Міністерство освіти і науки України

Харківський національний університет імені В. Н. Каразіна

Факультет комп'ютерних наук

Кафедра теоретичної та прикладної системотехніки

«Затверджую»
Зав. кафедри теоретичної та
прикладної системотехніки
_____ д.т.н., проф. С. І. Шматков
«___» грудня 2022 р.

Пояснювальна записка

до кваліфікаційної роботи
магістра

на тему: «**КОМП'ЮТЕРНА МОДЕЛЬ ОБРОБКИ ВЕЛИКИХ ОБСЯГІВ ДАНИХ»**

Захищено на засіданні
Атестаційної комісії № 44
протокол № __ від __.12.2022 р.
Оцінка _____ / _____
Голова Атестаційної комісії
_____**МІНУХІН С. В.**

Виконав:

студент 6 курсу, групи КІ– 61
за спеціальністю 121 – Комп'ютерна інженерія.
Галузь знань: 12 – Інформаційні технології
Каплун Владислав Миколайович___

Керівник:

д.т.н., с.н.с., професор кафедри теоретичної та прикладної системотехніки
ТОЛСТОЛУЗЬКА Олена Геннадіївна

Рецензент:

_____

Харків – 2022

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ 4
ВСТУП 5
РОЗДІЛ 1. АНАЛІЗ СУЧАСНИХ МЕТОДІВ РОЗРОБКИ WEB-САЙТІВ ТА ОБРОБКИ ВЕЛИКИХ ОБСЯГІВ ДАНИХ 6
1.1. Розробка веб-додатків 6
1.1.1. Front-end та Back-end 6
1.1.2. RESTful API 9
1.1.3. Односторінкова програма 11
1.2. Full Stack JavaScript 13
1.2.1. JavaScript 13
1.2.2. Поява Full Stack JavaScript *доповить* 15
1.3. MEAN Stack 17
1.3.1. Node.js 17
1.3.2. Express 22
1.3.3. MongoDB 25
1.4. Обробка даних за допомогою Python 29
1.4.1. NumPy і Pandas 29
1.4.2. Використання NumPy і Pandas в Python 31
1.4.3. Бібліотека Pandas 35
1.4.4. Бібліотека matplotlib 37
1.5. Постановка задачі 38
Висновки за розділом 1 39
РОЗДІЛ 2. РОЗРОБКА КОМП'ЮТЕРНОЇ МОДЕЛІ WEB-ДОДАТКУ ОБРОБКИ ВЕЛИКИХ ОБСЯГІВ ДАНИХ 40
2.1. Розробка структури сайту 40
2.2. Робота з JavaScript 42
2.2.1. Налаштування програмної оболонки Node.js 42
2.2.2. Інсталювання Express 43
2.2.3. Налагодження програми Express. 45
2.3. Робота Python & EEL 47

2.3.1. Розуміння бібліотеки Python EEl.....47
2.3.2. Реалізація бібліотеки Eel.....48
2.4. Обробка даних на Python.....52
2.4.1. Правила візуалізації даних.....52
2.4.2. Візуалізація з Matplotlib.....53
Висновки за розділом 2.....57
РОЗДІЛ 3. РЕКОМЕНДАЦІЇ ПО ВИКОРИСТАННЮ КОМП'ЮТЕРНОЇ МОДЕЛІ WEB-СЕРВІСУ ОРГАНІЗАЦІЇ РОБОТИ ПСИХОЛОГА.....59
3.1. Сторінки, що доступні всім користувачам. 59
3.2. Сторінки фахівця. 61
Висновки за розділом 3.....62
ВИСНОВКИ.....63
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....64
Додаток А.....65

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

NET — домен верхнього рівня

Web — World Wide Web

GUI — Graphical User Interface

СУБД — системою управління базами даних

MVC — Model View Controller

UML — Unified Modeling Language

URL Uniform Resource Locator

XSRF — Cross-Site Request Forgery

DBMS — Database Management System

MongoDB — Mongo Data Base

API — Application Programming Interface

JS — Javascript

CLI — Command Line Interpreter

CSS — Cascading Style Sheets

DOM — Document Object Model

HTML — HyperText Markup Language

FTP — File Transfer Protocol

ВСТУП

Програмісти різних напрямків надають послуги розробки веб-сервісів, які створюють можливість фахівцям з використанням необхідного інструментарію взаємодіяти онлайн з користувачами.

Тож, **метою роботи**

є розробка комп'ютерної модель обробки велиеих обсягів даних.

Для досягнення поставленої мети слід вирішити наступні завдання:

• розглянути сучасні технології проектування Web - сайтів;

• визначити комп'ютерну модель Web - сайту;

• розробити на основі Node JS Web - сайт;

• розробити інструкції користувача;

• заповнити матеріалом.

Об'єктом дослідження – процес розробки онлайн сервісу, що надає можливість користувачам з пристроїв які підключені но мережі проходити анкетування, психофізичне тестування та додаткову можливість фахівцям психологам проводити аналіз отриманих даних.

Предмет дослідження – методи та технології розробки веб-сайту з обробкою великих обсягів даних.

РОЗДІЛ 1. АНАЛІЗ СУЧАСНИХ МЕТОДІВ РОЗРОБКИ WEB-САЙТІВ ТА ОБРОБКИ ВЕЛИКИХ ОБСЯГІВ ДАНИХ

1.1. Розробка веб-додатків

1.1.1. Front-end та Back-end

Розробка веб-додатків — це поєднання Front-end та Back-end. Інтерфейсна веб-розробка, також відома як розробка на стороні клієнта, передбачає практику створення графічного інтерфейсу користувача (GUI) для клієнтів (користувачів), щоб користувачі могли взаємодіяти з програмою. Він передбачає використання основних веб-технологій і інструментів, таких як HTML, CSS і JavaScript. HTML — це мова розмітки, яка забезпечує структуру веб-сторінки. Він визначає, як буде виглядати веб-сторінка, тому її можна вважати скелетом будь-якої веб-програми. CSS, з іншого боку, є мовою таблиць стилів, яка забезпечує стиль і візуальні покращення документів, написаних у HTML. JavaScript є найдосконалішою мовою серед цих технологій. Він виконує маніпуляції HTML DOM (об'єктна модель документа), щоб надати користувачам динамічний інтерфейс. Крім того, він забезпечує інтерактивний інтерфейс для користувачів, створюючи спливаючі повідомлення, перевіряючи введені форми та змінюючи макет на основі подій, таких як введення користувача або клацання мишею. Усі ці технології контролюються браузером, щоб забезпечити зовнішній веб-інтерфейс. Внутрішня веб-розробка, також відома як розробка на стороні сервера, передбачає розробку комп’ютерних програм і баз даних для обслуговування клієнта. Веб-додаток у перші дні не потребував інтерфейсу, але функціонуючої серверної програми було достатньо, щоб вважати її веб-програмою. Відтоді в цій сфері було зроблено кілька змін. Сучасні складні веб-програми не можуть працювати без інтерфейсних і внутрішніх служб. Внутрішні технології зазвичай складаються з мов програмування, таких як PHP, Ruby, Python, Java, Node.js та різних фреймворків. Розробка веб-додатків охоплює всі веб-технології, пов’язані

з інтерфейсом і бек-ендом. Крім того, веб-додаток має забезпечувати зв'язок між клієнтом і сервером з використанням різних протоколів зв'язку. Протоколи - це набір правил для обміну повідомленнями в мережі зв'язку. Протоколи відрізняються залежно від завдань і рівнів. HTTP (протокол передачі гіпертексту), TCP/IP (протокол керування передачею/ Інтернет-протокол), FTP (протокол передачі файлів), SMTP (простий протокол передавання пошти), SOAP (простий протокол доступу до об’єктів) і REST (передавання стану представлення) є деякими типовими прикладами протоколів, які використовуються у веб-додатках. Веб-додаток у своїй найелементарнішій формі надсилає запит HTTP на сервер для встановлення з’єднання, а сервер надсилає відповідь HTTP відповідь клієнту. Типовий приклад спілкування у веб-додатку показано на рисунку 1.

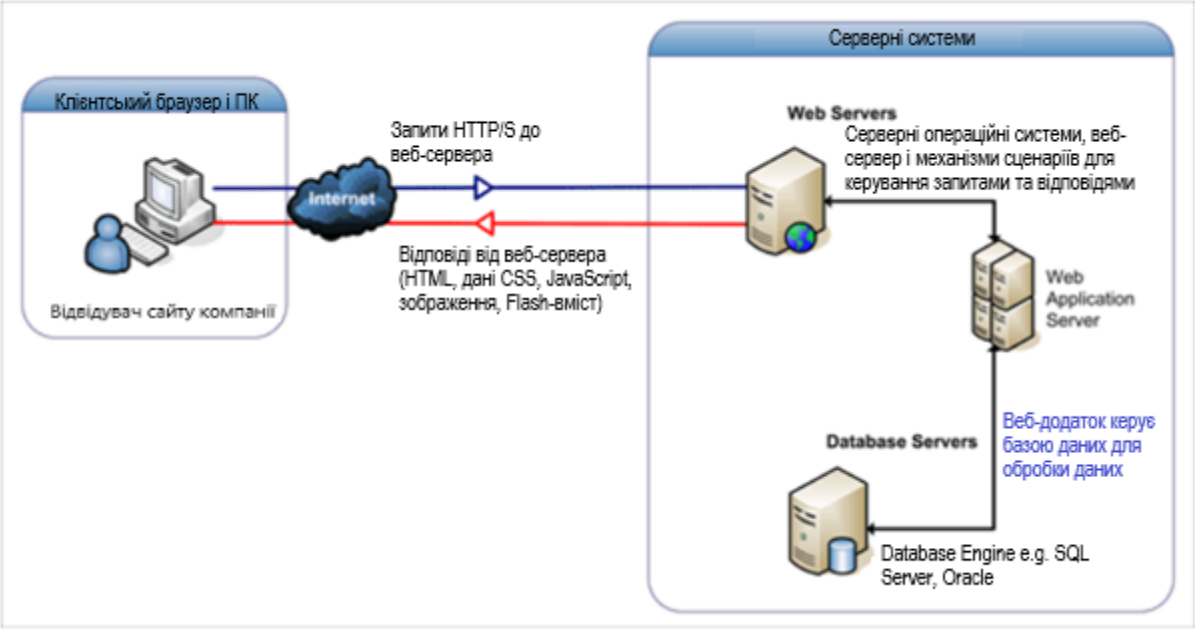


Рисунок 1. Модель веб-додатку.

На малюнку 1 показано взаємодію між трьома рівнями моделі веб-додатку. Перший рівень — це веб-браузер на стороні клієнта, другий — генератор

динамічного вмісту на стороні сервера, а третій — сервер бази даних. Користувач надсилає початковий запит за допомогою протоколу HTTP через браузер через Інтернет на сервер. Потім веб-сервер обробляє запит, звертаючись до сервера бази даних і отримуючи запитані дані. Потім веб-сервер надсилає відповідь користувачеві через Інтернет через браузер. Відповідь зазвичай містить дані, які запитує користувач.

Побудова хорошої системи управління базами даних (СУБД) для зберігання інформації є важливою частиною розробки веб-додатків. СУБД дозволяє користувачам створювати, зберігати, оновлювати та запитувати (пошук) дані у веб-додатку. Існує два типи баз даних: реляційні та нереляційні. Реляційні бази даних, також відомі як бази даних SQL, є традиційними базами даних, які зберігають дані в таблицях у формі рядків і стовпців. Таблиці підтримують певний зв’язок одна з одною, що дало їй назву «реляційна база даних». База даних Oracle, MySQL і SQL Server є популярними реляційними базами даних.

Нереляційні бази даних, також відомі як бази даних NoSQL, зберігають дані у формах, відмінних від таблиць. Бази даних NoSQL зберігають дані в парі ключ-значення, графіку, документі, стовпці або мультимоделі залежно від бази даних, вибраної для розробки програми. MongoDB, HBase, Cassandra, Redis і Riak є одними з прикладів популярних баз даних NoSQL. Реляційні бази даних, як правило, використовуються в корпоративних програмах, які можуть обробляти великі пов’язані дані, тоді як нереляційні бази даних, які є гнучкими та масштабованими за своєю природою, використовуються в керованих даними веб-додатках реального часу зі швидким збільшенням даних.

На додаток до хорошої бази даних, хороша структура для розробки веб-додатків також стала необхідністю в наш час. Використання веб-фреймворків у розробці настільки складних веб-додатків є більш практичним, ніж використання рідних мов програмування. Веб-фреймворки, також відомі як фреймворки веб-

додатків, призначені для полегшення завдання розробки веб-додатків, надаючи бібліотеки доступу до бази даних, шаблони, керування сеансами та можливість повторного використання коду. Фреймворки JavaScript, такі як AngularJS, ReactJS, Backbone.js, Ember.js і Knockout.js, є найпопулярнішими фреймворками у інтерфейсній розробці. У той час як фреймворки PHP (Laravel, Sym fony, CakePHP), фреймворки JavaScript (Node.js, Meteor, Express), Rails, Pyramid, ASP.NET, Java EE, Spring і Django є найпопулярнішими фреймворками в back-end розробці.

1.1.2. RESTful API

REST — це архітектурний стиль, який використовується у веб-розробці для створення веб-сервісів. REST лише визначає принципи, на яких розробляється веб-служба для зв’язку між клієнтом і сервером. Це не набір правил (протоколів) для створення веб-сервісів. Будь-які веб-служби або API (інтерфейси прикладного програмування), розроблені з використанням архітектури REST, називаються RESTful API або просто REST API. REST забезпечує хорошу продуктивність, масштабованість і надійність у розподіленій обчислювальній системі. Існує кілька обмежень для того, щоб програма стала програмою REST. Тим не менщ, конкретна реалізація REST API повинна дотримуватися щонайменше чотирьох основних принципів проектування:

- Використання методів HTTP: REST API повинні явно дотримуватися методів HTTP. Вони повинні використовувати GET для отримання ресурсу із сервера, POST для створення ресурсу, PUT для зміни чи оновлення ресурсу. ресурсу та DELETE для видалення ресурсу.
- Статичний зв'язок: Зв'язок між клієнтом та сервером має бути статичним, тобто кожен запит від клієнта повинен містити всю

інформацію, необхідну серверу для їх обробки. Сервер не повинен вимагати ніяких даних для обробки запиту.

- Використання структури каталогу, подібної до URI: REST API повинні використовувати URI, які є простими, правильно структурованими і легко розуміються.
    - Відправка даних у XML або JSON:: Дані, що передаються між клієнтом та сервісом-обмінником повинні бути у форматі XML або JSON, щоб дані були правильно структуровані та читані.

Крім того, веб-сервіси REST (API) повинні мати чіткий поділ логіки на стороні клієнта та логіки на стороні сервера. Єдиний інтерфейс розділяє клієнтів і сервери, що дозволяє розробникам працювати над окремою частиною веб-додатку та покращувати одну не торкаючись іншої. Клієнти та посередники повинні мати можливість кешувати відповіді сервера, щоб уникнути повторного використання застарілих даних у відповідь на майбутні запити. Клієнти також можуть припускати пряме з'єднання з сервером. У більшості випадків посередники між клієнтом та сервером обслуговують цикл запит-відповідь. Більше того, у веб-сервісах сервер REST може тимчасово розширити клієнта, передавши йому логіку.

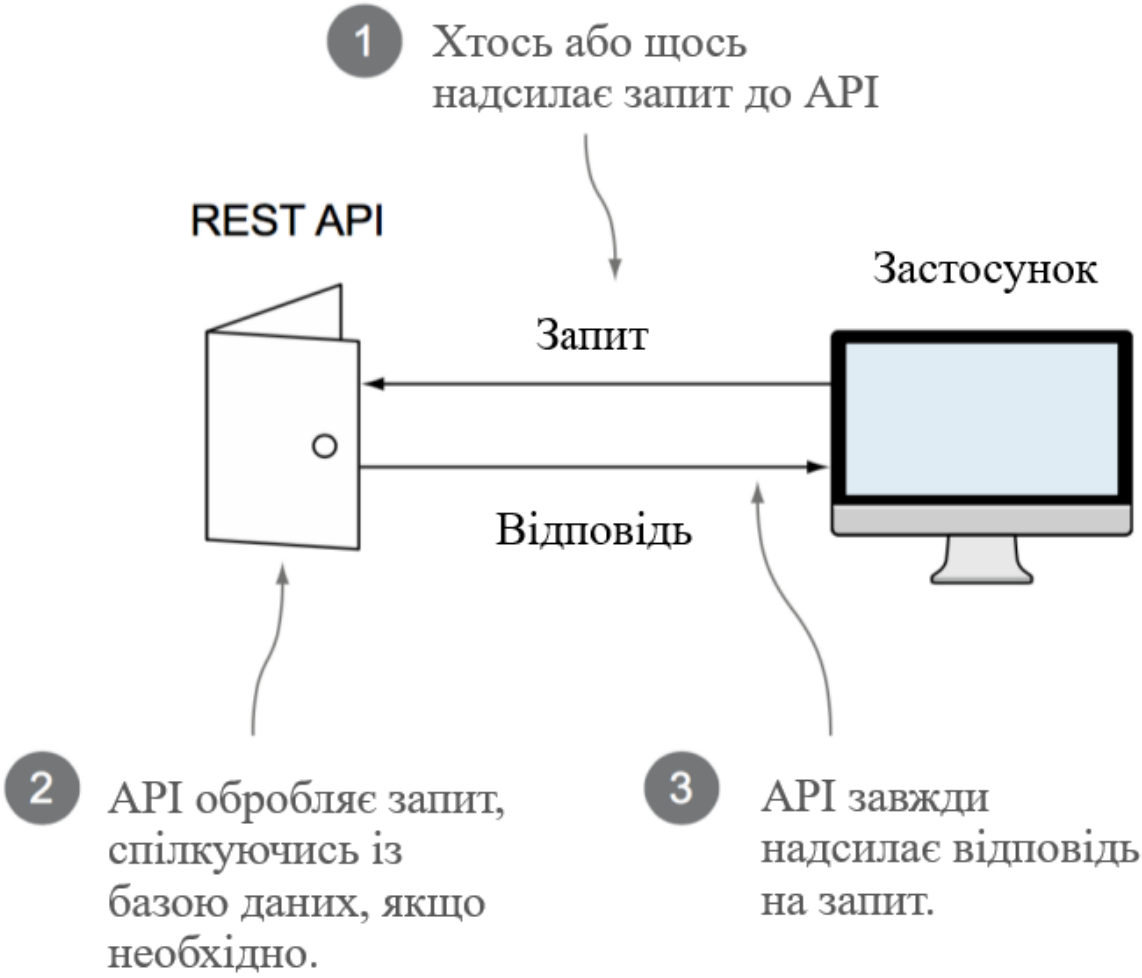


Рисунок 2. REST API

На рисунку 2 показано зв’язок між клієнтською програмою та REST API. Клієнт не знає реалізації сервера і не спілкується безпосередньо з сервером. REST API приймає всі вхідні HTTP-запити від клієнта, обробляє їх і надсилає HTTP-відповіді .

1.1.3. Односторінкова програма

Односторінкова програма (SPA) — це веб-програма, яка розміщується на одній веб-сторінці. На відміну від традиційного повного завантаження сторінки, SPA завантажує всі ресурси, необхідні для навігації по веб-програмі під час

завантаження першої сторінки. Потім він динамічно змінює вміст під час взаємодії користувача з програмою, тому жодного запиту повної сторінки більше ніколи не буде зроблено. Однак URL-адреси оновлюються в адресному рядку браузера хеш-тегом після назви доступних ресурсів.

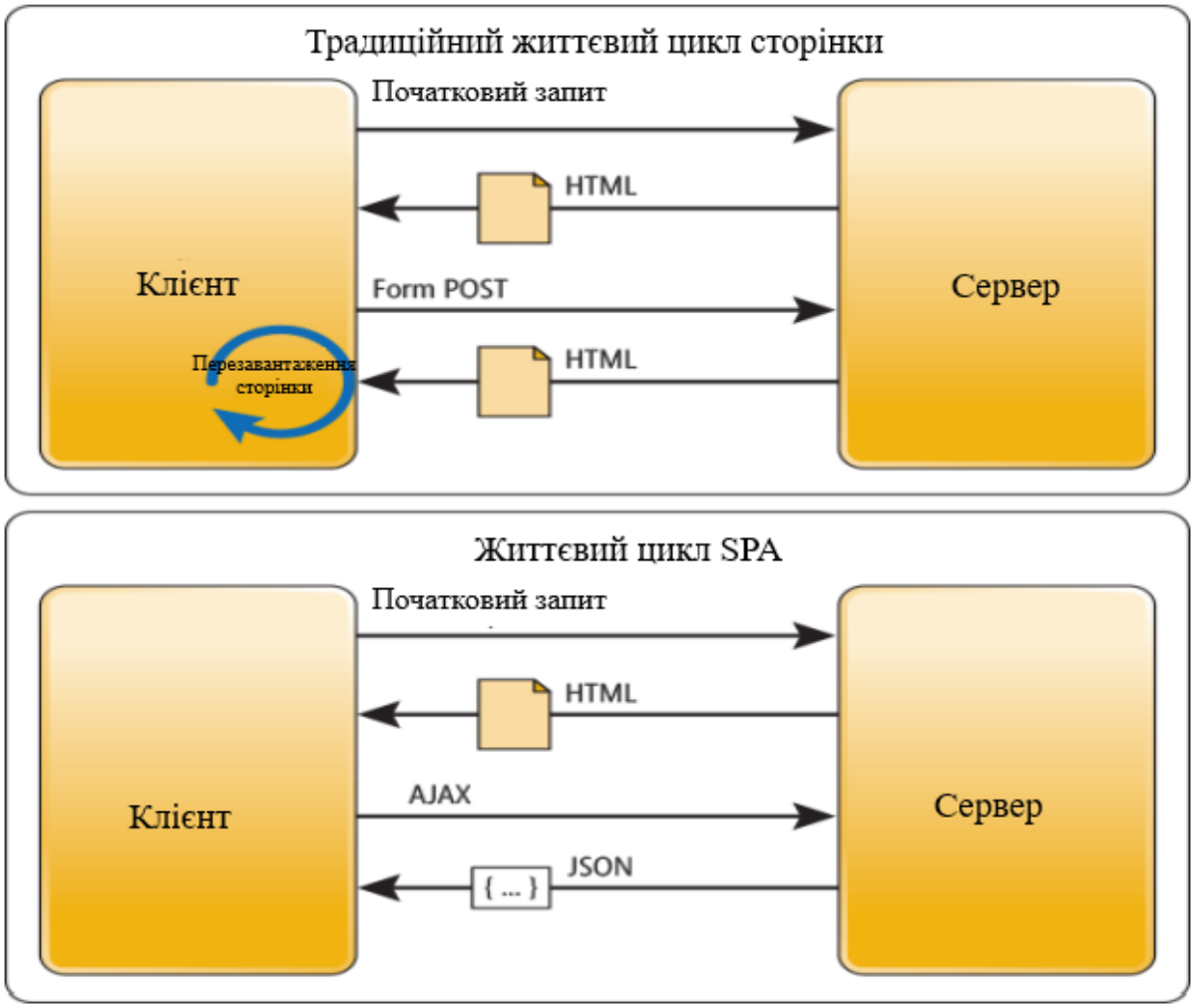


Рисунок 3. Порівняння життєвого циклу традиційної сторінки та життєвого циклу SPA.

Рисунок 3 ілюструє різницю між життєвим циклом традиційної веб-сторінки та веб-сторінки SPA. У традиційних веб-додатках щоразу, коли клієнт надсилає запит серверу, сервер у відповідь відтворює нову сторінку HTML. Усі наступні запити до сервера також обробляються подібним чином і кожного разу,

коли в браузері клієнта завантажується нова сторінка. У додатку SPA після завантаження першої сторінки всі взаємодії між клієнтом і сервером відбуваються за допомогою викликів AJAX (асинхронний JavaScript і XML), які в свою чергу надсилають дані у формі JSON (або XML). Потім браузер використовує дані JSON для динамічного оновлення сторінки без перезавантаження сторінки.

SPA використовують AJAX (для взаємодії з сервером), шаблони HTML, фреймворки MVC і JavaScript для виконання більшості навігаційних робіт на інтерфейсі. Сучасні зовнішні фреймворки JavaScript, такі як AngularJS, Ember.js, React і Meteor.js, спростили завдання створення SPA, забезпечивши багатофункціональне маніпулювання DOM і функції двостороннього зв'язування даних. SPA забезпечують багатий інтерфейс і гнучку взаємодію з користувачем. Крім того, SPA дають користувачам відчуття, ніби вони взаємодіють із настільною програмою.

1.2. Full Stack JavaScript

1.2.1. JavaScript

JavaScript — це кросплатформна, легка, об'єктно-орієнтована мова сценаріїв, розроблена в основному для додавання інтерактивності до сучасних веб-сторінок і веб-додатків. Це один із трьох основних компонентів інтерфейсних технологій, крім HTML (надає структуру веб-сторінці) і CSS (надає стилю веб-сторінці). Він був розроблений американським програмістом Бренданом Айхом у Netscape у 1995 році. Спочатку називався «Mocha» та «LiveScript», Javascript отримав свою нинішню назву в грудні 1995 року з випуском третьої бета-версії.

JavaScript було передано до ECMA (Європейської асоціації виробників комп'ютерів) для стандартизації, щоб інші браузери могли реалізувати його на основі стандартизації. Потім ECMA International опублікувала стандартні

специфікації мови JavaScript під назвою ECMAScript. Поточна специфікація JavaScript базується на ECMAScript 2016, який був випущений у червні 2016 року.

JavaScript надає багато функцій для створення динамічних та інтерактивних інтерфейсів у клієнтських програмах. Він може визначати браузер користувача та ОС (операційну систему) і виконувати операції, пов'язані з платформою. JavaScript, спочатку названий як інтерпретована мова, може виконуватися без попередньої компіляції браузером, щоб він міг виконувати прості обчислення на стороні клієнта. Він часто використовується для перевірки введених користувачем даних у формах і асинхронного надсилання даних за допомогою AJAX. Він також виконує маніпуляції HTML DOM для забезпечення динамічного інтерфейсу. Це об'єктно-орієнтована мова програмування, яка підтримує декілька вбудованих об'єктів із успадкуванням. Таким чином, JavaScript дозволяє розробникам додавати динамічні функції до статичних сторінок HTML, керувати мультимедіа та додавати анімацію. Крім того, JavaScript надає кілька сторонніх API і бібліотек для полегшення завдання створення динамічних веб-сторінок. Однією з найпопулярніших бібліотек JavaScript є JQuery.

JQuery — бібліотека на основі JavaScript, розроблена американським програмістом Джоном Резігом. Усі функції, які можна запрограмувати в традиційному JavaScript, можна виконати в JQuery, але за допомогою набагато простіших методів. Таким чином, функція, яка вимагає кількох рядків коду для написання в JavaScript, може бути легко написана в декілька рядків за допомогою JQuery. Він підтримується майже всіма сучасними браузерами і не вимагає додаткових плагінів або розширень для роботи. Однією з головних особливостей JQuery є маніпулювання DOM. DOM — це деревоподібне структурне представлення елементів на сторінці HTML. Синтаксис JQuery полегшує навігацію між цими елементами, знаходить і маніпулює ними. Наприклад, його

можна використовувати, щоб знайти всі елементи з певною властивістю на веб-сторінці, змінити властивості елементів або змусити їх реагувати на певні події, наприклад клацання мишею. Це завдання важко виконати в чистому JavaScript, але синтаксис JQuery полегшує його.

Наприклад, щоб змінити колір тла основного тексту на HTML-сторінці, потрібно написати таку функцію в JavaScript:

```
function changeBackground(color) {
        document.body.style.background = color;
}
onload="changeBackground('red');"
```

тоді як те саме завдання можна виконати в одному рядку коду за допомогою такої функції JQuery:

```
$('body').css('background', '#ccc');
```

Крім того, JQuery надає простіші функції для створення анімації, обробки різних подій і розробки програм AJAX.

AJAX (асинхронний JavaScript і XML) — це техніка веб-розробки для створення асинхронних веб-додатків. За допомогою AJAX можна здійснювати асинхронний зв’язок між клієнтом і сервером. AJAX використовує об’єкт XMLHttpRequest для зв’язку із сервером і надсилає та отримує інформацію у форматі XML, JSON, HTML або навіть у форматі текстового файлу. XMLHttpRequest — це API, який забезпечує асинхронний зв’язок між клієнтом і сервером без необхідності повного перезавантаження сторінки. Отже, AJAX дозволяє оновлювати певну частину веб-сторінки без оновлення (перезавантаження) сторінки.

1.2.2. Поява Full Stack JavaScript *доповить*

JavaScript на початку свого існування заслужив погану репутацію через низьку продуктивність і несумісність із відомими браузерами того часу. Однак

все почало змінюватися після того, як великі постачальники браузерів вклали час і гроші в покращення мови. JavaScript, нарешті, став фактичним стандартом сценаріїв на стороні клієнта. У той час як JavaScript залишався помітним на стороні клієнта, кілька нових технологій були представлені на стороні сервера, такі як PHP, JAVA, .NET, Ruby та Python. Розробники почали розуміти, що використання двох окремих мов у розробці клієнта та сервера ускладнює завдання веб-програмування. Було зроблено кілька спроб об'єднати обидві сторони шляхом створення клієнтських компонентів на сервері та компіляції їх у JavaScript (наприклад, веб-форми ASP.NET і GWT), але вони зазнали невдачі. Єдиним вирішенням цієї проблеми була реалізація JavaScript на стороні сервера, і був представлений Node.js.

Node.js насправді є основою веб-розробки Full Stack JavaScript. Нарешті він застосував потужність JavaScript на сервері з ідеєю неблокуючої параметри програмування. Node.js став популярним за короткий час завдяки своїм простим у використанні компонентам. Це дозволило розробникам швидко налаштувати сервер і почати створювати програми на його основі. Для полегшення впровадження Node.js почали з'являтися кілька фреймворків, наприклад Express і Connect.js. Експрес став найпомітнішим. Екосистема Node.js продовжувала розширюватися, і було представлено такий менеджер пакетів, як «npm».

Поки Node.js вторгся на сервер, база даних NoSQL почала набирати популярність у сфері баз даних. MongoDB, база даних NoSQL, була представлена з концепцією зберігання даних у двійковому JSON. Завдяки використанню JSON, який є способом зберігання даних на JavaScript, MongoDB стала переважною базою даних для програм Node.js. JavaScript також почав розвиватися на стороні клієнта, і нові фреймворки, такі як AngularJS, Back bone.js і ReactJS, почали використовувати традиційні JavaScript, JQuery та AJAX для створення односторінкових програм. Коли всі відомі інструменти для створення програми Full Stack JavaScript були готові, були представлені додаткові інструменти на

основі JavaScript, щоб покращити процес розробки, такі як Mocha.js і Chai.js для тестування програм, Gulp.js і Grunt. js для автоматизації завдань збірки.

Екосистема Node.js із додаванням AngularJS на інтерфейсі, Express як бек-енд фреймворк і MongoDB як база даних створила новий термін у розробці веб-додатків під назвою MEAN (MongoDB, Express.js , AngularJS, Node.js). Слово MEAN стек використовується для позначення Full Stack JavaScript, але воно все ще є підмножиною терміна Full Stack JavaScript. Node.js, Express і MongoDB є видатними членами Full Stack JavaScript. Однак AngularJS можна замінити на Backbone.js, ReactJS або Ember.js, щоб відповідати вимогам розробників. Тим не менш, термін Full Stack JavaScript зазвичай використовується для позначення стеку MEAN.

1.3. MEAN Stack

1.3.1. Node.js

Node.js — це програмна платформа, яка допомагає створювати асинхронні та керовані подіями мережеві програми. Він містить вбудовані бібліотеки HTTP-серверів, які дозволяють розробникам створювати власний веб-сервер і створювати на його основі високомасштабовані веб-додатки. Механізм виконання JavaScript V8, який використовує Node.js, — це той самий механізм, який використовується в браузері Google Chrome. Механізм V8 узгоджує коди безпосередньо з рідним машинним кодом, залишаючи поза увагою інтерпретатора та процес виконання байт-коду, що дає Node.js величезний приріст продуктивності. Крім двигуна V8, Node.js складається з кількох компонентів, як показано на малюнку 4.

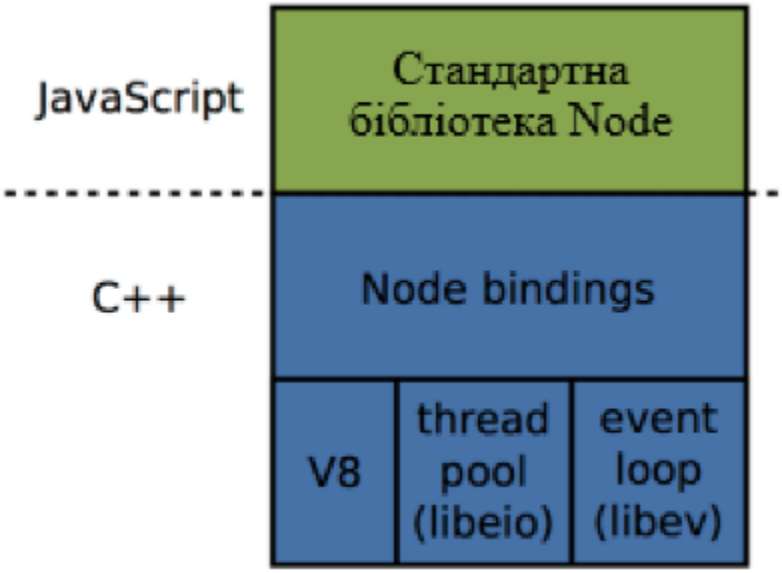


Рисунок 4. Архітектура Node.js

На рисунку 4 показано архітектуру Node.js. Node.js складається зі стандартної бібліотеки Node у верхній частині, тонких прив'язок вузлів C++ у середині та двигуна V8, libeio та libev унизу. Стандартна бібліотека JavaScript Node надає додатку функції операційної системи, тоді як прив'язки C++ надають JavaScript основні API основних елементів. Механізм V8 забезпечує середовище виконання для програми, а libeio обробляє пул потоків, щоб здійснювати асинхронні (неблокуючі) виклики вводу-виводу до libev, циклу подій.

Функція асинхронного неблокуючого вводу-виводу Node.js відіграє важливу роль в управлінні ресурсами та підвищенні продуктивності програм Node.js. На відміну від інших основних серверів, таких як Apache та IIS, Node.js використовує однопотокову неблокуючу операцію введення-виведення. Це означає, що замість запуску кожного сеансу (запиту) в 12 окремих потоках і надання відповідного обсягу оперативної пам'яті для кожного сеансу Node.js використовує один потік для виконання всіх запитів і реалізує цикл подій, щоб уникнути блокування I /O. У багатопоточному сервері зі збільшенням кількості потоків накладні витрати на сервер (планування та використання пам'яті)

збільшуються, що призводить до повільної продуктивності всієї системи. Node.js використовує керований подіями та неблокуючий підхід введення-виведення для обробки всіх запитів до сервера.

Рисунок 5. Модель обробки Node.js

На малюнку 5 Node.js створює подійний цикл із обробниками подій для всіх запитів. Коли відбувається операція введення-виведення, асоційований обробник ставиться в чергу на виконання, а функція зворотного виклику видає подію після завершення операції введення-виведення. Тим часом інші операції введення-виведення продовжують виконуватися поза циклом подій сервера. Таким чином, Node.js виконує операції введення-виведення асинхронно і не блокує виконання сценаріїв, дозволяючи циклу подій відповідати на інші запити. Одне поширене використання функції зворотного виклику, що відображає неблокуючу операцію введення/виведення Node.js, показано в рисунку 6.

```
var fs = require("fs");
fs.readFile('input.txt', function (err, data) {
if (err) return console.error(err);
console.log(data.toString());
});
console.log("Reading file");
```

Рисунок 6. Приклад неблокуючого коду Node.js

Рисунок 6 ілюструє приклад неблокуючого коду Node.js. Програма намагається виконати три операції: прочитати файл «input.txt», надрукувати дані файлу на консолі та надрукувати повідомлення 13 «Читання файлу» на консолі. Функція readFile() виконується першою в порядку появи, але програма не чекає завершення функції читання файлу. Як тільки функція починає читати файл, вона передає керування для негайного виконання наступної інструкції, тому програма друкує «Читання файлу» перед друком даних файлу. Після завершення введення-виведення файлу без будь-яких помилок він викличе функцію зворотного виклику (err, data) і поверне дані файлу як параметри, а потім надрукує дані під повідомленням «Читання файлу» на консолі. Таким чином, операції введення-виведення не блокуються, і всі операції виконуються асинхронно.

Node.js має багату екосистему розробки з кількома сумісними бібліотеками та менеджерами пакунків. Одним із основних менеджерів пакунків, який поставляється в комплекті з Node.js під час встановлення, є npm. Npm запускається з інтерпретатора командного рядка (CLI) і керує всіма залежностями для програм Node.js. Замість того, щоб вручну завантажувати та налаштовувати модулі JavaScript для використання в програмі, npm пропонує простішу альтернативу. Назви модулів і залежностей можна включити з номерами версій у файл «package.json» у структурі папок, а модулі завантажуються за допомогою простої команди «npm install» із CLI.

npm автоматично обробляє весь процес завантаження та зберігає всі названі

модулі в папці «node-modules». Модулі можна оновити, змінивши номер версії в «package.json», і їх можна легко розповсюдити, завантаживши один файл «package.json» на сервер замість завантаження великої папки «node-modules». Після оновлення або завантаження команда «npm install» встановлює певні модулі та залежності. Тому програми Node.js є легкими, гнучкими та доступними для спільного використання.

Node.js фактично є основою стеку MEAN. На її основі працюють усі інші технології. Node.js представляє потужність JavaScript на стороні сервера, дозволяючи розробникам створювати логіку як на стороні сервера, так і на стороні клієнта однією мовою «JavaScript». Однією з головних переваг Node.js перед іншими серверними платформами є вбудований цикл подій. Цикл подій використовується всіма доступними модулями та бібліотеками в Node.js, що робить операції вводу-виводу ефективними. Ця функція Node.js забезпечує швидкість і ефективність усієї системи.

Технологія веб-розробки	Обчислення значення Фібоначчі	Середній запит за секунду[#/c]	Середній час на запит [млс]
Node.js	Fib (10/ 20/ 30)	2491.77/ 1529.4/ 58.85	0.401/ 0.654/ 16.993
Python-Web	Fib (10/ 20/ 30)	633.68/ 209.89/ 2.9	1.578/ 4.764/ 345.307
PHP	Fib (10/ 20/ 30)	2051.22/ 168.8/ 1.78	0.488/ 5.942/ 560.553

Рисунок 7. Тест продуктивності Node.js, Python-Web і PHP

Рисунок 7 ілюструє результати продуктивності Node.js у порівнянні з Python Web і PHP при обчисленні Фібоначчі (10/20/30). Дослідження було проведено дослідниками з Пекінського університету, Китай, і результати були опубліковані в технічному звіті IEEE. Чітко видно, що Node.js краще обробляє запити та виконує обчислення, ніж дві інші технології. При обчисленні невеликого Fibo Nacci 10 Node.js і PHP обидва працюють добре, тоді як Python-Web відстає. Node.js обробляє близько 2500 запитів на секунду, займаючи 0,401

мс для обробки кожного запиту. PHP поступається лише кількома показниками, але розрив у їхній продуктивності різко збільшується, коли обчислення стають складнішими. Коли завдання досягає обчислення Фібоначчі 30, і PHP, і Python-Web працюють погано, майже обробляючи 2 або 3 запити на секунду, і обробка кожного запиту займає 345 і 560 мс відповідно. На відміну від них, Node.js все ще підтримує якість, обробляючи запити в 20 разів більше, ніж PHP, і займає відносно короткий час, близько 17 мс.

Незважаючи на те, що Node.js є ідеальним вибором для масштабованих додатків, керованих даними, інтенсивного введення-виведення, він не є ідеальним рішенням для всіх програм. Він використовує JavaScript, тому він більш ефективний при використанні з іншими технологіями на основі JavaScript. Використання одного потоку для обробки всіх запитів є ідеальним у деяких випадках, але це не є хорошою функцією для інтенсивних програм обчислення даних. Крім того, Node.js використовує тісний зв’язок між веб-сервером і веб-додатком, тому всі класи залежать один від одного. Таку тісно пов’язану систему важко підтримувати, оскільки проблема в одній області призводить до збою всієї системи. Node.js також погано працює з реляційними базами даних. Усі ці фактори необхідно враховувати перед тим, як вибрати Node.js як платформу для розробки будь-якої програми. Дослідження показали, що Node.js ідеально підходить для веб-додатків, керованих даними в реальному часі, у співпраці з технологією push і веб-сокетами.

1.3.2. Express

Express — це вузловий модуль, який забезпечує мінімальну гнучку структуру для веб-додатків Node.js. Він працює поверх основних модулів вузла, не приховуючи жодних функцій Node.js. Крім того, він надає надійні та чисті функції для додавання до модулів вузла, тому розробка програми Node.js за допомогою Express набагато легша, ніж використання власних модулів вузла.

[14.] Завдяки простоті Express його прийняли такі великі компанії, як MySpace, Paypal, Apiary.io, Persona та Ghost. Використання фреймворку Express поверх Node.js допомагає підтримувати ясність коду. Це також полегшує інтеграцію модулів і забезпечує структуру рішення для програм. Express встановлюється за допомогою менеджера пакунків npm за допомогою команди «npm install ex press»

Сервер Express складається з трьох будівельних блоків: маршрутизатора, маршрутів і проміжного програмного забезпечення. Основні функції веб-сервера залежать від його відмінних методів маршрутизації. Під час спілкування клієнт-сервер клієнт запитує деякі ресурси у сервера, сервер знаходить ресурси та відповідає, надсилаючи ресурси клієнту. Це основна функція веб-сервера, і вона вимагає відмінних методів маршрутизації для обслуговування запиту. Express робить цю виснажливу роботу справді легкою, дозволяючи розробникам створювати маршрути з простою структурою. Маршрут у Express — це комбінація дієслова HTTP та шляху. Дієслово HTTP зазвичай є одним із чотирьох методів HTTP: GET, POST, PUT і DELETE, а шлях — це розташування ресурсу (URI). Базовий маршрут у Express створюється так:

app.METHOD (PATH, HANDLER), де:

• app — екземпляр express;

• METHOD — це метод запиту HTTP;

• PATH — шлях маршруту (URI);

• HANDLER — це функція, яка виконується, коли маршрут збігається.

Проміжне програмне забезпечення в Express — це функції, які мають функцію шаблону (req, res, next). Req — це вхідний запит від клієнта, res — відповідь від сервера, а далі — функція зворотного виклику. Таким чином, функції проміжного програмного забезпечення виконують будь-який код всередині нього, обробляють об’єкти запиту та відповіді, завершують цикл запит-відповідь і викликають наступну функцію проміжного програмного забезпечення. Поточна функція проміжного програмного забезпечення повинна

завжди викликати наступну функцію проміжного програмного забезпечення, навіть у разі неповного циклу запит-відповідь, щоб уникнути зависання запиту. Простий веб-сервер Express, що містить усі три будівельні блоки, показаний на рисунку 8.

```
var express = require('express');
var app = express();



app.get('/', function(req, res, next) {
  next();
})

app.listen(3000);
```

Рисунок 8. Веб-сервер Express

На рисунку 8 показано три будівельні блоки програми Express: маршрутизатор, маршрут і проміжне програмне забезпечення. У перших двох рядках використовується метод Node.js require() для завантаження експрес-модуля в програму шляхом створення об’єкта програми. Третій рядок — це простий маршрутизатор із маршрутом до розташування «/» та функцією проміжного програмного забезпечення. Програма прослуховує порт 3000 для будь-якого запиту.

Express також надає простий інструмент генератора програм для надання

структури нашій програмі Node.js. Його можна встановити з CLI, виконавши команду «npm install ex press-generator». Він також надає параметри для створення механізму шаблонів для написання кодів HTML. Генератор програм за замовчуванням створює нефритові шаблони для представлень, але він також підтримує Handlebars, EJS, Pug і Moustache. Структура додатка, створена генератором Express, має окремий каталог для маршрутів, переглядів і файлів загальнодоступного відтворення. Однак структура додатка — це лише один із багатьох способів структурування програми Express. Його можна легко модифікувати під час процесу розробки програми, щоб відповідати вимогам програми.

1.3.3. MongoDB

MongoDB — це нереляційна документна база даних з відкритим кодом. Немає необхідності створення об’єктно-реляційного відображення (ORM) для швидкої розробки додатків. На відміну від реляційних баз даних, вона не містить стовпців і рядків. Однак концепція рядків все ще існує в MongoDB, але вона називається документом. Документ визначає як сам себе, так і дані, які він містить. Документ — це набір полів, і він може містити складні дані, як-от списки, масиви або весь документ. Кожен документ містить поле ID, яке можна використовувати як первинний ключ для операції запиту. Набір документів називається колекцією, а MongoDB містить набір колекцій. Формат, у якому Mon goDB зберігає дані, називається BSON, що означає бінарний JSON. Оскільки JSON — це спосіб зберігання даних JavaScript, MongoDB ідеально працює з програмами, створеними за допомогою стеку JavaScript. Основний приклад документа MongoDB проілюстровано в рисунку 9.

```
{
  "firstName": "Homer",
  "lastName": "Simpsons",
  _id: ObjectId("52279effc62ca8b0c1000007")
}
```

Рисунок 9. Приклад документа MongoDB

Рисунок 9 ілюструє фрагмент коду з колекції MongoDB. У документі зберігаються ім'я та прізвище клієнта. На відміну від традиційних реляційних баз даних, MongoDB не містить набір даних, що відповідає набору стовпців, натомість він використовує концепцію пари ім’я-значення для зберігання даних. _id — це унікальний ідентифікатор (первинний ключ) для цього набору даних (документа). Існують деякі варіації в термінах іменування реляційної бази даних і MongoDB, проілюстрованих на рисунку 10.

MySQL	MongoDB
Database	Database
Table	Collection
Index	Index
Row	BSON Document
Column	BSON Field
Join	Embedded documents and linking
Primary key	Primary key
Group by	Aggregation

Рисунок 10. Порівняння термінів MySQL і MongoDB

Як показано на рисунку 10, у MongoDB деякі терміни MySQL, такі як таблиця, називаються колекція, а рядок називається документом BSON. Замість операції Join MongoDB вбудовує піддокументи в основний документ і надає посилання на піддокументи. Mon goDB, як і MySQL, використовує унікальний ідентифікатор (первинний ключ) для кожного документа, щоб було легко

запитувати та знаходити дані. MongoDB підтримує операції вставки, запиту, оновлення та видалення, як і будь-які інші бази даних. Однією з особливостей, яка виділяє MongoDB серед традиційних баз даних, є включення динамічної схеми. Колекції в MongoDB мають різні схеми, і документи в одній колекції можуть мати скільки завгодно різних схем і форм. Ця функція дозволяє розробникам почати зберігати дані в базі даних, враховуючи структурний дизайн бази даних. Ключі та значення документів можна змінювати й оновлювати за потреби, оскільки немає заздалегідь визначеного правила, яке б регулювало перевірку типу даних.

Іншими важливими функціями MongoDB є автоматичне шардування та реплікація. Оскільки дані зберігаються в документі, їх можна зберігати в кількох місцях. Зі збільшенням розміру баз даних одна машина може бути не в змозі зберігати дані та виконувати операції читання і запису. MongoDB вирішує цю проблему, дозволяючи горизонтальне масштабування, тобто дані розподіляються між кількома серверами, і всі сервери можуть працювати разом, щоб підтримувати зростання даних і забезпечувати ефективні операції читання-запису. Подібним чином дані можна копіювати на інший сервер даних, щоб дані завжди були доступними на випадок збою одного сервера. Це дозволяє створювати високомасштабовані та ефективні сервери даних у порівнянні з реляційними базами даних, такими як бази даних MySQL і Oracle.

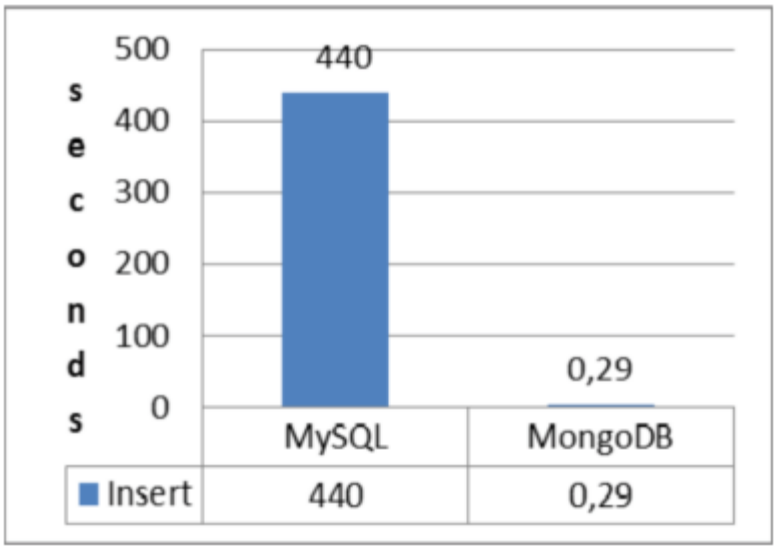


Рисунок 11. Вставка MySQL проти MongoDB

На рисунку 11 показано час, який витрачено MySQL і MongoDB на виконання сценарію для вставки даних 10 000 користувачів. MySQL, яка використовує традиційний підхід до бази даних, займає значно більше часу, ніж MongoDB. Це показує, що MySQL знадобилося 440 секунд, щоб вставити 10 000 користувачів, тоді як MongoDB знадобилося лише 0,29 секунди, щоб виконати те саме завдання. Це доводить, що MongoDB чудово виконує великі операції читання-запису.

Хоча MongoDB добре справляється з більшістю завдань, він має свої недоліки. Mon goDB не може приймати кілька операцій як одну транзакцію. Якщо будь-яка операція в одній транзакції зазнає невдачі, це призведе до збою всієї операції. Він також не може виконувати операції об’єднання, як база даних MySQL, тому він не є гарним вибором у програмі, де існує кілька зв’язків між даними. Дані потрібно шукати та оновлювати в кількох документах одночасно, і всі операції мають бути пов’язані в одній транзакції, щоб забезпечити оновлення даних у всіх колекціях. Транзакційні бази даних працюють у таких випадках краще, ніж нетранзакційні бази даних, такі як MongoDB. Незважаючи на те, що

гнучкість MongoDB, що дозволяє використовувати будь-які дані, у деяких випадках хороша, більшості програм все одно потрібна певна структура даних для належної роботи. Щоб вирішити цю проблему, компанія, що стоїть за MongoDB, створила Mongoose.

Mongoose — це моделювання даних MongoDB для Node.js. Він був створений, щоб вирішити проблему написання складної перевірки даних, кастингу та бізнес-логіки в MongoDB. Він надає програмі просту, елегантну функцію моделювання даних. Використовуючи mongoose, можна визначити, які дані можуть бути в документі та які дані повинні бути в документі. У технічному плані він забезпечує правила перевірки даних, функції побудови запитів і бізнес-логіку до даних програми. Крім того, він надає цілий набір нових функцій для використання поверх MongoDB. Він може керувати підключеннями до бази даних MongoDB, а також читати, записувати та зберігати дані. Він також дозволяє зберігати лише дійсні дані в MongoDB, надаючи правила перевірки на рівні схеми.

1.4. Обробка даних за допомогою Python

1.4.1. NumPy і Pandas

Python все частіше використовується як наукова мова. Матричні та векторні маніпуляції надзвичайно важливі для наукових обчислень. І NumPy, і Pandas стали важливими бібліотеками для будь-яких наукових обчислень, включаючи машинне навчання, у Python завдяки їх інтуїтивно зрозумілому синтаксису та можливостям високопродуктивних матричних обчислень.

Далі надамо огляд загальних функцій NumPy і Pandas. Побачимо схожість цих бібліотек з існуючими наборами інструментів у R і MATLAB. Ця подібність і додаткова гнучкість останнім часом призвели до широкого визнання python у

науковому співтоваристві. Теми, які розглядаються:

- Огляд NumPy
- Огляд Pandas
- Використання Matplotlib

NumPy означає «числовий Python». Це модуль Python з відкритим кодом, який забезпечує швидкі математичні обчислення з масивами та матрицями. Оскільки масиви та матриці є важливою частиною екосистеми машинного навчання, NumPy разом із модулями машинного навчання, такими як Scikit-learn, Pandas, Matplotlib, TensorFlow тощо, доповнюють екосистему машинного навчання Python.

NumPy надає основні багатовимірні обчислювальні функції, орієнтовані на масиви, призначені для математичних функцій високого рівня та наукових обчислень.

Основним об’єктом NumPy є однорідний багатовимірний масив. Це таблиця з однотипними елементами, тобто цілими чи рядковими чи символами (однорідними), як правило, цілими. У NumPy розміри називаються осями. Число осей називається рангом.

Є кілька способів створити масив у NumPy, наприклад np.array, np.zeros, no.ones тощо. Кожен із них забезпечує певну гнучкість.

Ось деякі з важливих атрибутів об’єкта NumPy:

1. Ndim: відображає розмірність масиву;
2. Shape: повертає кортеж цілих чисел, що вказує розмір масиву;
3. Size: повертає загальну кількість елементів у масиві NumPy;
4. Dtype : повертає тип елементів у масиві, тобто int64, символ;
5. Itemsize: повертає розмір кожного елемента в байтах;
6. Reshape : змінює форму масиву NumPy.

Доступ до елементів масиву NumPy можна отримати за допомогою індексації. Нижче наведено кілька корисних прикладів:

- A[2:5] надрукує елементи від 2 до 4. Індекс у масивах NumPy починається з 0;
- A[2::2] надрукує елементи 2, щоб завершити пропуск 2 елементів;
- A[::-1] надрукує масив у зворотному порядку;
- A[1:] друкуватиме від рядка 1 до кінця.

Сесія детально розглядає ці та деякі важливі атрибути об’єкта масиву NumPy.

1.4.2. Використання NumPy і Pandas в Python

У Python вектор можна представити різними способами, найпростішим є звичайний список чисел Python. Оскільки обробітка даних потребує багатьох наукових обчислень, набагато краще використовувати ndarray NumPy, який забезпечує багато зручних і оптимізованих реалізацій основних математичних операцій над векторами. Векторизовані операції виконуються швидше, ніж операції обробки матриць, які виконуються за допомогою циклів у Python. Наприклад, щоб виконати множення матриці 100 * 100, векторні операції за допомогою NumPy виконуються на два порядки швидше, ніж виконання за допомогою циклів.

Ось чим масиви NumPy відрізняються від звичайних масивів Python:

♦ Якщо ви призначаєте єдине значення фрагменту ndarray, воно копіюється на весь фрагмент

Масив NumPy

>>> a = np.array([1, 2, 5, 7, 8]) >>> a[1:3] = -1 >>> a array([1, -1, -1, 7, 8])

Звичайний масив Python

>>> b = [1, 2, 5, 7, 8] >>> b[1:3] = -1 TypeError: can only assign an iterable

Отже, простіше призначати значення фрагменту масиву в масиві NumPy порівняно зі звичайним масивом, де це, можливо, доведеться робити за допомогою циклів.

♦ Зрізи ndarray фактично є переглядами в тому самому буфері даних. Якщо ви зміните його, це також змінить оригінальний ndarray.

Фрагмент масиву NumPy

>>> a = np.array([1, 2, 5, 7, 8]) >>> a_slice = a[1:5] >>> a_slice[1] = 1000 >>> a array([1, 2, 1000, 7, 8])

Звичайний фрагмент масиву python

>>> a=[1,2,5,7,8] >>> b=a[1:5] >>> b[1]=3 >>> print(a) >>> print(b) [1, 2, 5, 7, 8] [2, 3, 7, 8]

Якщо нам потрібна копія масиву NumPy, нам потрібно використовувати метод копіювання як another_slice = another_slice = a[2:6].copy(). Якщо ми змінюємо another_slice, а залишається незмінним.

♦ Спосіб доступу до багатовимірних масивів за допомогою NumPy відрізняється від способу доступу до них у звичайних масивах python. Загальний формат багатовимірних масивів NumPy:

Масив [індекс початку_ряду:індекс_кінця_ряду, індекс_початку_стовпця: індекс_кінця_стовпця].

Доступ до масивів NumPy також можна отримати за допомогою логічного індексування. Наприклад,

>>> a = np.arange(12).reshape(3, 4) array([[0, 1, 2, 3], [4, 5, 6, 7], [8, 9, 10, 11]]) >>> rows_on = np.array([True, False, True]) >>> a[rows_on , :] # Рядки 0 і 2,

усі стовпці array([[0, 1, 2, 3], [8, 9, 10, 11]])

Масиви NumPy здатні виконувати всі основні операції, такі як додавання, віднімання, поелементний добуток, матричний скалярний добуток, поелементне ділення, поелементний модуль, поелементний показник степеня та умовні операції.

Важливою особливістю масивів NumPy є трансляція.

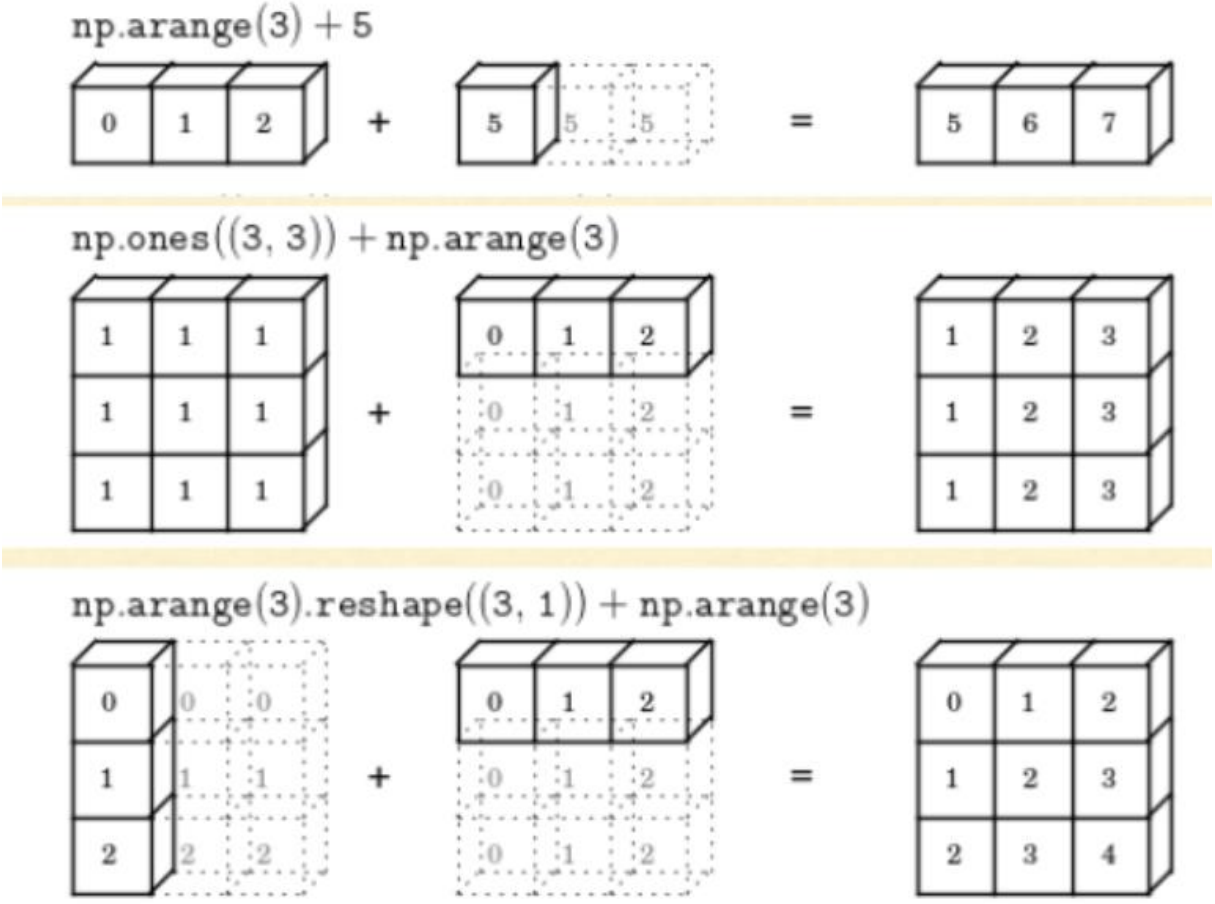


Рисунок 12. Правила трансляції.

Загалом, коли NumPy очікує масиви однакової форми, але виявляє, що це не так, він застосовує так звані правила трансляції.

Загалом, є 2 правила мовлення, які слід пам’ятати:

- Для масивів, які мають різний ранг, 1 додаватиметься до менших масивів ранжирування, доки їхні ранги не збігаються. Наприклад, при додаванні масивів A і B розмірів (3,3) і (,3) [ранг 2 і ранг 1], 1 буде додано до розмірності масиву B, щоб зробити його (1,3) [ранг= 2]. Два набори сумісні, якщо їхні розміри рівні або один із них дорівнює 1.
- Якщо один із порівнюваних розмірів є одним, використовується інший. Іншими словами, розміри з розміром 1 розтягуються або «копіюються», щоб відповідати іншим. Наприклад, після додавання двовимірного масиву A форми (3,3) до 2D масиву B форми (1, 3). NumPy застосує наведене вище правило трансляції. Він має розтягнути масив B і повторити перший рядок 3 рази, щоб створити масив B розмірів 3 x 3 і виконати операцію.

Num Py надає такі базові математичні та статистичні функції, як mean, mın, max, sum, prod, std, var, summatıon across dıfferent axes, тощо.

Особлива цікава функція NumPy — це розв'язування системи лінійних рівнянь. У NumPy є функція для вирішення лінійних рівнянь. Наприклад:

```
>>> coeffs  = np.array([[2, 6], [5, 3]])
>>> depvars = np.array([6, -9])
>>> solution = linalg.solve(coeffs, depvars)
>>> solution
array([-3.,  2.])
```

Рисунок 13. Розв'язування системи лінійних рівнянь

1.4.3. Бібліотека Pandas

Подібно до NumPy, Pandas є однією з найбільш широко використовуваних бібліотек Python у науці про дані. Він забезпечує високопродуктивні, прості у використанні структури та інструменти аналізу даних. На відміну від бібліотеки NumPy, яка надає об'єкти для багатовимірних масивів, Pandas надає двовимірний об'єкт таблиці в пам'яті під назвою Dataframe. Це як електронна таблиця з назвами стовпців і мітками рядків.

Таким чином, за допомогою двовимірних таблиць pandas здатна надавати багато додаткових функцій, як-от створення зведених таблиць, обчислення стовпців на основі інших стовпців і побудова графіків.

Pandas можна імпортувати в Python за допомогою:

```
>>> import pandas as pd
```

Нижче наведено деякі типові структури даних у Pandas.

- Series objects: одновимірний масив, подібний до стовпця в електронній таблиці;
- DataFrame objects: двовимірна таблиця, схожа на електронну таблицю;
- Panel objects: Словник DataFrames, схожий на аркуш у MS Excel.

Pandas Series object створюється за допомогою функції pd.Series. Кожному рядку надається індекс і за замовчуванням йому призначаються числові значення, починаючи з 0. Як і NumPy, Pandas також надає базові математичні функції, такі як додавання, віднімання та умовні операції та трансляція.

Об'єкт фрейму даних Pandas представляє електронну таблицю зі значеннями комірок, назвами стовпців і мітками індексів рядків. Dataframe можна візуалізувати як словники серії. Доступ до рядків і стовпців Dataframe простий і інтуїтивно зрозумілий. Pandas також надає функціональність, схожу на SQL, для фільтрації та сортування рядків на основі умов. Наприклад,

```
>>> people_dict = { "weight": pd.Series([68, 83, 112],index=
["alice", "bob", "charles"]),   "birthyear": pd.Series([1984, 1985,
1992], index=["bob", "alice", "charles"], name="year"),
"children": pd.Series([0, 3], index=["charles", "bob"]),
"hobby": pd.Series(["Biking", "Dancing"], index=["alice", "bob"]),}
```

```
>>> people = pd.DataFrame(people_dict)
>>> people
```

```
>>>
people[people["birthyear"] <
1990]
```

Рисунок 14. Фільтрація та сортування рядків.

Нові стовпці та рядки можна легко додати до фрейму даних. На додаток до основних функцій, фрейм даних pandas можна сортувати за певним стовпцем.

Фрейми даних також можна легко експортувати та імпортувати з баз даних HTML, CSV, Excel, JSON, і SQL. Деякі інші важливі методи, які присутні у фреймах даних:

- head(): повертає 5 верхніх рядків в об’єкті кадру даних;
- tail(): повертає 5 нижніх рядків у кадрі даних;
- info(): друкує підсумок фрейму даних;
- describe(): надає хороший огляд основних агрегованих значень для кожного стовпця.

1.4.4. Бібліотека matplotlib

Matplotlib — це бібліотека двовимірних графіків, яка створює цифри якості публікацій у різних форматах друкованих копій та інтерактивних середовищах. Matplotlib можна використовувати в сценаріях Python, оболонці Python і IPython, записнику Jupyter, серверах веб-додатків і наборах інструментів GUI.

matplotlib.pyplot — це набір функцій, завдяки яким matplotlib працює як MATLAB. Більшість команд побудови в pyplot мають аналоги MATLAB із подібними аргументами. Наведемо кілька прикладів:

```
>>> import matplotlib.pyplot
as plt
>>> plt.plot([1,2,3,4])
>>> plt.ylabel('some
numbers')
>>> plt.show()
```

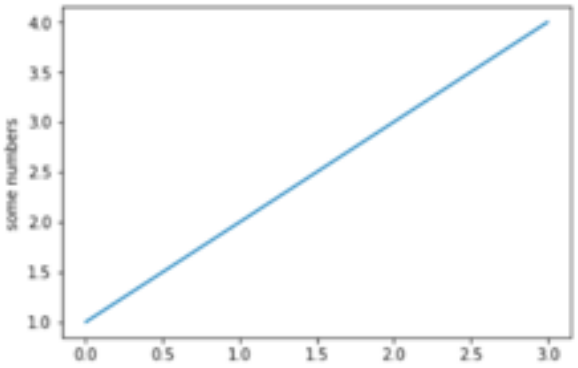


Рисунок 15. Приклад 1: Побудова лінійного графіка.

```
>>> import matplotlib.pyplot
as plt
>>> x =
[21,22,23,4,5,6,77,8,9,10,31
,32,33,34,35,36,37,18,49,50,
100]
>>> num_bins = 5
>>> plt.hist(x, num_bins,
facecolor='blue')
>>> plt.show()
```

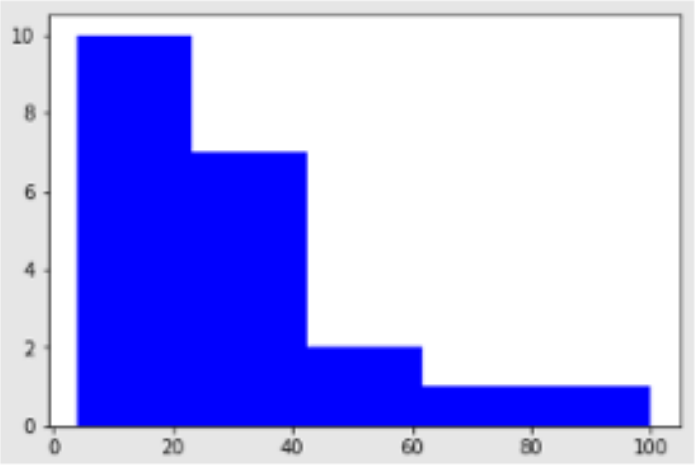


Рисунок 16. Приклад 2: Побудова гістограми.

Отже, ми бачимо, що NumPy і Panda спрощують маніпулювання матрицею. Ця гнучкість робить їх дуже корисними для розробки моделей машинного навчання.

1.5. Постановка задачі

Мені було поставлено завдання оглянувши технології створення web-сайтів та методи оброки даних, з’ясовати яка мова та фреймворками вважається найбільш популярними та є прийнятними. Було обрано мови програмування javaScript та Python, фреймворк express.js та бібліотеки Pandas, EEL.

JavaScript це високорівнева мова програмування, яка надає багатоплановість її застосування. Мова програмування Python обрана в зв’язку з тим, що останнім часом все частіше використовується для аналізу даних як у науці, так і комерційній сфері. Цьому сприяє простота мови, а також велика різноманітність відкритих бібліотек.

Необхідо було продемонструвати модель web-застосунку обробки великих обсягів даних, які накопичуються після проходження тестуванань користувачами даного web-сайту. Для цього було обрано діаграма варіантів використання (UseCase diagram). Показану групи користувачів та функціонал який їм доступний.

У web-застосунку розроблено звичайні користувачі мають можливість реєструватися, авторизуватися та відновлювати свій обліковий запис. Для релізації даного функціоналу було задіяно платформу SendGrid. На мові програмування JavaScrip розроблено функціонал анкетування та психофізичного тестування, результати якого автомаично зберігаються в MongoDB - документоорієнтована система управління базами даних.

Фахівці мають додатково доступ до сторінки де надається доступ до

перегляду результатів тестування всіх користувачів. Та до заздалегіть запрограмованих методів обробки даних після обрання деяких параметрів.

Адміністратор має доступ не лише до сторінок описаних раніше. Додатково має можливість змінювати статус «Користувач» на «Фахівець» при необхідності.

Також для кращої орієтації всіх відвідувачів сайту розроблено інструкцію користувача.

Висновки за розділом 1

** ДОПОВНИТЬ **

Отже, ми бачимо, що NumPy і Panda спрощують маніпулювання матрицею. Ця гнучкість робить їх дуже корисними для розробки моделей машинного навчання.

Оглянувши технології створення web-сайтів було з’ясовано, що найбільш популярними мовами та фреймворками вважається javaScript і express.js, PHP і Perl, Adobe Flash. і SilverLight.

Технологія Model-view-controller є часто використовуваною, при побудові архітектурного каркаса. Було, з’ясовано, що графічні редактори Adobe Photoshop, GIMP є кращими.

З’ясовано основні завдання та особливий функціонал, який необхідно реалізувати.

РОЗДІЛ 2. РОЗРОБКА КОМП'ЮТЕРНОЇ МОДЕЛІ WEB-ДОДАТКУ ОБРОБКИ ВЕЛИКИХ ОБСЯГІВ ДАНИХ

2.1. Розробка структури сайту

У UML діаграми варіантів використання моделюють поведінку системи та допомагають зафіксувати вимоги до системи. Діаграми варіантів використання описують високорівневі функції та сферу застосування системи. Ці діаграми також визначають взаємодію між системою та її учасниками. Варіанти використання та дійові особи у діаграмах варіантів використання описують, що робить система і як дійові особи її використовують, але не те, як система працює всередині.

Діаграми варіантів використання ілюструють та визначають контекст та вимоги або всієї системи, або її важливих частин. Ви можете змоделювати складну систему за допомогою однієї діаграми варіантів використання або створити множину діаграм варіантів використання для моделювання компонентів системи. Зазвичай діаграми варіантів використання розробляються

на ранніх стадіях проекту, і до них звертаються протягом процесу розробки.

Діаграми варіантів використання корисні у таких ситуаціях:

- Перед початком проекту ви можете створити діаграми варіантів використання для моделювання бізнесу, щоб усі учасники проекту мали загальне уявлення про працівників, клієнтів та діяльність бізнесу.
- При зборі вимог можна створити діаграми варіантів використання, щоб зафіксувати вимоги до системи та уявити іншим, що має робити система.
- На етапах аналізу та проектування ви можете використовувати сценарії використання та дійові особи з діаграм сценаріїв використання для визначення класів, які потрібні системі.
- На етапі тестування можна використовувати діаграми варіантів використання для визначення тестів для системи.

У наступних темах описуються елементи моделі у діаграмах варіантів використання:

Варіанти використання Варіант використання визначає функцію, яку система виконує для досягнення мети користувача. Варіант використання повинен давати результат, що спостерігається, який представляє цінність для користувача системи.

Актори

Актор є роль користувача, який взаємодіє з системою, яку ви моделюєте. Користувач може бути людиною, організацією, машиною чи іншою зовнішньою системою.

Підсистеми

У моделях UML підсистеми - це тип стереотипних компонентів, які є незалежними, поведінковими одиницями системи. Підсистеми використовуються в діаграмах класів, компонентів і сценаріїв використання для

представлення великомасштабних компонентів системи, що моделюється.

Відносини у діаграмах варіантів використання

У UML ставлення – це зв'язок між елементами моделі. Відносини UML – це тип елемента моделі, який додає семантику до моделі, визначаючи структуру та поведінку між елементами моделі.

Загальна діаграма «Комп'ютерної модель обробки великих обсягів даних» зображена на рисунку 17.

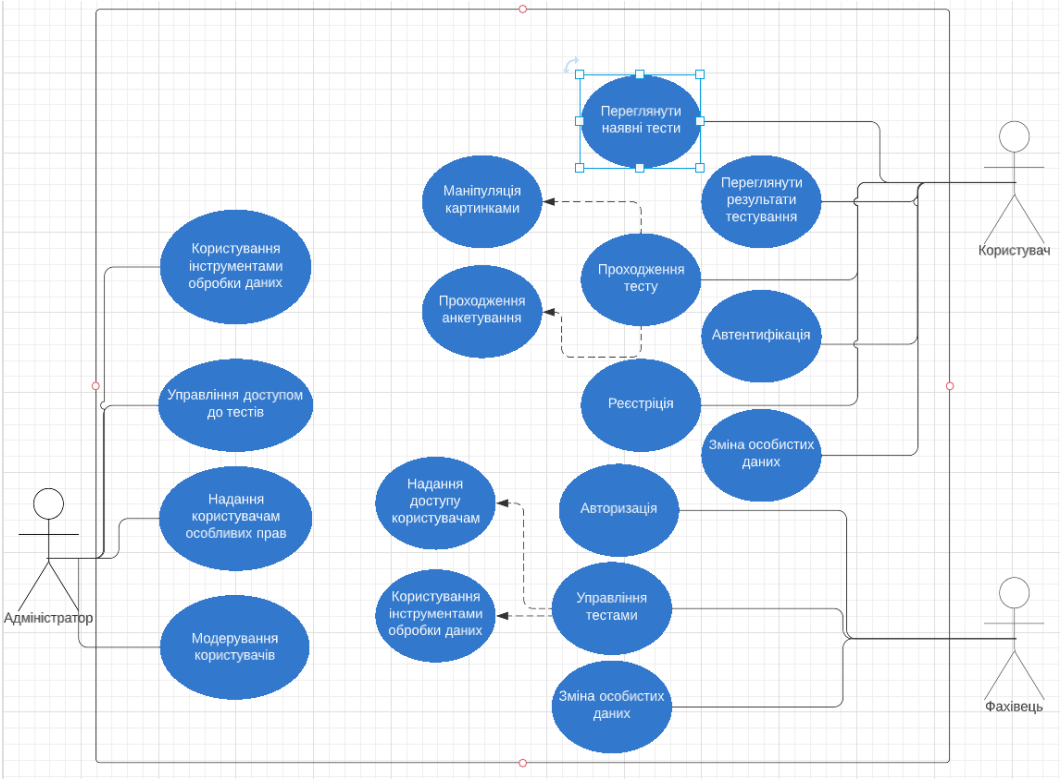


Рисунок 17. Діаграма варіантів використання (UseCase diagram) "Комп'ютерної модель обробки великих обсягів даних "

2.2. Робота з JavaScript

2.2.1. Налаштування програмної оболонки Node.js

Node.js — це платформа для створення швидких і масштабованих

серверних програм за допомогою JavaScript. Node.js — це середовище виконання, а npm — це менеджер пакетів для модулів Node.js.

У Visual Studio Code готова підтримка мов JavaScript і TypeScript, а також налагодження Node.js. Однак, щоб запустити програму Node.js, потрібно встановити середовище виконання Node.js на машині.

Інсталюємо Node.js для своєї платформи . Менеджер пакетів Node входить до дистрибутива Node.js. Потрібно відкрити новий термінал (командний рядок), щоб інструменти node і npm.

Щоб перевірити, чи правильно встановлено Node.js на комп'ютері, необхідно відкрити новий термінал і вести node --version, і буде видно встановлену поточну версію Node.js.

Linux : Існують спеціальні пакети Node.js для різних варіантів Linux. Слід перегляньте розділ « Встановлення Node.js за допомогою менеджера пакунків », щоб знайти пакет Node.js та інструкції зі встановлення, адаптовані до потрібної версії Linux.

Підсистема Windows для Linux : якщо використовується Windows, WSL — чудовий спосіб розробки Node.js. Можна запускати дистрибутиви Linux у Windows і інсталювати Node.js у середовищі Linux. У поєднанні з розширенням WSL отримуємо повну підтримку редагування коду VS і налагодження під час роботи в контексті WSL.

2.2.2. Інсталювання Express

Express — це дуже популярна програма для створення та запуску програм Node.js. Можна розмістити (створити) нову програму Express за допомогою інструменту Express Generator. Експрес-генератор постачається як модуль npm і встановлюється за допомогою інструмента командного рядка npm npm.

Для того щоб перевірити, чи npm правильно встановлено на комп'ютері, необхідно ввести npm –help із терміналу, і буде видно документацію щодо

використання.

Після встановлення Express Generator, запускається наступне з терміналу: *npm install -g express-generator*

–g встановлює Express Generator глобально на машину, тож є можливість запускати його з будь-якого місця. Тепер можемо створювати нову програму Express, викликану myExpressApp за допомогою:

express myExpressApp --view pug

Буде створено нову папку myExpressAppз вмістом вашої програми. Параметри *--view pug* повідомляють генератору використовувати механізм шаблонів pug .

Щоб інсталювати всі залежності програми (знову поставляються як модулі npm), необхідо перейти до нової папки та виконати npm install:

cd myExpressApp

npm install

На цьому етапі повинні перевірити, чи працює програма. Створена програма Express містить package.json файл, який містить start сценарій для запуску node ./bin/www.

Це запустить програму Node.js.

З терміналу в папці програми Express треба виконати:

npm start

Веб-сервер Node.js запуститься, і зможемо перейти до http://localhost: , щоб побачити запущену програму.

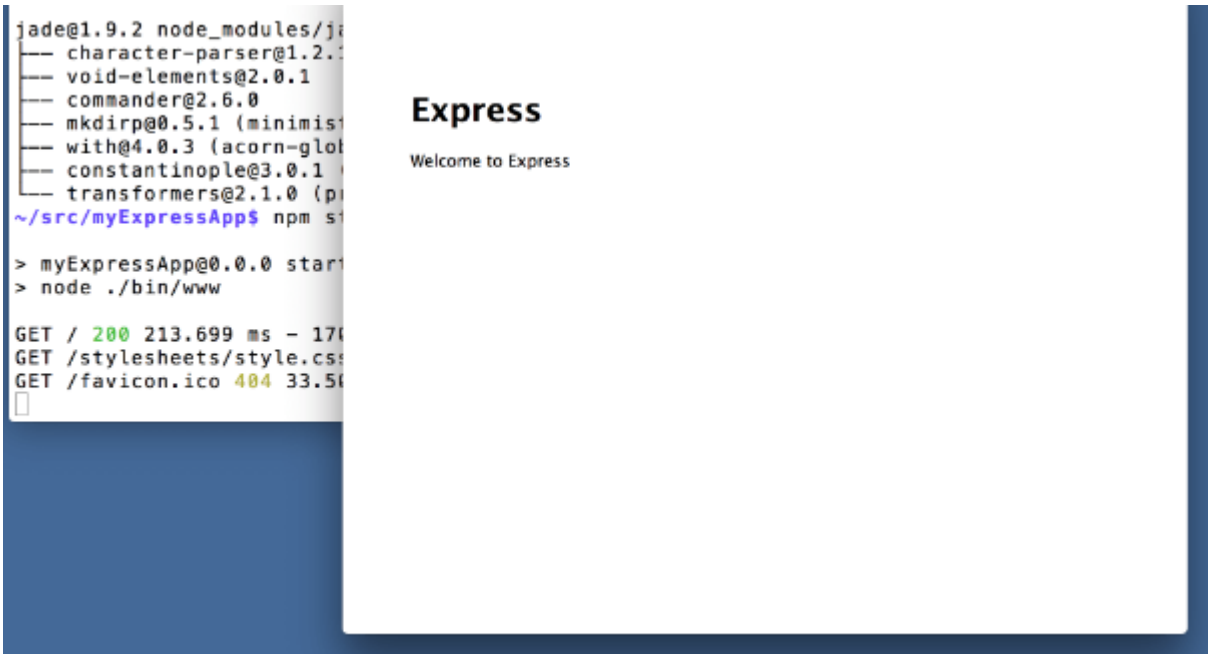


Рисунок 18. Вигляд запущеної програми.

2.2.3. Налагодження програми Express.

Потрібно створити файл конфігурації налагоджувача launch.json для програми Express. Необхідно натиснути «Запустити та налагодити» на панелі активності (Ctrl+Shift+D), а потім обирається посилання створити файл launch.json, щоб створити файл за замовчуванням launch.json. Обирається середовище Node.js, переконавшись, що type властивість у configurations має значення "node". Коли файл створюється вперше, VS Code шукатиме package.json сценарій starti використовуватиме це значення як program для конфігурації програми запуску .

```
{
  "version": "0.2.0",
  "configurations": [
    {
      "type": "node",
      "request": "launch",
      "name": "Launch Program",
      "program": "${workspaceFolder}\\bin\\www"
    }
  ]
}
```

Рисунок 19. Конфігурація програми.

При збереженні нового файл і переконання, що в спадному списку конфігурації у верхній частині перегляду « Запуск і налагодження » вибрано « Запустити програму» відкриваємо та встановіть точку зупину у верхній частині файлу, де створено об’єкт програми Express, клацнувши в області ліворуч від номера рядка. Натискаємо F5 , щоб почати налагодження програми. VS Code запустить сервер у новому терміналі та досягне встановленої нами точки зупину. Звідти можемо перевіряти змінні, створювати годинники та покроково виконувати код app.js.

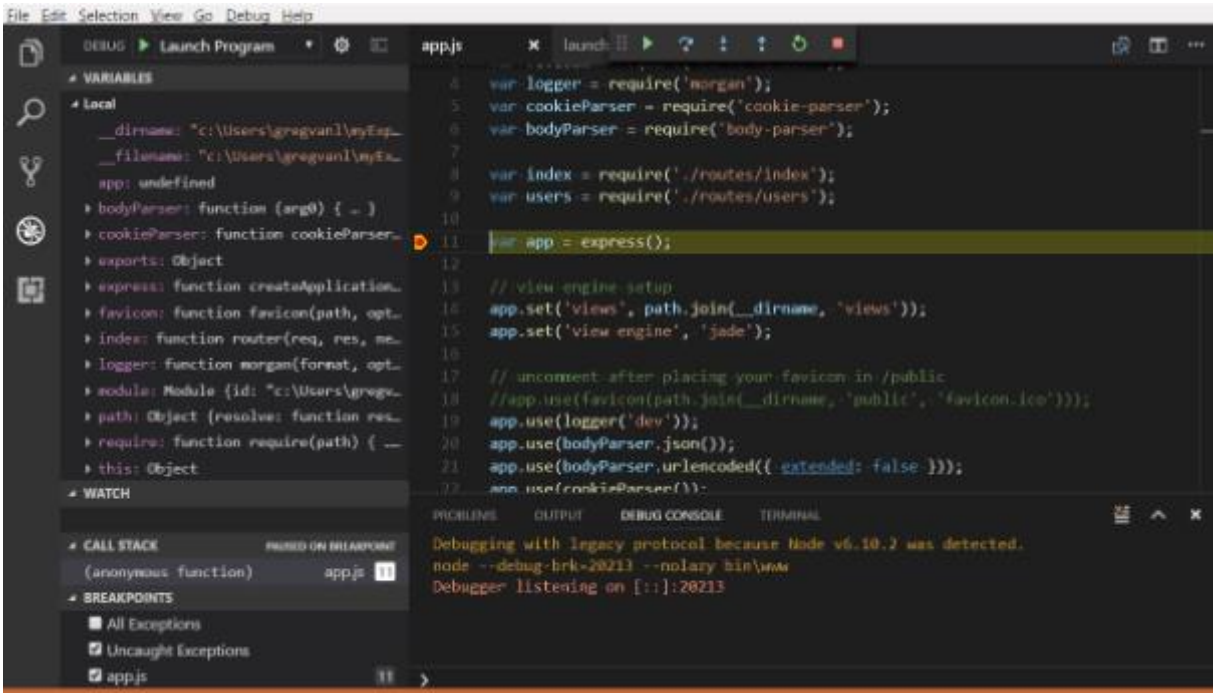


Рисунок 20. Демонстрація точки зупину в коді.

2.3. Робота Python & EEL

2.3.1. Розуміння бібліотеки Python EEl

Eel — це невелика бібліотека на мові програмування Python, яка дозволяє програмістам створювати прості офлайн-додатки з графічним інтерфейсом користувача (GUI), схожі на Electron, на основі HTML і JavaScript із повним доступом до можливостей і бібліотек Python.

Eel здатний розмістити локальний веб-сервер, а потім дозволяє програмістам коментувати методи та функції в Python, щоб ми могли викликати їх із JavaScript і навпаки. Ця бібліотека створена для того, щоб усунути проблеми з написанням простих і коротких графічних програм.

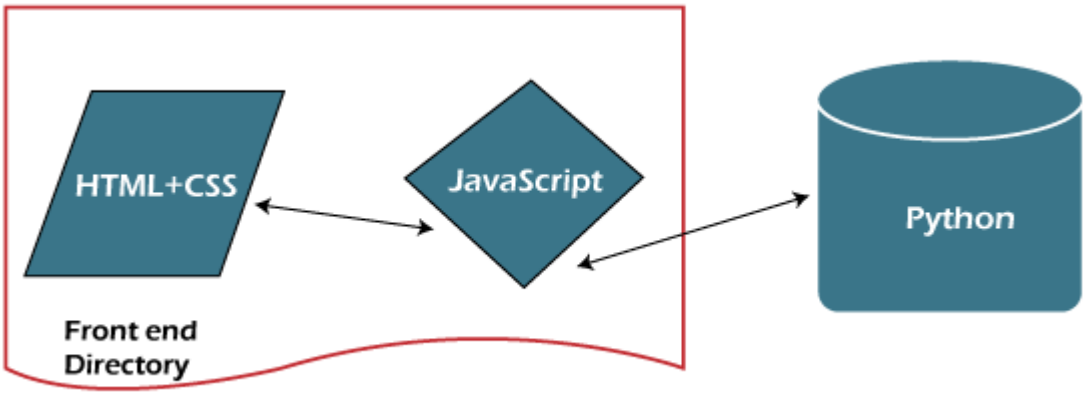


Рисунок 21. Відображення поняття EEL.

Файли HTML, CSS і JavaScript зберігаються в одному каталозі та ініціалізуються за допомогою програми python. Файл JS забезпечує зв'язок між сторінкою HTML і Python за допомогою функцій, активованих подіями onclick зі сторінки HTML.

2.3.2. Реалізація бібліотеки Eel

У розділі йдеться про створення простого додатку за допомогою бібліотеки Eel для демонстраційних цілей. Ідея програми полягає в тому, що програма приймає два числа зі сторінки HTML, додає їх у Python і показує результат користувачеві.

Але перш ніж почати створювати програму, необхідно встановити бібліотеку Eel.

Використаємо інсталятор pip, щоб встановити необхідну бібліотеку. Синтаксис для того ж наведено нижче:

Синтаксис: $pip install eel

Далі використовується код Visual Studio для створення цього проекту. Створили папку з назвою «Web_test». У цій папці створили іншу папку під назвою «myWeb». У цій папці зберігатимуться такі файли:

- Html
- js-file.js
- py

Для сторінки HTML створено три текстові області та кнопку. Давайте розглянемо наступний фрагмент коду для того ж.

```
<html>
  <head>
    <title>
      Sum Application
    </title>
    <script type = "text/javascript" src = "js-file.js"></script>
    <script type = "text/javascript" src = "/eel.js"></script>
  </head>
  <body>
    <label>Enter First Number
    <textarea id = "int1"></textarea>
    <label>Enter Second Number
    <textarea id = "int2"></textarea>
    <label>Result
    <textarea id = "res"></textarea>

    <button type = "button" id = "add" onclick = "summation()">
      Sum
    </button>

  </body>
</html>
```

Рисунок 22. Файл myWebpage.html.

У наведеному вище фрагменті коду створили HTML-сторінку, де визначили три текстові області з різними ідентифікаторами. «int1» та «int2» призначені для отримання вхідних даних від користувача, а текстова область

«res» — для відображення результату. Кнопка з id = 'sum' має функцію onclick, приєднану до неї, визначену у файлі JS.

Тепер давайте розглянемо фрагмент коду для файлу JS.

```
document.querySelector("button").onclick = function summation(){
    var data_1 = document.getElementById("int1").value
    var data_2 = document.getElementById("int2").value
    eel.add(data_1, data_2)(call_Back)
}

function call_Back(output){
    document.getElementById("res").value = output
}
```

Рисунок 23. Файл js-file.js

У наведеному вище фрагменті коду визначили функцію як summation(), яка активується через подію onclick. У цій функції отримали значення «int1» і «int2» і передали їх у функцію «sum», яка буде визначена у файлі Python. Функція з назвою call_Back() приймає «вихід» як аргумент, що повертається методом Python під назвою «sum». Цей «вихід» повертається в текстовій області з ідентифікатором «res». Тепер давайте розглянемо фрагмент коду для файлу Python.

```
# importing the eel library
import eel
# initializing the application
eel.init("myWeb")

# using the eel.expose command
@eel.expose
# defining the function for addition of two numbers
def add(data_1, data_2):
    int1 = int(data_1)
    int2 = int(data_2)
    output = int1 + int2
    return output

# starting the application
eel.start("myWebpage.html")
```

Рисунок 24. Файл my_sum.py.

У наведеному вище фрагменті коду імпортували бібліотеку eel та ініціалізували програму за допомогою команди eel.init("folder name"), де призначили "myweb" як назву папки.

Потім ми використали команду eel.expose, у якій визначили функцію «add», яка додасть «data_1» до «data_2» і поверне результуюче значення як «вихід». Значення, що зберігається у «виводі», отримується функцією «call_Back» у файлі JS і розміщується на веб-сторінці HTML.

Потім ініціалізували програму за допомогою команди eel.start("html filename"), вказавши назву файлу як "myWebpage.html".

Далі зберігається файл python і виконується, використовуючи наступний синтаксис, щоб побачити результат

Синтаксис:

$ python my_sum.py

На наступному рисунку показано розроблений інтерфейс користувача.

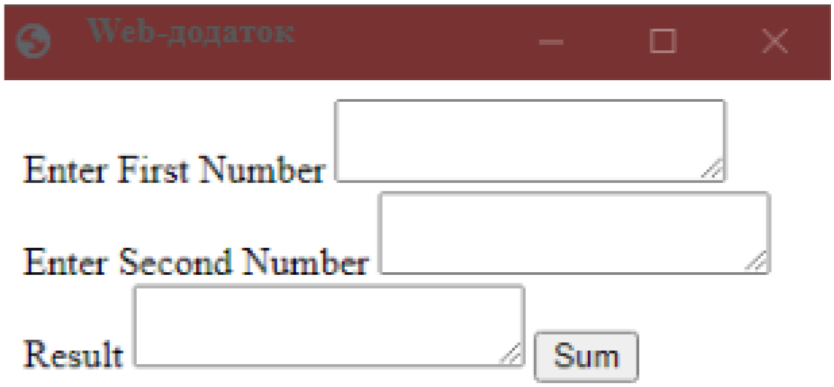


Рисунок 25. Тетсовий інтерфейс користувача

За замовчуванням бібліотека Python Eel використовує веб-браузер Google Chrome для запуску програми; однак ми також можемо згадати браузер за допомогою параметра «режим». Окрім «режиму», доступно кілька інших варіантів програм, таких як «положення», «розмір», «геометрія» та багато іншого, які передаються всередині команди eel.start().

2.4. Обробка даних на Python

2.4.1. Правила візуалізації даних

Візуалізація даних є дуже важливою частиною аналізу даних. Можна використовувати його для вивчення своїх даних. Якщо добре розумієте свої дані, то буде більше шансів знайти певну інформацію. Коли є інформація, можна використовувати візуалізацію, щоб мати можливість поділитися своїми висновками з іншими людьми.

Перш ніж ми розглядати типи діаграм, познайомимося з деякими основними правилами. Ці правила допомагають створювати красиві та інформативні діаграм замість заплутаних.

- Перший крок - вибрати відповідний тип ділянки. Якщо є різні варіанти, можемо спробувати їх порівняти та вибрати той, який найкраще підходить для даної моделі.
- По-друге, коли обираємо тип графіка, одна з найважливіших речей – це позначити вісь . Якщо цього не зробити, то діаграма недостатньо інформативна. Якщо міток осей немає, можна спробувати переглянути код, щоб побачити, які дані використовуються, і якщо нам пощастить, то зрозуміємо діаграми. Але що, якщо маємо лише діаграми як зображення?
- По-третє, можемо додати назву, щоб зробити діаграми більш інформативним .
- По-четверте, за потреби слід додати мітки для різних категорій.
- По-п’яте, за бажанням можемо додати текст або стрілку в цікаві точки даних .
- По- шосте, у деяких випадках можемо використовувати деякі розміри та кольори даних, щоб зробити графік більш інформативним.

2.4.2. Візуалізація з Matplotlib

Існує багато типів візуалізацій. Деякі з найвідоміших: лінійна діаграма, точкова діаграма, гістограма, коробчаста діаграма, стовпчаста діаграма та кругова діаграма. Але як серед багатьох варіантів вибрати правильну візуалізацію? По-перше, потрібно провести дослідницький аналіз даних. Коли ми дізнаємося форму даних, типи даних та іншу корисну статистичну інформацію, буде легше вибрати правильний тип візуалізації.

У Python є багато пакетів візуалізації. Одним з найвідоміших є Matplotlib. Його можна використовувати в сценаріях Python, оболонках Python і IPython, блокноті Jupyter і серверах веб-додатків.

Перш ніж перейти до визначень і прикладів, хочу показати деякі основні функції matplotlib.pyplot підпакетів, які побачимо в прикладах нижче. Тут припускається, що matplotlib.pyplot підпакет імпортовано з псевдонімом plt.

- plt.title(“My Title”) - додасть назву «Мій заголовок» до вашої ділянки;
- plt.xlabel(“Year”) - додасть мітку «Рік» до вашої осі x;
- plt.ylabel(“Population”) - додасть мітку «Населення» до вашої осі Y;
- plt.xticks([1, 2, 3, 4, 5])встановіть числа на осі x як 1, 2, 3, 4, 5. Також можемо передати мітки і як другий аргумент. Наприклад, якщо використовуємо цей код `plt.xticks([1, 2, 3, 4, 5], ["1M", "2M", "3M", "4M", "5M"])`, він встановить мітки 1M, 2M, 3M, 4M, 5M на осі x;
- plt.yticks() - працює так само, як plt.xticks(), але для осі y.

Лінійний графік : тип графіка, який відображає інформацію як ряд точок даних , які називаються «маркерами» , з’єднаних прямими лініями . У цьому типі графіка потрібно впорядкувати точки вимірювання (зазвичай за їхніми значеннями на осі X). Цей тип графіка часто використовується для візуалізації тенденції в даних за проміжки часу - часовий ряд.

Щоб створити лінійний графік за допомогою Matplotlib, викликаємо plt.plot(). Перший аргумент використовується для даних на горизонтальній осі, а другий – для даних на вертикальній осі. Ця функція створює графік, але не відображає його. Щоб відобразити графік, потрібно викликати функію plt.show(). Це зручно, тому що можемо додати деякі додаткові налаштування до нашого графіку, перш ніж відобразити його. Наприклад, можемо додати мітки до осі та назву графіка.

```
1  import matplotlib.pyplot as plt
2
3  years = [1983, 1984, 1985, 1986, 1987]
4  total_populations = [8939007, 8954518, 8960387, 8956741, 8943721]
5
6  plt.plot(years, total_populations)
7  plt.title("Year vs Population ")
8  plt.xlabel("Year")
9  plt.ylabel("Total Population")
10 plt.show()
```

Рисунок 26. Лістинг простого лінійного графіку

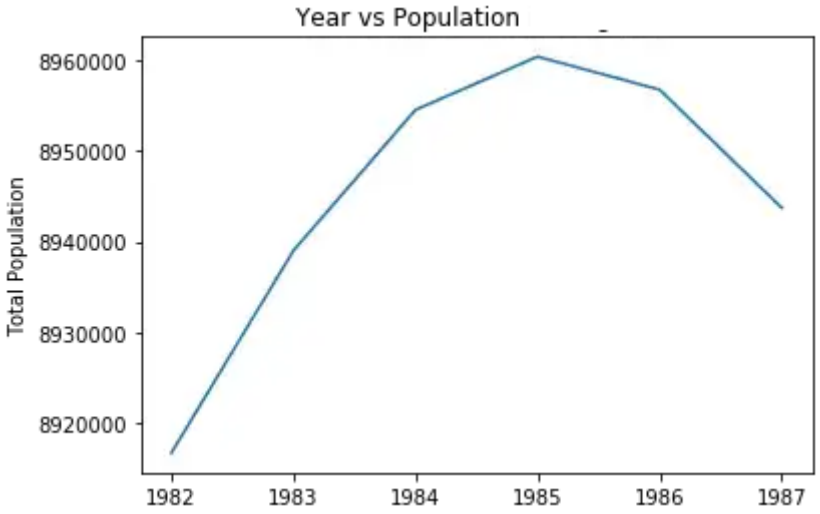


Рисунок 27. Простого лінійного графіку

Точкова діаграма : цей тип діаграми показує всі окремі точки даних. Тут вони не пов'язані лініями. Кожна точка даних має значення на осі X і значення на осі Y. Цей тип графіка можна використовувати для відображення тенденцій або кореляцій. У науці про дані це показує, як порівнюються дві змінні. Щоб створити точкову діаграму за допомогою Matplotlib, можемо використати plt.scatter() функцію. Знову перший аргумент використовується для даних по горизонтальній осі, а другий - для вертикальної осі.

```python
import matplotlib.pyplot as plt

temp = [30, 32, 33, 28.5, 35, 29, 29]
ice_creams_count = [100, 115, 115, 75, 125, 79, 89]

plt.scatter(temp, ice_creams_count)
plt.title("Temperature vs. Sold ice creams")
plt.xlabel("Temperature")
plt.ylabel("Sold ice creams count")
plt.show()
```

Рисунок 28. Лістинг точкової діаграми.

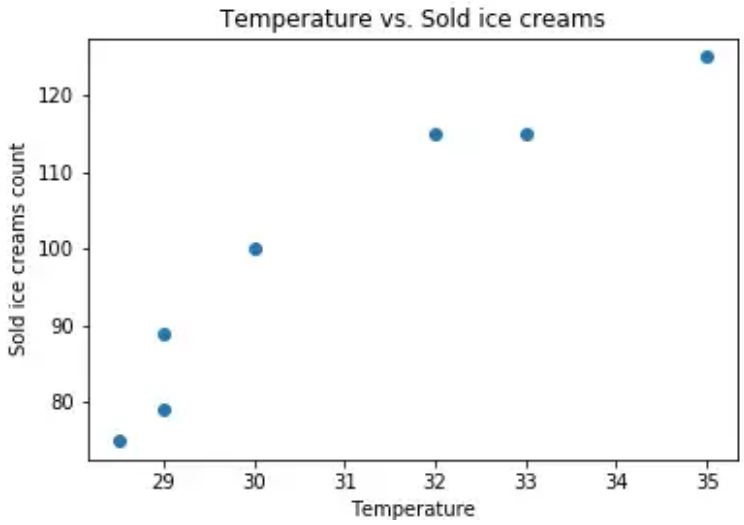


Рисунок 29. Точкова діаграма.

Гістограма : точне представлення розподілу числових даних. Щоб створити гістограму, по-перше, ділимо весь діапазон значень на серію інтервалів, а по-друге, підраховуємо, скільки значень потрапляє в кожен інтервал . Інтервали також називають бінами . Біни — це послідовні інтервали змінної, які не перекриваються. Вони повинні бути поруч і часто однакового розміру. Щоб створити гістограму за допомогою Matplotlib, можемо використати plt.hist() функцію. Перший аргумент - це числові дані, другий аргумент - кількість бінів. Значення binsаргументу за замовчуванням — 10.

```
1  import matplotlib.pyplot as plt
2
3  numbers = [0.1, 0.5, 1, 1.5, 2, 4, 5.5, 6, 8, 9]
4
5  plt.hist(numbers, bins = 3)
6  plt.xlabel("Number")
7  plt.ylabel("Frequency")
8  plt.show()
```

Рисунок 30. Лістинг гістограми

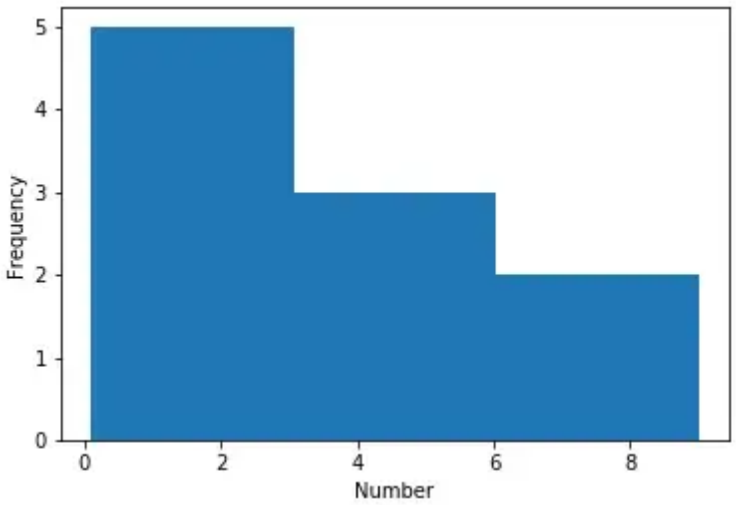


Рисунок 31. Гістограма.

З гістограми вище ми бачимо, що є:

- 5 значень від 0 до 3;
- 3 значення від 3 до 6 (включно) ;
- 2 значення між 6 (за винятком) і 9.

Висновки за розділом 2

** ДОПОВНИТЬ **

У наведеному вище ми обговорювали бібліотеку Eel та її використання в

Python. Ми дізналися, що ця бібліотека допомагає програмістам створювати автономні програми на основі HTML і JavaScript. За допомогою бібліотеки Eel ми отримуємо можливості Python, включаючи свободу проектування інтерфейсу користувача за допомогою HTML, а також CSS. Хоча за популярністю Eel поступається своєму основному конкуренту Electron; однак є багато програмістів, які першим вибирають Eel для створення простих програм із графічним інтерфейсом користувача (GUI).

В даному розділі було описано поняття діаграми варіантів використання (UseCase diagram) та з’ясовано принцип її побудови. Розроблено діаграму “Комп’ютерної моделі Web-сервісу організації роботи психолога”.

Також описано основне програмне забезпечення та утиліти. А саме йде мова про:

- Серидовище виконання JavaScript – Node.js.
- Утиліти nodemon.
- Handlebars – шаблонний процесор.
- MongoDB – СУБД для Node.js.
- Mongoose – бібліотека JavaScript.
- CSRF-захист.
- SendGrid – поштовий сервіс.
- Heroku – хмарна PaaS-платформа.

РОЗДІЛ 3. РЕКОМЕНДАЦІЇ ПО ВИКОРИСТАННЮ КОМП'ЮТЕРНОЇ МОДЕЛІ WEB-СЕРВІСУ ОРГАНІЗАЦІЇ РОБОТИ ПСИХОЛОГА

3.1. Сторінки, що доступні всім користувачам.

При заході на web-сайт перш за все користувач потрапляє на головну сторінку рис. 32, де прописана інформація про цей web-застосунок.

При спробі перейти на інші сторіки сайту, користувачу вивовдиться повідоблення рис. 33 з пропозицією авторизуватися чи пройти реєстрацію.

Після підтвердження заросу користувач потрамляє на сторінку реєстрації рис. 34 та авторизації рис. 35.

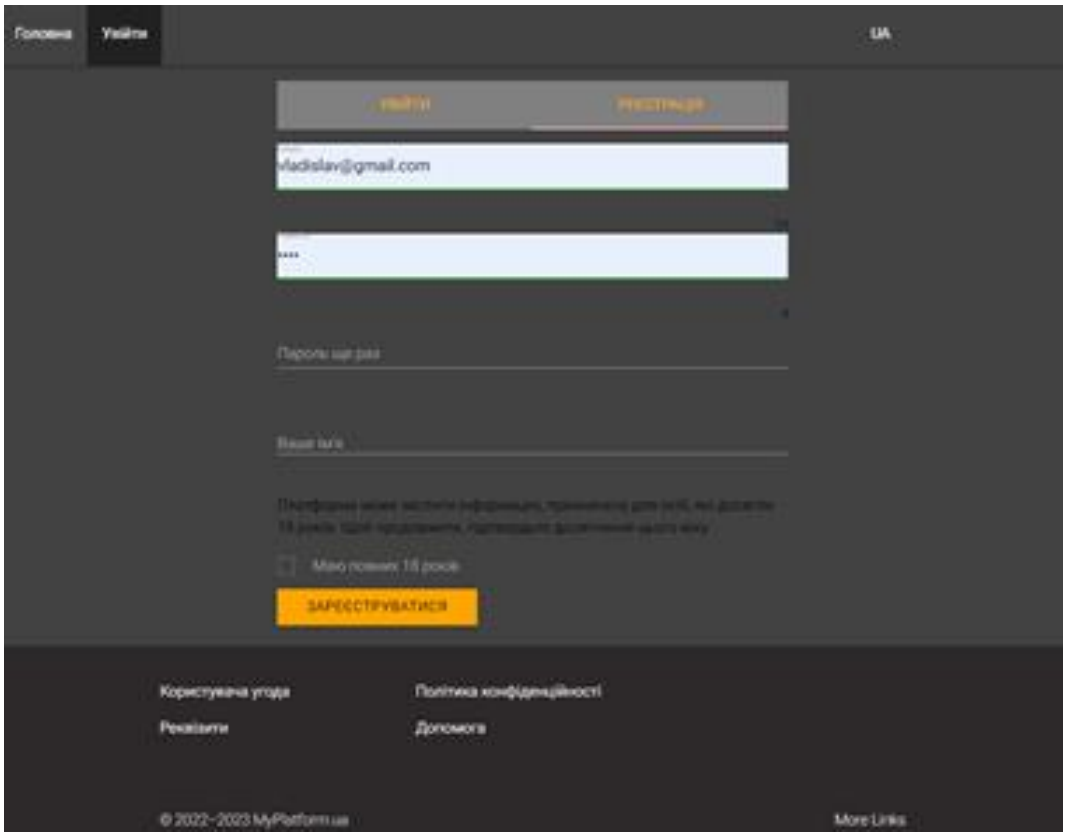


Рисунок 34. Сторінка реєстрації.

Для того щоб успішно пройти реєстрацію необхідно вказати свою дійсну електронну адресу, пароль та повторний пароль. Після погодження з правилами web-вебсайту можна натискати на кнопку «ЗАРЕЄСТРУВАТИСЯ». Після цього завантажиться головна сторінка.

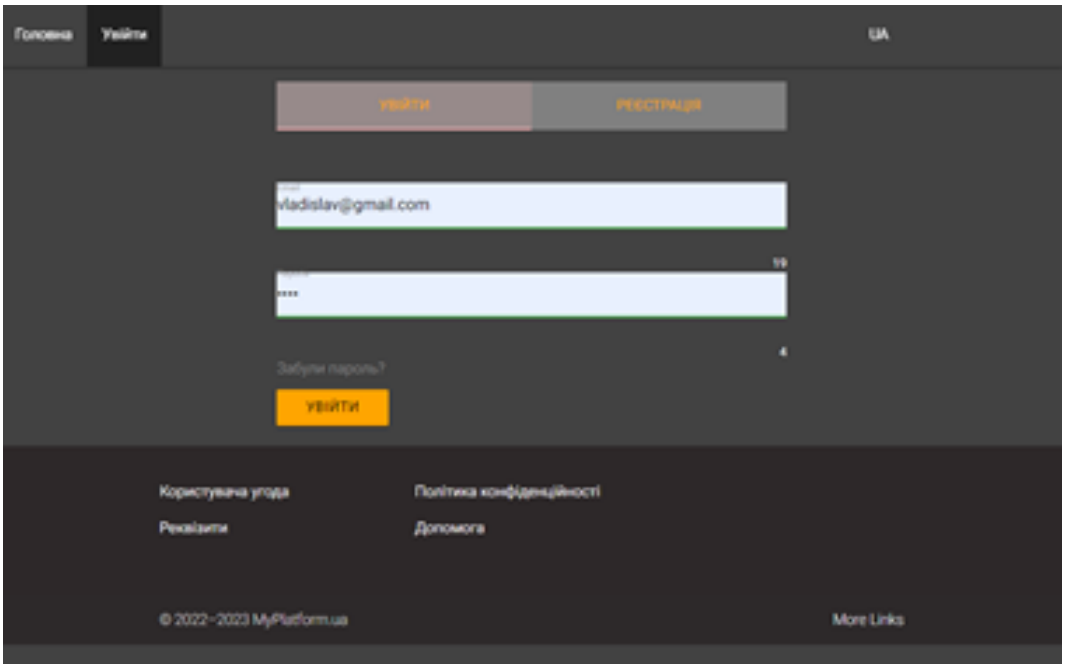


Рисунок 35. Сторінка авторизації.

У випадку якщо не виходить авторизуватися рис. ** передбачен функція відновлення паролю.

При натисканні на «Забули пароль?» користувача просять ввести свій поштовий адрес рис. **.

На вашу почту приходе повідомлення з інструкцією відновлення пароля рис. **.

При переході на сторінку тестів користувач обирає одини із тестів рис. **.

Обравши тремер тест користувач потрапляю на сторінку зображену на рисунку **.

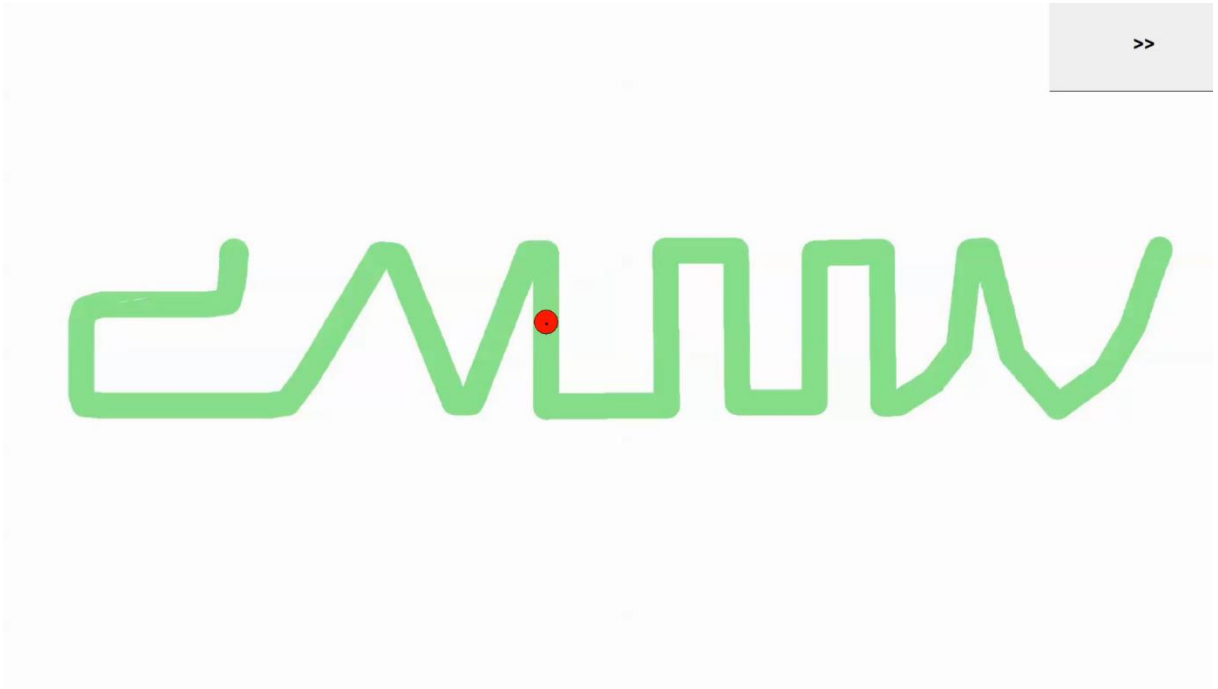


Рисунок **. Тремер тест.

3.2. Сторінки фахівця.

Після завершення тесту лише фахівець може бачити результати тестування рис. **.

ПІБ: Ткаченко Микола Іванович, Вік: 41, Стать: Ч;

Тест: Тремометрія (координаціяметрія)

Дата дослідження: 21.10.2022

Права спіраль:

№	Показник	Права рука	Ліва рука	Асим.
1	Закальна к-ть помилок, шт.	101	86	15
2	Час виконання, с	15,5	15,5	0,0
3	Середян частота помилки, 1/с	6,7	5,7	14,9
4	Сумарна тривалість помилки, мс	681,0	9408,0	27,5
5	Середня тривалість помилки, мс	109,4	67,5	38,3
6	Відносна тривалість помилки, мс	45,5	62,7	27,5
7	Дистанція, мм	5917,0	6133,0	3,5
8	Середня швидкість тремора, мм/с	0,4	0,4	3,5
9	Максимальне відхилення, мм	55,0	71,0	22,5

Висновок:

Кількість помилок на робочих етапах теста більше при роботі правою рукою, а ціна діяльності в цьому режимі нижче, ніж при роботі лівою рукою.

Рисунок **. Результати тремер тесту.

Висновки за розділом 3

** ДОПОВНИТЬ **

В розділі 3 було описано основні сторінки доступні користувачеві, їх функціонал. А саме:

- Головна сторінка. Містить інформацію про web-сайт.

ВИСНОВКИ

** ДОПОВНИТЬ **

Перш за все слід зазначити, що JS є однією з найбільш популярних мов для фронтенд-розробки. Для фронетнд розробника базовим інструментарієм слід вважати JavaScrip, CSS та HTML. На JavaScript створюються додатки, які виконуються в браузері на стороні клієнта. Вони забезпечують інтерактивність сайтів.

Сфери застосування JavaScript не обмежуються браузерами і веб-додатками. За допомогою цієї мови вирішують такі завдання:

Розробка нативних додатків. Наприклад, за допомогою фреймворка React Native створюються додатки для Android і iOS.

Розробка десктопних додатків.

Програмування обладнання та побутової техніки, наприклад, платіжних терміналів, телевізійних приставок.

Серверна розробка. Node.js застосовується для бекенд-розробки. Про цей напрямок у нас є окрема стаття.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. HTML & CSS, and JavaScript & JQuery (2 book set) by Jon Duckett.
2. Резиг Джон , Бибо Беэр Секреты JavaScript ниндзя; Вильямс - М., 2015. - 416 с.
3. Front-End Developer Handbook — 2019.
4. Хэррон Д. «Node.js. Разработка серверных веб-приложений в JavaScript»: Пер. с англ. Слинкина А.А. - М.: ДМК Пресс, 2012. - 144с.: ил.
5. The Pragmatic Programmer: Your Journey to Mastery.
6. Don’t Make Me Think: A Common Sense Approach to Web and Mobile Usability.
7. Макфарланд Дэвид JavaScript. Подробное руководство; Эксмо - М., 2015. - 608 с.
8. Макфарланд Дэвид JavaScript и jQuery. Исчерпывающее руководство (+ DVD-ROM); Эксмо - М., 2015. - 688 с.
9. Learn JavaScript VISUALLY with Interactive Exercises.
10. Ботт Эд. Разработка веб-сайтов. – М., 2004.
11. Дарнелл P.html 4 Енциклопедія користувача ДіаСофт 1999
12. Кінгслі-Хью Е., JavaScript: навчальний курс. - СПб.: Пітер, 2002
13. Чаффер Д. Изучаем jQuery 1.3. Эффективная веб-разработка на JavaScript; Символ-плюс - М., 2015. - 391 с.
14. Дронов Владимир JavaScript и AJAX в Web-дизайне; БХВ-Петербург - М., 2015. - 736 с.
15. Климов Александр JavaScript на примерах; БХВ-Петербург - М., 2017. - 812 с.

Додаток А

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В. Н. Каразіна

Факультет комп'ютерних наук
Кафедра теоретичної та прикладної системотехніки
Рівень вищої освіти (освітньо-кваліфікаційний рівень) магістр
Галузь знань: 12 – Інформаційні технології
Спеціальність: 123 «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри теоретичної та прикладної системотехніки
д.т.н., проф. Шматков С. І.

«10» 12 2021 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ (ПРОЕКТ)

Каплун Владислав Миколайович
(прізвище, ім'я, по батькові студента)

1. Тема роботи: комп'ютерна модель обробки великих обсягів даних.

керівник роботи Толстолузька Олена Генадіївна д.т.н., с.н.с,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від "___"_____20__року №_____

2. Строк подання студентом роботи 30 листопада 2022 р

3. Перелік питань, які потрібно розробити
1) Аналіз методів розробки Web-застосунку з обробкою великих обсягів даних.
2) Аналіз технологій розробки комп'ютерних моделей з великим обсягом даних.
3) Розробка комп'ютерної моделі Web-застосунку обробки великих обсягів даних.
4) Розробка програмної реалізації комп'ютерної моделі Web-застосунку обробки великих обсягів даних на основі визначених психофізичних здібностей людини.
5) Розробка рекомендацій з використання комп'ютеної моделі.

4. План роботи

№ з/п	Назви етапів роботи	Термін виконання виконання етапів роботи
1.	Аналіз методів розробки Web-застосунку з обробкою великих обсягів даних.	21.10.2021 - 12.11.2021
2.	Аналіз технологій розробки комп'ютерних моделей з веиким обсягом даних.	13.11.2021 - 04.12.2021
3.	Розробка комп'ютерної моделі Web-застосунку обробки великих обсягів даних на основі визначених психофізичних здібностей людини.	05.12.2021 - 02.02.2022
4.	Розробка програмної реалізації комп'ютерної моделі Web-застосунку обробки великих обсягів даних на основі визанчених психофізичних здібностей людини.	03.02.2022 - 10.04.2022
5.	Відладка та тестування розробленої комп'ютерної моделі.	11.04.2022- 27.06.2022
6.	Розробка рекомендацій з використання комп’ютеної моделі	28.06.2022- 21.08.2022
7.	Розробка та представлення дипломного проекту керівнику дипломної роботи та рецензенту.	21.08.2022- 30.11.2022

5. Дата видачі завдання 10.12.2021

Студент В. М. Каплун _____
ініціали, прізвище підпис

Керівник роботи О. Г. Толстолузька _____
ініціали, прізвище підпис

ІНДИВІДУАЛЬНЕ ТЕХНІЧНЕ ЗАВДАННЯ.

Додаток Б

Затверджую

_____

«_____»_____2022 р.

Технічне завдання

на розробку програмного виробу «Комп’ютерна модель обробки великих обсягів даних**»**

1.	Введення	1.1. Назва: Комп’ютерна модель обробки великих обсягів даних. 1.2. Галузь застосування: комп’ютерні технології.
2.	Підстава для розробки	2.1. Навчальний план за спеціальністю 123 – Комп’ютерна інженерія 2.2. Завдання на дипломну роботу магістра № _____ _____ від «___» _____2022 (представити як Додаток А до пояснювальної записки до дипломної роботи).
3.	Призначення розробки	3.1. Мета розробки: дослідження методів розробки комп'ютерної моделі Web-застосунку обробки великих обсягів даних на основі отриманих результатів тестування на визначення психофізичних здібностей людини. 3.2. Призначення розробки надає можливість користувачам з пристроїв які підключені но мережі проходити анкетування, психофізичне

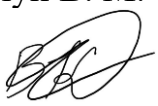
		тестування та додаткову можливість фахівцям психологам проводити аналіз отриманих даних. 3.3. Вихідні дані розробки: розроблена модель з функціоналом обробки даних отриманих після проходження тестування користувачами.
4.	Технічні вимоги до програмного виробу	4.1. Вимоги до функціональних характеристик:немає 4.2. Вимоги до надійності: забезепчувати точність і достовірність обробки даних та тестування. 4.3.Вимоги до умов експлуатації: немає 4.4. Вимоги до складу і параметрів технічних засобів: цілодобовий доступ з ПК, тощо на яому є доступ до мережі Internet 4.5. Вимоги до інформаційної та програмної сумісності: немає 4.6. Вимоги до маркування та упаковки: немає 4.7. Вимоги до транспортування і зберігання: на звичайних носіях інформації 4.8. Спеціальні вимоги: немає.
5.	Вимоги до програмної документації	Програмною документацією до виробу «комп'ютерної моделі Web-застосунку обробки великих обсягів даних на основі отриманих результатів» вважати: 1) Справжнє Технічне завдання на розробку виробу (представити у вигляді Додатку Б до пояснювальної записки до кваліфікаційної роботи). 2) Розроблена структура сайту та налуштавання програмної оболонки (у вигляді розділу 2 пояснювальної записки до кваліфікаційної роботи).

		3) Рекомендацій по використанню (представити в розділі 3 пояснювальної записки до кваліфікаційної роботи)	
6.	Вимоги до техніко-економічних показників	Програмною документацією до виробу «комп'ютерної моделі Web-застосунку обробки великих обсягів даних на основі отриманих результатів» вважати: 1) Справжнє Технічне завдання на розробку виробу (представити у вигляді Додатку Б до пояснювальної записки до кваліфікаційної роботи). 2) Розроблена структура сайту та налуштавання програмної оболонки (у вигляді розділу 2 пояснювальної записки до кваліфікаційної роботи). 3) Рекомендацій по використанню (представити в розділі 3 пояснювальної записки до кваліфікаційної роботи)	
7.	Стадії і етапи розробки	Дата	Назва етапу
		від 21 жовтня 2021 до 12 листопада 2021	Аналіз методів розробки Web-застосунку з обробкою великих обсягів даних.
		від 13 листопада 2021 до 4 грудня 2022	Аналіз технології розробки комп’ютерних моделей з великим обсягом даних. .
		від 5 грудня 2021 до 02 січня 2022	Розробка комп’ютерної моделі з великим обсягом даних на основі визначених

			психофізичних здібностей людини.
		від 03 січня 2022 до 10 квітня 2022	Розробка програмної реалізації комп'ютерної моделі з великим обсягом даних на основі визначених психофізичних здібностей людини.
		від 11 квітня 2022 до 27 червня 2022	Відладка та тестування розробленої комп'ютеної моделі.
		від 28 червня 2022 до 21 серпня 2022	Розробка рекомендацій з використання комп'ютерної моделі.
		від 21 серпня 2022 до 30 листопада 2022	Розробка та представлення дипломного проекту керівнику дипломної роботи та рецензенту.
8.	Порядок контролю і приймання програмного продукту	1. Перевірку ходу розробки програми виконувати раз в 3 тижні. 2. Захист розробленої моделі провести на засіданні Атестаційної комісії.	

	(моделі)	3. Пояснювальну записку подати на паперових носіях в 1 примірнику і в електронному вигляді в 1 примірнику на CD-R компакт-диску.

Виконавець
студент групи КІ- 61
Каплун В. М.



_____

Замовник
д.т.н., с.н.с.
Толстолузька О. Г.

_____

Додаток В

Затверджую

_____

«____»_____2022р.

Програма і методика випробувань

програмного вироби

«Комп'ютерна модель обробки великих обсягів даних»

1. Об'єкт випробувань

1.1 Найменування випробуваного програмного виробу: модель Web-сервісу організації роботи психолога.

1.2 Область його застосування: *комунікація спеціаліста з клієнтом для проведення тестування*.

2. Мета випробувань

Перевірка працездатності продукту та правильності роботи розробленої моделі. Підтвердження характеристик розробленого виробу вимогам, які сформульовані в ТЗ (Додаток Б).

3. Загальні положення

3.1 Підстави для проведення випробувань

Підставою для проведення випробувань є наказ про призначення атестаційної комісії та перевірку даного виробу.

3.2 Місце і тривалість випробувань

Приймальні (приймально-здавальні) випробування проводяться на базі комп'ютерного класу або будь-якого приміщення в період роботи атестаційної комісії.

3.3 Обсяг випробувань

Приймальні випробування програмного вироби проводяться в обсязі відповідному цієї програми і методики випробувань.

3.4 Організації, які беруть участь у випробуваннях

Приймальні випробування проводяться атестаційною комісією напередодні засідання (або в процесі засідання) за участю Замовника, Виконавця та інших осіб, присутніх на засіданні.

4. Вимоги до програми або програмного виробу

Модель повинна представляти можливість користувачеві створити чи авторизувати свій обліковий запис та пройти тестування, фахівець повинен мати можливість проводити обробку даних.

5. Вимоги до програмної документації

Склад програмної документації, що подається на випробування, включає:

1) Технічне завдання на розробку моделі (Наведене в Додатку А пояснювальної записка до дипломної роботи)
2) Програму і методику випробувань розробленого програмного виробу (Наведено в Додатку В пояснювальної записка до дипломної роботи)
3) Опис виробу (Буде представлено в Розділі 2 пояснювальної записка до дипломної роботи)
4) Лістинги програми (Буде наведено в Додатку Г пояснювальної записка до дипломної роботи)

6. Засоби і порядок випробувань

6.1 Засоби випробувань

Необхідна наявність ПК з встановленим браузер та надійним з’єднання з Інтернетом.

6.2 Порядок проведення випробувань

6.2.1 Перевірка документацій.

Перелік перевірок, що проводяться на 1 етапі випробувань, включає в себе:

1) Перевірка відповідності та актуальності програмної документації за критерієм наявності зазначеної в ТЗ документації.
2) Перевірка якості документації на відповідність до стандартів *Єдиної системи програмної документації (ЄСПД)*.

6.2.2 Перевірка робоздатності моделі.

Користувач з використанням ПК, попередньо маючи посилання на веб-додаток, переходить до нього та протрапляю на сторінку авторизації.

6.2.3 Перевірка ступеня виконання функціональних вимог до Web-сервісу.

1) Реєстрація користувача та авторизація користувача (Рис. 1).

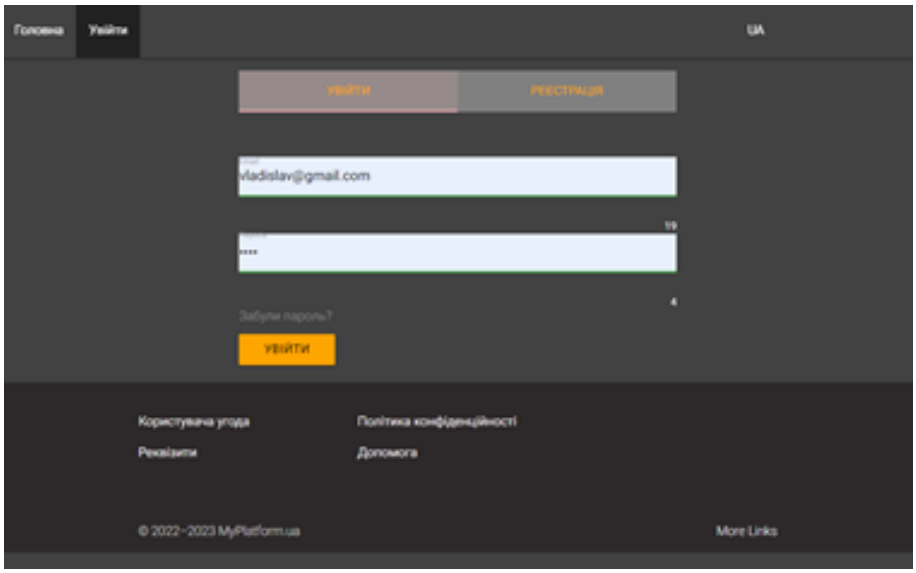


Рис. 1 - Інтерфейс входу користувача в обліковий запис.

2) Користувач повинен мати доступ до бажаного тесту (Рис. 2).

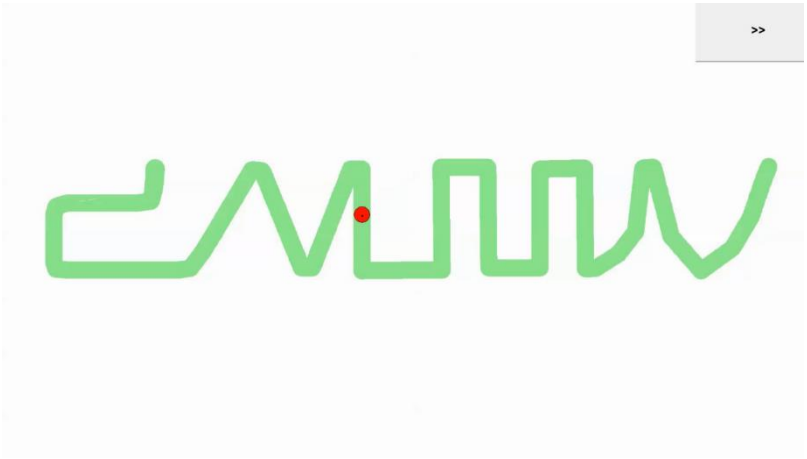


Рис. 2 - Сторінка перегляду тремер тесту.

3) Фахівецю доступа сторінка результатів «тремер тесту». (Рис. 3).

ПІБ: Ткаченко Микола Іванович, Вік: 41, Стать: Ч;

Тест: Тремометрія (координацияметрія)

Дата дослідження: 21.10.2022

Права спіраль:

№	Показник	Права рука	Ліва рука	Асим.
1	Закальна к-ть помилок, шт.	101	86	15
2	Час виконання, с	15,5	15,5	0,0
3	Середян частота помилки, 1/с	6,7	5,7	14,9
4	Сумарна тривалість помилки, мс	681,0	9408,0	27,5
5	Середня тривалість помилки, мс	109,4	67,5	38,3
6	Відносна тривалість помилки, мс	45,5	62,7	27,5
7	Дистанція, мм	5917,0	6133,0	3,5
8	Середня швидкість тремора, мм/с	0,4	0,4	3,5
9	Максимальне відхилення, мм	55,0	71,0	22,5

Висновок:

Кількість помилок на робочих етапах теста більше при роботі правою рукою, а ціна діяльності в цьому режимі нижче, ніж при роботі лівою рукою.

Рис. 3 – Сторінка результатів «тремер тесту».

4) Можливість проводити аналіз даних отриманих після проходження групи тестованих людей методом головних компонентів (Рис. 4).

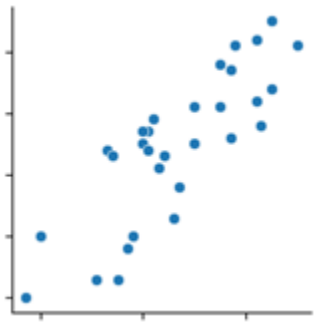


Рис. 4 - Результат аналізу даних.

Виконавець  Каплун В.М.