertText }}</p>
        </b-alert>

        <b-alert
            :show="dismissCountDownWarning"
            dismissible
            variant="warning"
            @dismissed="dismissCountDownWarning=0"

```

```

    >
    <p>{{ warningAlertText }}</p>
  </b-alert>
</b-row>
</b-container>
</template>

<script>
  // define a new console
  let console = (function(oldCons){
    return {
      log: function(text){
        oldCons.log(text);

        let outputElement = document.getElementById('pythonCodeOutput')
        outputElement.value += text + '\n'

        // this.displayPythonMessages(oldCons, text);
        // Your code
      },
      info: function (text) {
        oldCons.info(text);
        // Your code
      },
      warn: function (text) {
        oldCons.warn(text);
        // Your code
      },
      error: function (text) {
        oldCons.error(text);
        // Your code
      }
    };
  })(window.console);

  //Then redefine the old console
  window.console = console;

  function displayPythonMessages(oldCons, text) {
    oldCons.log(text + ' !!')
    // this.$store.commit('setPythonCodeErrors', text);
  }

  import {test} from '../pyodide/pyodide';

```

```

// Import component
import Loading from 'vue-loading-overlay';
// Import stylesheet
import 'vue-loading-overlay/dist/vue-loading.css';
import RepositoryFactory from '../apiAccess/repositoryFactory';

const ProgramRepository = RepositoryFactory.get('program');
var VueScrollTo = require('vue-scrollto');

export default {
  name: 'code-mirror-control-panel',
  data() {
    return {
      isLoading: false,
      file: null,
      dataFileUrl: "",
      dismissSecs: 3,
      dismissCountDownSuccess: 0,
      dismissCountDownWarning: 0,
      successAlertText: "",
      warningAlertText: "",
      codemirrorThemes: [
        { value: 'default', text: 'default' },
        { value: '3024-day', text: '3024-day' },
        { value: '3024-night', text: '3024-night' },
        { value: 'abcdef', text: 'abcdef' },
        { value: 'ambiance', text: 'ambiance' },
        { value: 'base16-dark', text: 'base16-dark' },
        { value: 'base16-light', text: 'base16-light' },
        { value: 'bespin', text: 'bespin' },
        { value: 'blackboard', text: 'blackboard' },
        { value: 'cobalt', text: 'cobalt' },
        { value: 'colorforth', text: 'colorforth' },
        { value: 'darcula', text: 'darcula' },
        { value: 'dracula', text: 'dracula' },
        { value: 'duotone-dark', text: 'duotone-dark' },
        { value: 'duotone-light', text: 'duotone-light' },
        { value: 'eclipse', text: 'eclipse' },
        { value: 'elegant', text: 'elegant' },
        { value: 'erlang-dark', text: 'erlang-dark' },
        { value: 'gruvbox-dark', text: 'gruvbox-dark' },
        { value: 'hopscotch', text: 'hopscotch' },
        { value: 'icecoder', text: 'icecoder' },
        { value: 'idea', text: 'idea' },

```

```

    { value: 'isotope', text: 'isotope' },
    { value: 'lesser-dark', text: 'lesser-dark' },
    { value: 'liquibyte', text: 'liquibyte' },
    { value: 'lucario', text: 'lucario' },
    { value: 'material', text: 'material' },
    { value: 'material-darker', text: 'material-darker' },
    { value: 'material-palenight', text: 'material-palenight' },
    { value: 'material-ocean', text: 'material-ocean' },
    { value: 'mbo', text: 'mbo' },
    { value: 'mdn-like', text: 'mdn-like' },
    { value: 'midnight', text: 'midnight' },
    { value: 'monokai', text: 'monokai' },
    { value: 'moxer', text: 'moxer' },
    { value: 'neat', text: 'neat' },
    { value: 'neo', text: 'neo' },
    { value: 'night', text: 'night' },
    { value: 'nord', text: 'nord' },
    { value: 'oceanic-next', text: 'oceanic-next' },
    { value: 'panda-syntax', text: 'panda-syntax' },
    { value: 'paraiso-dark', text: 'paraiso-dark' },
    { value: 'paraiso-light', text: 'paraiso-light' },
    { value: 'pastel-on-dark', text: 'pastel-on-dark' },
    { value: 'railscasts', text: 'railscasts' },
    { value: 'rubyblue', text: 'rubyblue' },
    { value: 'seti', text: 'seti' },
    { value: 'shadowfox', text: 'shadowfox' },
    { value: 'solarized dark', text: 'solarized dark' },
    { value: 'solarized light', text: 'solarized light' },
    { value: 'the-matrix', text: 'the-matrix' },
    { value: 'tomorrow-night-bright', text: 'tomorrow-night-bright' },
    { value: 'tomorrow-night-eighties', text: 'tomorrow-night-eighties' },
    { value: 'ttcn', text: 'ttcn' },
    { value: 'twilight', text: 'twilight' },
    { value: 'vibrant-ink', text: 'vibrant-ink' },
    { value: 'xq-dark', text: 'xq-dark' },
    { value: 'xq-light', text: 'xq-light' },
    { value: 'yeti', text: 'yeti' },
    { value: 'yonce', text: 'yonce' },
    { value: 'zenburn', text: 'zenburn' },
  ],
  logs: "",
};
},
components: {

```

```

Loading
},
computed: {
  selectedCodemirrorTheme: {
    get() {
      return this.$store.state.codemirrorTheme;
    },
    set(selectedCodemirrorTheme) {
      this.$store.commit('setCodemirrorTheme', selectedCodemirrorTheme);
    },
  },
  programName: {
    get() {
      return this.selectedProgram.name;
    },
    set(newName) {
      const { selectedProgram } = this;

      if (selectedProgram.name !== newName) {
        selectedProgram.name = newName;
        selectedProgram.changed = true;
        this.$store.commit('setSelectedProgram', selectedProgram);
      }
    },
  },
  selectedProgram() {
    if (this.$store.state.selectedProgram == null) {
      return "";
    }
    return this.$store.state.selectedProgram;
  },
},
methods: {
  runProgram() {
    const { selectedProgram } = this.$store.state;
    let codeResultOutput = null;
    let pyPlotDiv = document.getElementById("pyPlotDiv");
    pyPlotDiv.classList.add('d-none');

    let outputElement = document.getElementById('pythonCodeOutput')
    outputElement.value = "";

    this.isLoading = true;

```

```

test().then(() => {
  pyodide.loadPackage(['numpy', 'pandas', 'matplotlib']).then(() => {
    codeResultOutput
  })
  pyodide.runPython(selectedProgram.program.replace('${file}', this.dataFileUrl));
  const imgStr = pyodide.globals.img_str;

  // console.log(pyodide.globals);

  if (imgStr) {
    this.displayImage(imgStr);
  } else {
    let pyPlotDiv = document.getElementById("pyPlotDiv");
    pyPlotDiv.classList.add("d-none");
  }

  if (codeResultOutput) {
    let jsChart = document.getElementById("jsChart");
    jsChart.classList.remove("d-none");
    this.buildChart(codeResultOutput);
  } else {
    let jsChart = document.getElementById("jsChart");
    jsChart.classList.add("d-none");
  }

  if (!imgStr && !codeResultOutput) {
    // TODO Something display to show that no valid data were returned
  }

  this.isLoading = false;
  cancelScroll = VueScrollTo.scrollTo('#dataVisualisation', 500)

}).catch(e => {
  // this.$store.commit('setPythonCodeErrors', e);
  this.isLoading = false;
});
}).catch(e => {
  this.isLoading = false;
});
},

displayImage(imgStr) {
  let pyPlotDiv = document.getElementById("pyPlotDiv");
  let pyPlotFigure = document.getElementById("pyPlotFigure");

```

```

    pyPlotDiv.classList.remove("d-none");
    pyPlotFigure.src = imgStr
  },

  buildChart(codeResultOutput) {
    if (codeResultOutput == null) {
      alert('You didn`t return any data. Chart won`t be drawn');
    } else {
      this.$store.commit('setCodeResultOutput', codeResultOutput);
    }
  },

  newProgram() {
    let programs = this.$store.state.savedPrograms;
    let maxId = Math.max(...programs.map(program => program.id));
    let newProgram = {
      id: maxId + 1,
      name: 'New program',
      program: "",
      changed: true,
      created: true,
    };

    this.$store.commit('setSelectedProgram', newProgram);
    this.$store.commit('addProgramToSavedPrograms', newProgram);
  },

  saveProgram() {
    let program = this.selectedProgram;
    let data = null;

    if (Object.prototype.hasOwnProperty.call(program, 'created')) {
      data = ProgramRepository.save(program);
    } else {
      data = ProgramRepository.update(program);
    }

    data.then((response) => {
      this.showSuccessAlert('Program saved.');
```

```

      let savedProgram = {
        id: response.data.id,
        name: response.data.name,
        program: response.data.program,
```

```

        changed: false
    };

    this.$store.commit('deleteSavedProgram', program);
    this.$store.commit('setSelectedProgram', savedProgram);
    this.$store.commit('addProgramToSavedPrograms', savedProgram);
  }).catch((errorResponse) => {
    console.log(errorResponse.response);
    this.showWarningAlert('Program NOT saved.' +
errorResponse.response.data.message);
  });
},

showSuccessAlert(alertText) {
  this.successAlertText = alertText;
  this.dismissCountDownSuccess = this.dismissSecs
},

showWarningAlert(alertText) {
  this.warningAlertText = alertText;
  this.dismissCountDownWarning = this.dismissSecs
},

deleteProgram() {
  let programs = this.$store.state.savedPrograms;
  const minProgramId = Math.min(...programs.map(program =>
program.id));
  const selectedProgram = programs.find(program => program.id ===
minProgramId);

  this.$store.dispatch('deleteProgramAction', this.selectedProgram);
  this.$store.commit('setSelectedProgram', selectedProgram);
}
},
};
</script>

```

Д.8 Код файла front-end/src/components/workspace/code-mirror-editor.vue

```

<template>
  <b-container fluid>

```

```

<b-row>
  <code-mirror-control-panel></code-mirror-control-panel>
</b-row>

<b-row>
  <b-container fluid>
    <codemirror v-model="code"
      :options="cmOptions"
      class="CodeMirrorEditor text-left border row"/>
  </b-container>
</b-row>
</b-container>
</template>

<script>
import { codemirror } from 'vue-codemirror';
import codeMirrorControlPanel from './code-mirror-control-panel.vue';

import 'codemirror/lib/codemirror.css';

// language
import 'codemirror/mode/python/python';
// theme css
import 'codemirror/theme/3024-day.css';
import 'codemirror/theme/3024-night.css';
import 'codemirror/theme/abcdef.css';
import 'codemirror/theme/ambiance.css';
import 'codemirror/theme/ambiance-mobile.css';
import 'codemirror/theme/base16-dark.css';
import 'codemirror/theme/base16-light.css';
import 'codemirror/theme/bespin.css';
import 'codemirror/theme/blackboard.css';
import 'codemirror/theme/cobalt.css';
import 'codemirror/theme/colorforth.css';
import 'codemirror/theme/darcula.css';
import 'codemirror/theme/dracula.css';
import 'codemirror/theme/duotone-dark.css';
import 'codemirror/theme/duotone-light.css';
import 'codemirror/theme/eclipse.css';
import 'codemirror/theme/elegant.css';
import 'codemirror/theme/erlang-dark.css';
import 'codemirror/theme/gruvbox-dark.css';
import 'codemirror/theme/hopscotch.css';
import 'codemirror/theme/icecoder.css';

```

```
import 'codemirror/theme/idea.css';
import 'codemirror/theme/isotope.css';
import 'codemirror/theme/lesser-dark.css';
import 'codemirror/theme/liquibyte.css';
import 'codemirror/theme/lucario.css';
import 'codemirror/theme/material.css';
import 'codemirror/theme/material-darker.css';
import 'codemirror/theme/material-ocean.css';
import 'codemirror/theme/material-palenight.css';
import 'codemirror/theme/mbo.css';
import 'codemirror/theme/mdn-like.css';
import 'codemirror/theme/midnight.css';
import 'codemirror/theme/monokai.css';
import 'codemirror/theme/moxer.css';
import 'codemirror/theme/neat.css';
import 'codemirror/theme/neo.css';
import 'codemirror/theme/night.css';
import 'codemirror/theme/nord.css';
import 'codemirror/theme/oceanic-next.css';
import 'codemirror/theme/panda-syntax.css';
import 'codemirror/theme/paraiso-dark.css';
import 'codemirror/theme/paraiso-light.css';
import 'codemirror/theme/pastel-on-dark.css';
import 'codemirror/theme/railscasts.css';
import 'codemirror/theme/rubyblue.css';
import 'codemirror/theme/seti.css';
import 'codemirror/theme/shadowfox.css';
import 'codemirror/theme/solarized.css';
import 'codemirror/theme/ssms.css';
import 'codemirror/theme/the-matrix.css';
import 'codemirror/theme/tomorrow-night-bright.css';
import 'codemirror/theme/tomorrow-night-eighties.css';
import 'codemirror/theme/ttcn.css';
import 'codemirror/theme/twilight.css';
import 'codemirror/theme/vibrant-ink.css';
import 'codemirror/theme/xq-dark.css';
import 'codemirror/theme/xq-light.css';
import 'codemirror/theme/yeti.css';
import 'codemirror/theme/yonce.css';
import 'codemirror/theme/zenburn.css';

// active-line.js
import 'codemirror/addon/selection/active-line';
// styleSelectedText
```

```

import 'codemirror/addon/selection/mark-selection';
import 'codemirror/addon/search/searchcursor';
// highlightSelectionMatches
import 'codemirror/addon/scroll/annotatescrollbar';
import 'codemirror/addon/search/matchesonscrollbar';

import 'codemirror/addon/search/match-highlighter';
// keyMap
import 'codemirror/mode/clike/clike';
import 'codemirror/addon/edit/matchbrackets';
import 'codemirror/addon/comment/comment';
import 'codemirror/addon/dialog/dialog';
import 'codemirror/addon/dialog/dialog.css';

import 'codemirror/addon/search/search';
import 'codemirror/keymap/sublime';
// foldGutter
import 'codemirror/addon/fold/foldgutter.css';
import 'codemirror/addon/fold/brace-fold';
import 'codemirror/addon/fold/comment-fold';
import 'codemirror/addon/fold/foldcode';
import 'codemirror/addon/fold/foldgutter';
import 'codemirror/addon/fold/indent-fold';
import 'codemirror/addon/fold/markdown-fold';
import 'codemirror/addon/fold/xml-fold';

export default {
  components: {
    codemirror,
    codeMirrorControlPanel,
  },
  name: 'code-mirror-editor',

  data() {
    return {
      cmOptions: {
        mode: 'text/x-python',
        tabSize: 1,
        styleActiveLine: true,
        lineNumbers: true,
        line: true,
        foldGutter: true,
        styleSelectedText: true,
        keyMap: 'sublime',
      },
    };
  },
};

```

```

    matchBrackets: true,
    showCursorWhenSelecting: true,
    theme: 'default',
    extraKeys: { Ctrl: 'autocomplete' },
    hintOptions: {
      completeSingle: false,
    },
  },
};
},
computed: {
  selectedCodemirrorTheme() {
    return this.$store.state.codemirrorTheme;
  },
  code: {
    get() {
      return this.selectedProgram.program;
    },
    set(newCode) {
      const { selectedProgram } = this;

      if (selectedProgram.program !== newCode) {
        selectedProgram.program = newCode;
        selectedProgram.changed = true;
        this.$store.commit('setSelectedProgram', selectedProgram);
        this.$store.commit('updateSavedPrograms', selectedProgram);
      }
    },
  },
  selectedProgram() {
    if (this.$store.state.selectedProgram == null) {
      return "";
    }
    return this.$store.state.selectedProgram;
  },
},
watch: {
  created() {

  },
};
</script>

```

Д.9 Код файла front-end/src/components/workspace/code-result-output.vue

```

<template>
  <b-container fluid>
    <b-row id="dataVisualisation">
      <h2>Data visualisation</h2>
    </b-row>
    <b-row class="halfViewPortHeight" id="jsChart">
      <highcharts :options="chartOptions"></highcharts>
    </b-row>

    <b-row class="d-none" id="pyPlotDiv">
      <b-img id="pyPlotFigure"/>
    </b-row>

    <b-row class="halfViewPortHeight">
      <h2 style="max-height: 1.5em">Python console</h2>
      <b-form-textarea id="pythonCodeOutput" class="bg-light" plaintext
rows="15"></b-form-textarea>
    </b-row>
  </b-container>
</template>

<script>
import { Chart } from 'highcharts-vue';

export default {
  name: 'code-result-output',
  components: {
    highcharts: Chart,
  },
  data() {
    return {
      chartOptions: {
        chart: {
          type: 'line',
        },
        series: [{
          data: [
            [0, 29.9],

```

```

        [1, 71.5],
        [3, 106.4],
    ],
    }],
  },
};
},
computed: {
  pythonCodeErrors() {
    return this.$store.state.pythonCodeErrors;
  },
},
watch: {
  '$store.state.codeResultOutput': {
    handler(chartData) {
      this.chartOptions.series = chartData;
    },
    deep: true,
  },
},
methods: {
},
};
</script>

<style scoped>

</style>

```

Д.10 Код файла front-end/src/components/workspace/saved-programs-list.vue

```

<template>
  <div>
    <span>The list of saved methods</span>

    <b-list-group class="text-left">
      <b-list-group-item button v-for="(program) in programs"
        v-bind:key="program.id"
        :class="{ 'active': (selectedProgram.id) === program.id }"
        @click="openProgram(program.id)">
        {{ program.name }}
    </b-list-group>
  </div>
</template>

```

```

        <b-badges variant="warning"
          :class="{ 'd-none': program.changed === false }">Unsaved</b-
badge>
      </b-list-group-item>
    </b-list-group>
  </div>
</template>

<script>
export default {
  name: 'saved-programs-list',

  data() {
    return {
    };
  },

  methods: {
    openProgram(programId) {
      const programs = this.$store.state.savedPrograms;
      const selectedProgram = programs.find(program => program.id ===
programId);

      this.$store.commit('setSelectedProgram', selectedProgram);
    },
  },
  computed: {
    selectedProgram() {
      const { selectedProgram } = this.$store.state;
      return selectedProgram;
    },
  },
};
</script>

```

Д.11 Код файла front-end/src/components/workspace/nav-header.vue

```

<template>
  <b-navbar toggleable="lg" type="dark" variant="primary">
    <b-navbar-brand to="/">Project name</b-navbar-brand>

    <b-navbar-toggle target="nav-collapse"></b-navbar-toggle>

```

```

<b-collapse id="nav-collapse" is-nav>
  <b-navbar-nav>
    <b-nav-item to="/">Home</b-nav-item>
    <b-nav-item to="/workspace">Workspace</b-nav-item>
    <b-nav-item to="/faq">About</b-nav-item>
  </b-navbar-nav>

  <b-navbar-nav class="ml-auto">
    <b-nav-item v-for="item in menuItems"
      v-bind:key="item.title"
      :to="item.link">
      {{ item.title }}
    </b-nav-item>

    <b-nav-item v-if="userIsAuthenticated"
      @click="onLogout">
      Logout
    </b-nav-item>
  </b-navbar-nav>
</b-collapse>
</b-navbar>
</template>

<script>
export default {
  name: 'nav-header',
  data() {
    return {

    };
  },
  computed: {
    menuItems() {
      let menuItems = [
        { title: 'Register', link: '/register' },
        { title: 'Login', link: '/login' },
      ];

      if (this.userIsAuthenticated) {
        menuItems = [];
      }

      return menuItems;
    },
  },

```

```

    userIsAuthenticated() {
      return !!this.$store.getters.user.token;
    },
  },
  methods: {
    onLogout() {
      const user = {
        token: null,
      };

      localStorage.removeItem('user-token');
      this.$store.commit('setSelectedProgram', null);
      this.$store.commit('setSavedPrograms', []);
      this.$store.commit('setUser', user);
      this.$store.commit('setCodeResultOutput', '');
      this.$store.commit('setPythonCodeErrors', '');

      this.$router.push({ name: 'home' });
    },
  },
};
</script>

```

Д.12 Код файла front-end/src/pyodide/pyodide.js

```

/**
 * The main bootstrap script for loading pyodide.
 */

var languagePluginLoader = new Promise((resolve, reject) => {
  // This is filled in by the Makefile to be either a local file or the
  // deployed location. TODO: This should be done in a less hacky
  // way.
  var baseUrl = self.languagePluginUrl || 'https://pyodide-
cdn2.iodide.io/v0.15.0/full/';
  baseUrl = baseUrl.substr(0, baseUrl.lastIndexOf('/') + 1);

  //////////////////////////////////////
  // Package loading
  let loadedPackages = {};
  var loadPackagePromise = new Promise((resolve) => resolve());

```

```

// Regexp for validating package name and URI
var package_name_regexp = '[a-z0-9_][a-z0-9_\-]*'
var package_uri_regexp =
  new RegExp('^https?://.*?(' + package_name_regexp + ').js$', 'i');
var package_name_regexp = new RegExp('^' + package_name_regexp + '$',
'i');

let _uri_to_package_name = (package_uri) => {
  // Generate a unique package name from URI

  if (package_name_regexp.test(package_uri)) {
    return package_uri;
  } else if (package_uri_regexp.test(package_uri)) {
    let match = package_uri_regexp.exec(package_uri);
    // Get the regexp group corresponding to the package name
    return match[1];
  } else {
    return null;
  }
};

// clang-format off
let preloadWasm = () => {
  // On Chrome, we have to instantiate wasm asynchronously. Since that
  // can't be done synchronously within the call to dlopen, we instantiate
  // every .so that comes our way up front, caching it in the
  // `preloadedWasm` dictionary.

  let promise = new Promise((resolve) => resolve());
  let FS = pyodide._module.FS;

  function recurseDir(rootpath) {
    let dirs;
    try {
      dirs = FS.readdir(rootpath);
    } catch {
      return;
    }
    for (let entry of dirs) {
      if (entry.startsWith('.')) {
        continue;
      }
      const path = rootpath + entry;
      if (entry.endsWith('.so')) {

```

```

    if (Module['preloadedWasm'][path] === undefined) {
      promise = promise
        .then(() => Module['loadWebAssemblyModule'](
          FS.readFile(path), {loadAsync: true}))
        .then((module) => {
          Module['preloadedWasm'][path] = module;
        });
    }
  } else if (FS.isDir(FS.lookupPath(path).node.mode)) {
    recurseDir(path + '/');
  }
}
}

recurseDir('/');

return promise;
}
// clang-format on

function loadScript(url, onload, onerror) {
  if (self.document) { // browser
    const script = self.document.createElement('script');
    script.src = url;
    script.onload = (e) => { onload(); };
    script.onerror = (e) => { onerror(); };
    self.document.head.appendChild(script);
  } else if (self.importScripts) { // webworker
    try {
      self.importScripts(url);
      onload();
    } catch {
      onerror();
    }
  }
}

let _loadPackage = (names, messageCallback, errorCallback) => {
  if (messageCallback === undefined) {
    messageCallback = () => {};
  }
  if (errorCallback === undefined) {
    errorCallback = () => {};
  }
}

```

```

let _messageCallback = (msg) => {
  console.log(msg);
  messageCallback(msg);
};
let _errorCallback = (errMsg) => {
  console.error(errMsg);
  errorCallback(errMsg);
};

// DFS to find all dependencies of the requested packages
let packages = self.pyodide._module.packages.dependencies;
let loadedPackages = self.pyodide.loadedPackages;
let queue = [].concat(names || []);
let toLoad = {};
while (queue.length) {
  let package_uri = queue.pop();

  const pkg = _uri_to_package_name(package_uri);

  if (pkg == null) {
    _errorCallback(`Invalid package name or URI '${package_uri}'`);
    return;
  } else if (pkg == package_uri) {
    package_uri = 'default channel';
  }

  if (pkg in loadedPackages) {
    if (package_uri != loadedPackages[pkg]) {
      _errorCallback(`URI mismatch, attempting to load package ` +
        `${pkg} from ${package_uri} while it is already ` +
        `loaded from ${loadedPackages[pkg]}!`);
      return;
    } else {
      _messageCallback(`${pkg} already loaded from
${loadedPackages[pkg]}`)
    }
  } else if (pkg in toLoad) {
    if (package_uri != toLoad[pkg]) {
      _errorCallback(`URI mismatch, attempting to load package ` +
        `${pkg} from ${package_uri} while it is already ` +
        `being loaded from ${toLoad[pkg]}!`);
      return;
    }
  } else {

```

```

console.log(
  `${pkg} to be loaded from ${package_uri}`); // debug level info.

toLoad[pkg] = package_uri;
if (packages.hasOwnProperty(pkg)) {
  packages[pkg].forEach((subpackage) => {
    if (!(subpackage in loadedPackages) && !(subpackage in toLoad)) {
      queue.push(subpackage);
    }
  });
} else {
  _errorCallback(`Unknown package '${pkg}'`);
}
}
}

self.pyodide._module.locateFile = (path) => {
  // handle packages loaded from custom URLs
  let pkg = path.replace(/\.data$/, "");
  if (pkg in toLoad) {
    let package_uri = toLoad[pkg];
    if (package_uri !== 'default channel') {
      return package_uri.replace(/\.js$/, ".data");
    }
  };
  return baseUrl + path;
};

let promise = new Promise((resolve, reject) => {
  if (Object.keys(toLoad).length === 0) {
    resolve('No new packages to load');
    return;
  }
});

let packageList = Array.from(Object.keys(toLoad));
_messageCallback(`Loading ${packageList.join(', ')}`)

// monitorRunDependencies is called at the beginning and the end of each
// package being loaded. We know we are done when it has been called
// exactly "toLoad * 2" times.
var packageCounter = Object.keys(toLoad).length * 2;

self.pyodide._module.monitorRunDependencies = () => {
  packageCounter--;

```

```

if (packageCounter === 0) {
  for (let pkg in toLoad) {
    self.pyodide.loadedPackages[pkg] = toLoad[pkg];
  }
  delete self.pyodide._module.monitorRunDependencies;
  self.removeEventListener('error', windowErrorHandler);

  let resolveMsg = `Loaded `;
  if (packageList.length > 0) {
    resolveMsg += packageList.join(', ');
  } else {
    resolveMsg += 'no packages'
  }

  if (!isFirefox) {
    preloadWasm().then(() => {
      console.log(resolveMsg);
      resolve(resolveMsg);
    });
  } else {
    console.log(resolveMsg);
    resolve(resolveMsg);
  }
}
};

////////////////////////////////////
// Add a handler for any exceptions that are thrown in the process of
// loading a package
var windowErrorHandler = (err) => {
  delete self.pyodide._module.monitorRunDependencies;
  self.removeEventListener('error', windowErrorHandler);
  // Set up a new Promise chain, since this one failed
  loadPackagePromise = new Promise((resolve) => resolve());
  reject(err.message);
};
self.addEventListener('error', windowErrorHandler);

for (let pkg in toLoad) {
  let scriptSrc;
  let package_uri = toLoad[pkg];
  if (package_uri === 'default channel') {
    scriptSrc = `${baseUrl}${pkg}.js`;
  } else {

```

```

    scriptSrc = `${package_uri}`;
  }
  _messageCallback(`Loading ${pkg} from ${scriptSrc}`)
  loadScript(scriptSrc, () => {}, () => {
    // If the package_uri fails to load, call monitorRunDependencies twice
    // (so packageCounter will still hit 0 and finish loading), and remove
    // the package from toLoad so we don't mark it as loaded, and remove
    // the package from packageList so we don't say that it was loaded.
    _errorCallback(`Couldn't load package from URL ${scriptSrc}`);
    delete toLoad[pkg];
    let packageListIndex = packageList.indexOf(pkg);
    if (packageListIndex !== -1) {
      packageList.splice(packageListIndex, 1);
    }
    for (let i = 0; i < 2; i++) {
      self.pyodide._module.monitorRunDependencies();
    }
  });
}

////////////////////////////////////
// We have to invalidate Python's import caches, or it won't
// see the new files. This is done here so it happens in parallel
// with the fetching over the network.
self.pyodide.runPython('import importlib as _importlib\n' +
  '_importlib.invalidate_caches()\n');
});

return promise;
};

let loadPackage = (names, messageCallback, errorCallback) => {
  /* We want to make sure that only one loadPackage invocation runs at any
  * given time, so this creates a "chain" of promises. */
  loadPackagePromise = loadPackagePromise.then(
    () => _loadPackage(names, messageCallback, errorCallback));
  return loadPackagePromise;
};

////////////////////////////////////
// Fix Python recursion limit
function fixRecursionLimit(pyodide) {
  // The Javascript/Wasm call stack may be too small to handle the default
  // Python call stack limit of 1000 frames. This is generally the case on

```

```
// Chrom(ium), but not on Firefox. Here, we determine the Javascript call
Module.noWasmDecoding = true;
// to set the maximum Python call stack depth.
Module.noWasmDecoding = true;
```

```
let depth = 0;
function recurse() {
  depth += 1;
  recurse();
}
try {
  recurse();
} catch (err) {
  ;
}
```

```
let recursionLimit = depth / 50;
if (recursionLimit > 1000) {
  recursionLimit = 1000;
}
pyodide.runPython(
  `import sys; sys.setrecursionlimit(int(${recursionLimit}))`);
};
```

```
////////////////////////////////////
```

```
// Rearrange namespace for public API
```

```
let PUBLIC_API = [
  'globals',
  'loadPackage',
  'loadedPackages',
  'pyimport',
  'repr',
  'runPython',
  'runPythonAsync',
  'checkABI',
  'version',
  'autocomplete',
];
```

```
////////////////////////////////////
```

```
function makePublicAPI(module, public_api) {
  var namespace = {_module : module};
  for (let name of public_api) {
```

```

    namespace[name] = module[name];
  }
  return namespace;
}

////////////////////////////////////
// Loading Pyodide
let wasmURL = `${baseUrl}pyodide.asm.wasm`;
let Module = {};
self.Module = Module;

Module.noImageDecoding = true;
Module.noAudioDecoding = true;
Module.noWasmDecoding = true;
Module.noWasmDecoding = true;
Module.noWasmDecoding = true;
Module.preloadedWasm = {};
let isFirefox = navigator.userAgent.toLowerCase().indexOf('firefox') > -1;

////////////////////////////////////
let wasm_promise, wasm_fetch = fetch(wasmURL);
const compileBuffer = () =>
  wasm_fetch.then(response => response.arrayBuffer())
    .then(bytes => WebAssembly.compile(bytes));
if (WebAssembly.compileStreaming === undefined) {
  wasm_promise = compileBuffer();
} else {
  wasm_promise = WebAssembly.compileStreaming(wasm_fetch);
  wasm_promise = wasm_promise.catch(e => {
    if (e instanceof TypeError) {
      console.error("pyodide streaming compilation failed:", e,
        "- falling back to buffered compilation");
      return compileBuffer()
    }
    throw e;
  });
}

////////////////////////////////////
Module.instantiateWasm = (info, receiveInstance) => {
  wasm_promise.then(module => WebAssembly.instantiate(module, info))
    .then(instance => receiveInstance(instance));
  return {};
};

```

```

Module.checkABI = function(ABI_number) {
  if (ABI_number !== parseInt('1')) {
    var ABI_mismatch_exception =
      `ABI numbers differ. Expected 1, got ${ABI_number}`;
    console.error(ABI_mismatch_exception);
    throw ABI_mismatch_exception;
  }
  return true;
};

// the package from packageList so we don't say that it was loaded.

```

```

Module.autocomplete =
function(path) {
  var pyodide_module = Module.pyimport("pyodide");
  return pyodide_module.get_completions(path);
}

```

```

Module.locateFile = (path) => baseURL + path;
var postRunPromise = new Promise((resolve, reject) => {
  Module.postRun = () => {
    delete self.Module;
    fetch(`${baseURL}packages.json`)
      .then((response) => response.json())
      .then((json) => {
        fixRecursionLimit(self.pyodide);
////////////////////////////////////
        self.pyodide.globals =
          self.pyodide.runPython('import sys\nsys.modules["__main__"]');
        self.pyodide = makePublicAPI(self.pyodide, PUBLIC_API);
        self.pyodide._module.packages = json;
////////////////////////////////////
        if (self.iodide !== undefined) {
          // Perform some completions immediately so there isn't a delay on
          // the first call to autocomplete
          self.pyodide.runPython('import pyodide');
          self.pyodide.runPython('pyodide.get_completions("")');
        }
        resolve();
      });
  });
});

```

```

////////////////////////////////////

```

```

var dataLoadPromise = new Promise((resolve, reject) => {
  Module.monitorRunDependencies =
    (n) => {
      if (n === 0) {
        delete Module.monitorRunDependencies;
        resolve();
      }
    }
});

```

```

Promise.all([ postRunPromise, dataLoadPromise ]).then(() => resolve());

```

```

const data_script_src = `${baseUrl}pyodide.asm.data.js`;
loadScript(data_script_src, () => {
  const scriptSrc = `${baseUrl}pyodide.asm.js`;
  loadScript(scriptSrc, () => {
    // The emscripten module needs to be at this location for the core
    // filesystem to install itself. Once that's complete, it will be replaced
    // by the call to `makePublicAPI` with a more limited public API.
    self.pyodide = pyodide(Module);
    self.pyodide.loadedPackages = {};
    self.pyodide.loadPackage = loadPackage;
  }, () => {});
}, () => {});

```

```

////////////////////////////////////
// Iodide-specific functionality, that doesn't make sense
// if not using with Iodide.
if (self.iodide !== undefined) {
  // Load the custom CSS for Pyodide
  let link = document.createElement('link');
  link.rel = 'stylesheet';
  link.type = 'text/css';
  link.href = `${baseUrl}renderedhtml.css`;
  document.getElementsByTagName('head')[0].appendChild(link);

```

```

// Add a custom output handler for Python objects
self.iodide.addOutputRenderer({
  shouldRender : (val) => {
    return (typeof val === 'function' &&
      pyodide._module.PyProxy.isPyProxy(val));
  },

```

```

render : (val) => {
  let div = document.createElement('div');
  div.className = 'rendered_html';
  var element;
  if (val._repr_html_ !== undefined) {
    let result = val._repr_html_();
    if (typeof result === 'string') {
      div.appendChild(new DOMParser()
        .parseFromString(result, 'text/html')
        .body.firstChild);
      element = div;
    } else {
      element = result;
    }
  } else {
    let pre = document.createElement('pre');
    pre.textContent = val.toString();
    div.appendChild(pre);
    element = div;
  }
  return element.outerHTML;
}
});
}
});
////////////////////////////////////

// languagePluginLoader
export function test() {
  return languagePluginLoader;
}

```

Д.13 Код файла front-end/src/router/index.js

```

import Vue from 'vue';
import VueRouter from 'vue-router';
import Home from '../views/Home.vue';
import Login from '../views/Login.vue';
import Register from '../views/Register.vue';
import Workspace from '../views/Workspace.vue';
// import AuthGuard from './auth-guard';
// import About from '../views/About.vue';

```

```

Vue.use(VueRouter);

const routes = [
  {
    path: '/',
    name: 'home',
    component: Home,
  },
  {
    path: '/login',
    name: 'login',
    component: Login,
  },
  {
    path: '/register',
    name: 'register',
    component: Register,
  },
  {
    path: '/workspace',
    name: 'workspace',
    component: Workspace,
    // beforeEnter: AuthGuard,
  },
  {
    path: '/faq',
    name: 'faq',
    // route level code-splitting
    // this generates a separate chunk (about.[hash].js) for this route
    // which is lazy-loaded when the route is visited.
    component: () => import(/* webpackChunkName: "about" */
'../views/About.vue'),
  },
];

const router = new VueRouter({
  mode: 'history',
  base: process.env.BASE_URL,
  routes,
});

export default router;

```

Д.14 Код файла front-end/src/router/auth-guard.js

```
import store from '../store';

export default (to, from, next) => {
  if (store.getters.user.token) {
    next();
  } else {
    next('/login');
  }
};
```

Д.16 Код файла front-end/src/views/Workspace.vue

```
<template>
  <b-container fluid>
    <b-row class="ml-1 mr-1">
      <b-col sm="3" md="2">
        <saved-programs-list></saved-programs-list>
      </b-col>

      <b-col sm="9" md="10">
        <b-row class="mb-3">
          <code-mirror-editor></code-mirror-editor>
        </b-row>

        <b-row>
          <code-result-output></code-result-output>
        </b-row>
      </b-col>
    </b-row>
  </b-container>
</template>

<script>
import CodeMirrorEditor from '../components/workspace/code-mirror-
editor.vue';
import SavedProgramsList from '../components/workspace/saved-programs-
list.vue';
```

```

import CodeResultOutput from '../components/workspace/code-result-
output.vue';

export default {
  name: 'Workspace',
  components: {
    CodeMirrorEditor,
    CodeResultOutput,
    SavedProgramsList,
  },

  methods: {
    createFirstProgram() {
      const newProgram = {
        id: 1,
        name: 'New program',
        program: '',
        changed: true,
      };

      this.$store.commit('setSelectedProgram', newProgram);
      this.$store.commit('addProgramToSavedPrograms', newProgram);
    },
    selectProgramWithMinId(programs) {
      const minProgramId = Math.min(...programs.map(program =>
program.id));
      const selectedProgram = programs.find(program => program.id ===
minProgramId);
      this.$store.commit('setSelectedProgram', selectedProgram);
    },
  },
  created() {
    const programs = this.$store.state.savedPrograms;

    if (programs.length === 0) {
      this.createFirstProgram();
    } else {
      this.selectProgramWithMinId(programs);
    }
  },
};
</script>

```

Д.17 Код файла back-end/setup.cfg

```
[coverage:report]
precision = 2
exclude_lines =
    pragma: no cover
    raise NotImplementedError
    raise NotImplemented

[coverage:run]
source = app
branch = True

[mypy]
plugins = pydantic.mypy

strict_optional = True
warn_redundant_casts = True
warn_unused_ignores = True
disallow_any_generics = True
check_untyped_defs = True

disallow_untyped_defs = True

[pydantic-mypy]
init_forbid_extra = True
init_typed = True
warn_required_dynamic_aliases = True
warn_untyped_fields = True

[mypy-loguru]
ignore_missing_imports = True

[mypy-asyncpg.*]
ignore_missing_imports = True

[mypy-bcrypt]
ignore_missing_imports = True

[mypy-passlib.*]
ignore_missing_imports = True

[mypy-slugify.*]
```

```

ignore_missing_imports = True

[mypy-pypika.*]
ignore_missing_imports = True

[mypy-slugify.*]
ignore_missing_imports = True

[mypy-pypika.*]
ignore_missing_imports = True

[flake8]
format = wemake
max-line-length = 88
per-file-ignores =
    # ignore error on builtin names for TypedTable classes, since just mapper for
SQL table
    app/db/queries/tables.py: WPS125,

    # ignore black disabling in some places for queries building using pypika
    app/db/repositories/*.py: E800,
ignore =
    # common errors:
    # FastAPI architecture requires a lot of functions calls as default arguments,
so ignore it here.
    B008,
    # docs are missing in this project.
    D, RST

    # WPS: 3xx
    # IMO, but the obligation to specify the base class is redundant.
    WPS306,

    # WPS: 4xx
    # FastAPI architecture requires a lot of complex calls as default arguments,
so ignore it here.
    WPS404,
    # again, FastAPI DI architecture involves a lot of nested functions as DI
providers.
    WPS430,

```

```

# WPS: 6xx
# pydantic defines models in dataclasses model style, but not supported by
WPS.
    WPS601,
no-accept-encodings = True
nested-classes-whitelist=Config
inline-quotes = double

```

Д.18 Код файла **back-end/app/models/domain/users.py**

```

from typing import Optional

from app.models.common import DateTimeModelMixin, IDModelMixin
from app.models.domain.rwmodel import RWModel
from app.services import security
class User(RWModel):
    username: str
    email: str
    image: Optional[str] = None

class UserInDB(IDModelMixin, DateTimeModelMixin, User):
    salt: str = ""
    hashed_password: str = ""

    def check_password(self, password: str) -> bool:
        return security.verify_password(self.salt + password,
self.hashed_password)

    def change_password(self, password: str) -> None:
        self.salt = security.generate_salt()
        self.hashed_password = security.get_password_hash(self.salt +
password)

```

Д.19 Код файла **back-end/app/models/domain/programs.py**

```

from typing import List

```

```

from app.models.common import DateTimeModelMixin, IDModelMixin

class Programs(IDModelMixin, DateTimeModelMixin, RWModel):
    name: str
    code: str
    author: Profile

```

Д.21 Код файла `back-end/app/api/routes/comments.py`

```

from typing import Optional

from fastapi import APIRouter, Body, Depends
from starlette import status

from app.api.dependencies.programs import get_programs
from app.api.dependencies.authentication import get_current_user_authorizer
from app.api.dependencies.programs import get_programs
from app.api.dependencies.programs import get_programs
from app.api.dependencies.programs import get_programs
from app.api.dependencies.database import get_repository
from app.db.repositories.comments import CommentsRepository
from app.models.domain.programs import Program

router = APIRouter()

@router.get(
    "",
    response_model=ListOfProgramsInResponse,
    name="programs:get-programs ",
)

@router.post(
    "",
    status_code=status.HTTP_201_CREATED,
    response_model=ProgramInResponse,
    name="program:create-program",
)

@router.delete(
   ("/{ program_id} ",

```

```
status_code=status.HTTP_204_NO_CONTENT,  
name="comments:delete- program",  
dependencies=[Depends(check_program_modification_permissions)],  
)  
async def delete_program(  
    program: Program = Depends(get_program_by_id_from_path),  
) -> None:  
    await programs_repo.delete_program(program=program)
```

Додаток Е
Слайди презентації

Дослідження, розробка та програмна реалізація інтерактивної платформи для інтелектуального аналізу даних

Виконав:
Кулик Л.І., Студент гр. Кнт-119м

Керівник:
Табунщик Г.В., професор, к.т.н., доцент

1

Рисунок Е.1 – Слайд №1

Об'єкт, предмет і мета розробки

- Об'єкт розробки – процес інтелектуального аналізу даних.
- Предмет розробки – інтелектуальний аналіз даних з використанням веб-технологій.
- Мета розробки - покращити процес інтелектуального аналізу даних за допомогою створення уніфікованого середовища.

2

Рисунок Е.2 – Слайд №2

Огляд існуючих рішень

Критерій порівняння	Програмні продукти			
	Jupyter Notebook	Apache Zeppelin	Iodide	Програма, що розробляється
Виконання програми на сервері	+	+	-	-
Виконання програми у браузері	-	-	+	+
Збереження програми	+	+	-	+
Просте відтворення коду на інших машинах	-	-	+	+
Завантаження даних з багатьох джерел	-	+	±	±
Простий та зручний інтерфейс	-	-	±	+

3

Рисунок Е.3 – Слайд №3

Формальна постановка задачі

Розроблюване ПЗ має відповідати наступним основним вимогам:

- запуск Python скриптів у веб-браузері;
- підтримка популярних бібліотек для роботи з даними у Python;
- збереження розроблених програм;
- завантаження даних з файлів;
- відображення результатів виконання програми;
- авторизація для створення Python скриптів;
- запуск програми та відтворення її результатів на різних персональних комп'ютерах(ПК);

4

Рисунок Е.4 – Слайд №4

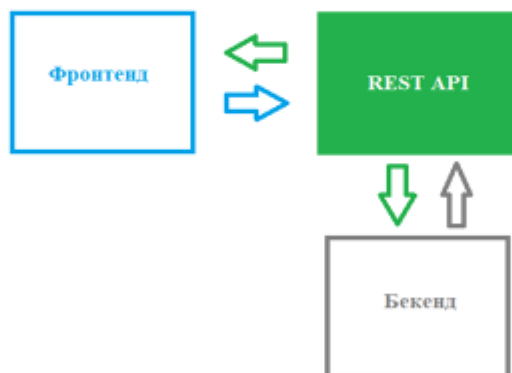
Порівняння методів запуску користувацьких Python скриптів

Методи	Критерії			
	Місце виконання	Час виконання	Складність реалізації	Вимоги до ресурсів
Використання <code>py sandbox</code>	Сервер	JIT	+	Середні
Використання вісочинці PyPy Python	Сервер	JIT	++	Середні
Використання можливостей ОС Linux	Сервер	JIT	+++	Середні
Використання Docker	Сервер	JIT	++	Високі
Створення мультисерверної архітектури	Сервер	JIT	+++	Високі
Компіляція Python до JavaScript	Клієнт	AOT	++	Низькі
Створення <code>runtime</code> Python	Клієнт	JIT	++	Низькі

5

Рисунок Е.5 – Слайд №5

Загальна архітектура системи



6

Рисунок Е.6 – Слайд №6

Основні програмні модулі бекенду



7

Рисунок Е.7 – Слайд №7

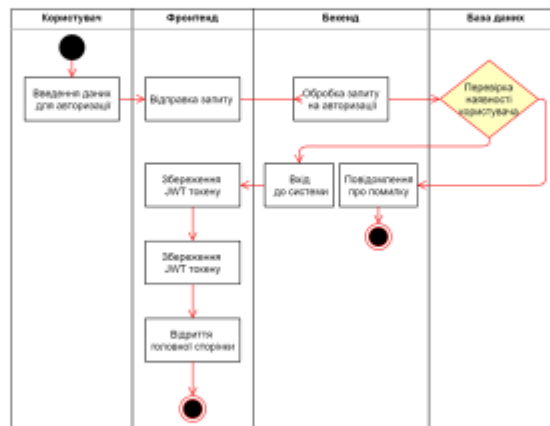
Основні програмні модулі фронтенду



8

Рисунок Е.8 – Слайд №8

Діаграма діяльності авторизації користувача



9

Рисунок Е.9 – Слайд №9

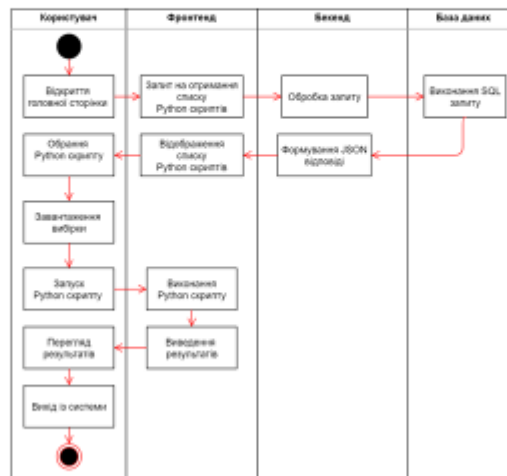
Діаграма діяльності створення нового скрипта



10

Рисунок Е.10 – Слайд №10

Діаграма діяльності використання збереженого скрипта



11

Рисунок Е.11 – Слайд №11

Порівняння мов програмування для веб-сервера

Критерій	Мови програмування			
	Python	PHP	JavaScript	C#
Динамічна типізація	+	+	+	+
Явна типізація	±	±	-	+
Неявна типізація	+	+	+	±
Багатоплатформність	+	+	+	-
Підтримка інтерфейсів	+	+	±	+
Декларація типів та повертаємих значень	+	+	-	+

12

Рисунок Е.12 – Слайд №12

Порівняння Python фреймворків

Критерій	Фреймворки		
	Fastapi	Django	flask
ORM за замовчуванням	SqlAlchemy	Django ORM	немає
Перший випуск	2019	2005	2010
Мікрофреймворк	+	-	+
Зосереджений на створенні API	+	±	+
Активне використання можливостей Python 3.6+	+	-	-
Вбудована асинхронність	+	-	+

13

Рисунок Е.13 – Слайд №13

	Vue.js	Angular	React
Приблизний розмір фреймворку	80КБ	500КБ	100КБ
Кількість зірочок на github	122 тис.	43 тис.	71 тис.
Підтримка	Створено колишнім співробітником Google	Розроблено в Google	Підтримується Facebook
Написано на	JavaScript	TypeScript	JavaScript
Ідеально підходить для	Веб-розробка та односторінкові програми	Масштабні, багатофункціональні програми	Сучасна веб-розробка та нативні програми для iOS та Android

14

Рисунок Е.14 – Слайд №14

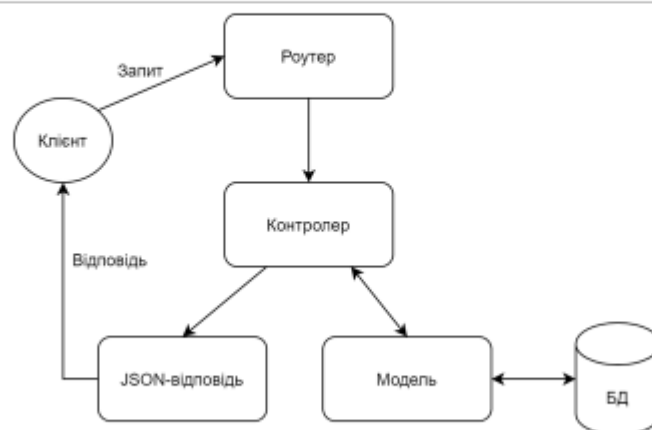
Порівняння середовищ розробки

Критерії	Середовища розробки		
	Pycharm	VS Code	Komodo
Інтегроване середовище розробки	+	-	+
Безкоштовна версія для студентів	+	+	+
Кросплатформеність	+	+	+
Робота з БД	+	±	-
Підтримка Docker	+	±	-
Інструменти розробки	+	±	±

15

Рисунок Е.15 – Слайд №15

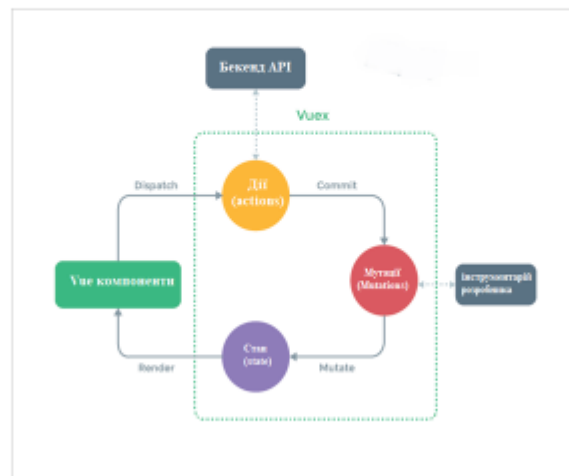
Архітектура Python веб-застосунок



16

Рисунок Е.16 – Слайд №16

Архітектура фронтенду

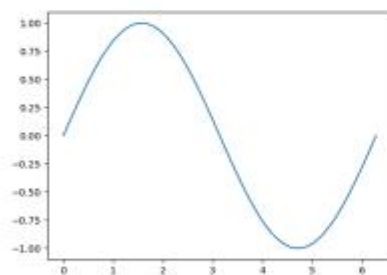


17

Рисунок Е.17 – Слайд №17

Результати роботи

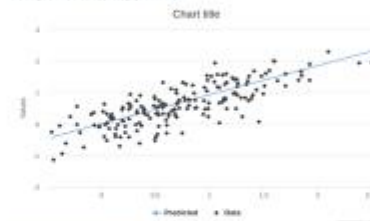
Візуалізація даних



Консоль Python

matplotlib already loaded from default channel
pandas already loaded from default channel
numpy already loaded from default channel

Візуалізація даних



Консоль Python

kwik solver to be loaded from default channel
cycler to be loaded from default channel
pandas to be loaded from default channel
Loading matplotlib, pyplot, python-dateutil, peewee, numpy, kwi solver, cycler, pandas
Loading matplotlib from https://pyodide-cdn2.iodide.io/v0.15.0/full/matplotlib.js
Loading pyplot from https://pyodide-cdn2.iodide.io/v0.15.0/full/pyplot.js
Loading python-dateutil from https://pyodide-cdn2.iodide.io/v0.15.0/full/python-dateutil.js
Loading peewee from https://pyodide-cdn2.iodide.io/v0.15.0/full/peewee.js
Loading numpy from https://pyodide-cdn2.iodide.io/v0.15.0/full/numpy.js
Loading kwi solver from https://pyodide-cdn2.iodide.io/v0.15.0/full/kwi solver.js
Loading cycler from https://pyodide-cdn2.iodide.io/v0.15.0/full/cycler.js
Loading pandas from https://pyodide-cdn2.iodide.io/v0.15.0/full/pandas.js
Loaded matplotlib, pyplot, python-dateutil, peewee, numpy, kwi solver, cycler, pandas
https://raw.githubusercontent.com/Blaschou/LassoRegression/master/wget HEIGHT.CSV

18

Рисунок Е.18 – Слайд №18

Висновки

Під час виконання магістерської роботи, було розроблено програмне забезпечення для інтерактивної платформи для інтелектуального аналізу даних.

В ході виконання роботи було вирішено наступні задачі:

- ознайомлення і аналіз предметної області;
- розгляд методів виконання користувацьких Python скриптів;
- розгляд мов програмування Python та JavaScript;
- розгляд середовища розробки PyCharm;
- розроблення інтерактивної платформи для інтелектуального аналізу даних;

19

Рисунок Е.19 – Слайд №19

Дякую за увагу!

20

Рисунок Е.20 – Слайд №20