Міністерство освіти і науки України
Криворізький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерних систем та мереж

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА
за спеціальністю 123 «Комп'ютерна інженерія»

на тему: МЕТОДИ ТА ЗАСОБИ РОЗРОБКИ WEB-ДОДАТКУ «ПІДТРИМКА СОЦІАЛЬНИХ ПРОЕКТІВ»

Проектував _____ І. О.Шевченко

Керівник роботи _____ С. В. Сьомочкина

Нормоконтроль _____ Д. І. Кузнєцов

Завідувач кафедри _____ А. І. Купін

Кривий Ріг
2023

Криворізький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерних систем та мереж

Ступінь вищої освіти магістр
Спеціальність 123 «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри, голова циклової комісії

_____ А. І. Купін

"____" _____ 20__ року

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

_____
(прізвище, ім'я, по батькові)

1. Тема роботи _____
_____
_____

керівник роботи _____,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від "___"_____ 20__року №__

2. Строк подання студентом роботи _____

3. Вихідні дані до роботи _____
_____
_____
_____
_____
_____

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

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

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк виконання етапів роботи	Примітка

Студент _____ _____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка: 91 сторінка, 40 рисунків, 34 таблиці, 17 використаних джерел.

Мета магістерської роботи – розробка інформаційної системи підтримки соціальних проектів.

Магістерська робота складається з чотирьох розділів.

Перший розділ присвячений вибору засобів розробки – хмарному середовищу та СУБД.

Другий розділ спрямовано на розробку структури БД. Створено базу даних «Social_projects» у середовищі PgAdmin та таблиці для розробленої БД.

У третьому розділі розглянуто технологію ASP.NET MVC та розроблено веб-додаток. Також запропоновано складові AWS для розгортання додатку.

У четвертому розділі проведено дослідження оптимальності вибору засобів розробки, виконано порівняння методів суміщеного ідеалу і лінійного програмування.

Зроблено висновки за всіма розділами.

ІНФОРАЦІЙНА СИСТЕМА, БАЗА ДАНИХ, POSTGRESQL, PGADMIN, AWS

Explanatory note: 91 pages, 40 figures, 34 tables, 17 used sources.

The purpose of the master's work is to develop an information system for supporting social projects.

The master's thesis consists of four sections.

The first section is devoted to the choice of development tools - the cloud environment and DBMS.

The second section is aimed at developing the database structure. Created database "Social_projects" in the PgAdmin environment and tables for the developed database.

In the third chapter, ASP.NET MVC technology is considered and a web application is developed. AWS components are also offered for application deployment.

In the fourth chapter, a study of the optimality of the choice of development tools was carried out, a comparison of the combined ideal and linear programming methods was performed.

Conclusions have been drawn for all sections.

INFORMATION SYSTEM, DATABASE, POSTGRESQL, PGADMIN, AWS

ЗМІСТ

ВСТУП

В умовах війни в Україні розвинуті цифрові послуги дуже сталися у пригоді - мобільний застосунок та портал Дія, проекти Дія.Цифрова освіта, Дія.Бізнес тощо. Цифровізація може сприяти вирішенню соціальних проблем, полегшивши доступ до основних послуг у сфері охорони здоров'я (електронна система охорони здоров'я) та освіти (дистанційне навчання), наданню фінансових послуг, прозорості та ефективності діяльності уряду (електронний уряд: система електронних регламентів та реєстрацій), тощо.

Для відбудови зруйнованої інфраструктури України після перемоги вже збираються матеріали щодо знищеного та зіпсованого майна, як особистого так і громадського. Тому доцільною є розробка web-ресурсу, де було б сконцентровано інформацію щодо запропонованих проектів відновлення та відбудови. Фінансування таких об'єктів може здійснюватися не лише державою, але й фізичними та юридичними особами, які мають бажання підтримати обрані рішення. Наприклад, є комплексне IT-рішення для ефективного бюджету участі, яке пропонує базовий пакет послуг для підключення міста або регіону для впливу громадян на розподіл бюджету. Це рішення взяте, як прототип для подальшої роботи.

Актуальність роботи. Більшість сучасних процесів громадського спілкування через умови війни та ковіду переведено в онлайн формат. Сайт для роботи з web-додатком буде створено за допомогою сучасних засобів: безпосередньо веб-сторінка на базі технологій ASP.NET; структуру бази даних для веб-додатку реалізовано в PostgreSQL, керування якою здійснюється засобами PgAdmin; у якості мови програмування обрано C#; у середовищі Visual Studio було виконано інтеграцію бази даних за допомогою ODBC драйвера; для розгортання та підтримки сайту Amazon Web Services (AWS); для підключення і роботи із базою даних web-додатку, що розробляється, буде застосовано службу реляційних баз даних Amazon RDS (Relational Database Service), яка спрощує налаштування, експлуатацію та масштабування розгортання PostgreSQL у хмарі.

Мета і завдання дослідження. Основною метою дослідження є огляд та порівняння сучасних методів і засобів розробки веб-додатків для оптимізації їх вибору.

Головним завданням роботи є розробка веб-додатку, який дозволив би:

1) Оприлюднити пропозиції щодо соціальних проектів, привернувши увагу громадськості та забезпечивши взаємозв'язок авторів з особами, які можуть забезпечити фінансову чи іншу підтримку.

2) Залучити громадськість до вирішення соціальних проблем, шляхом розміщення або підтримки проектних пропозицій, проведення відкритого голосування за такі проекти, а також забезпечення відкритості та прозорості діяльності.

Об’єкт дослідження – процес взаємодії ініціаторів соціальних проектів із громадськістю за допомогою спеціалізованого веб-додатку.

Предмет дослідження – складові для розробки та розгортання веб-додатку для підтримки соціальних проектів.

Методи досліджень. Метод суміщего ідеалу, лінійне програмування.

Наукова новизна одержаних результатів. Запроповано застосувати метод суміщеного ідеалу для оптимізації вибору засобів розробки веб-додатків.

Практичне значення отриманих результатів. Запропоновано алгоритм та програму для реалізації методу суміщеного ідеалу. Розроблено веб-додаток для підтримки соціальних проектів.

Апробація роботи. Апробація досліджень відбувалась на конференції КІСМ 2023 від Криворізького національного університету. Стаття - Застосування AWS-сервісів для розробки web- додатку "Підтримка соціальних проектів" опублікована у збірнику матеріалів конференції.

РОЗДІЛ 1. ВИБІР ЗАСОБІВ РОЗРОБКИ

1.1 AWS vs Azure vs Google Cloud

Існує три основні типи хмарних сервісів: інфраструктура як послуга (IaaS), платформа як послуга (PaaS) та програмне забезпечення як послуга (SaaS). Давайте розберемо види хмарних сервісів. Постачальники хмарних послуг розміщують основне апаратне забезпечення, програмне забезпечення, інфраструктуру, сервери та засоби зберігання, якими клієнти можуть користуватися на основі оплати за користування. Ці сервіси мають високу масштабованість і підтримують швидкість і гнучкість.

Рисунок 1.1 – Лідери хмарних сервісів

Amazon Web Services або AWS є одним із лідерів платформ хмарних обчислень. Будучи дочірньою компанією Amazon, вона пропонує широкий спектр послуг для великих підприємств, окремих розробників, фінансових послуг, ігрових технологій і навіть урядів.

Заснована в 2006 році, вона спочатку пропонувала хмарне сховище Amazon S3 і послуги еластичної обчислювальної хмари (EC2). Тепер можна вибрати з 200+ повнофункціональних послуг.[1]

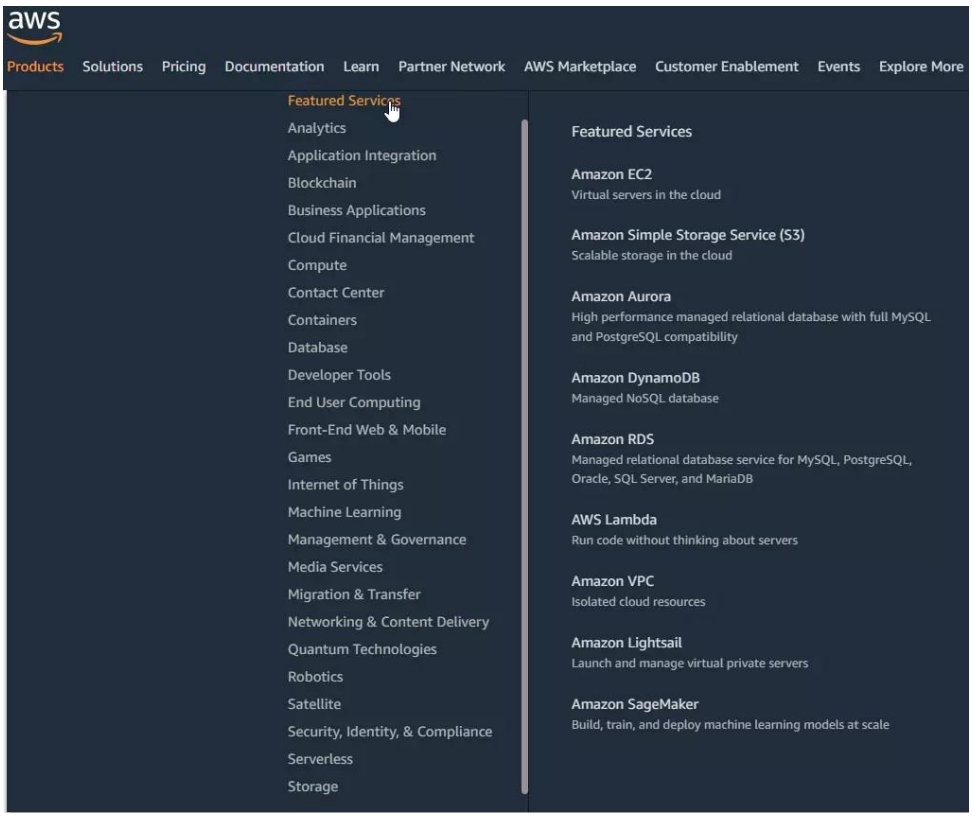


Рисунок 1.2 – Amazon Web Services

Microsoft Azure — це одна з хмарних платформ, що найшвидше розвиваються у світі, яка вже завоювала своє місце лідера ринку. Він розпочав свій шлях у 2010 році, але зараз його клієнтами є велика кількість компаній зі списку Fortune 500. Будучи продуктом Microsoft, Azure пропонує персоналізовані послуги для підприємств, орієнтованих на Microsoft.

Він відомий тим, що обслуговує клієнтів корпоративного рівня та має понад 200 послуг. Окрім служб на основі Windows, Azure підтримує платформи, технології та мови з відкритим кодом.

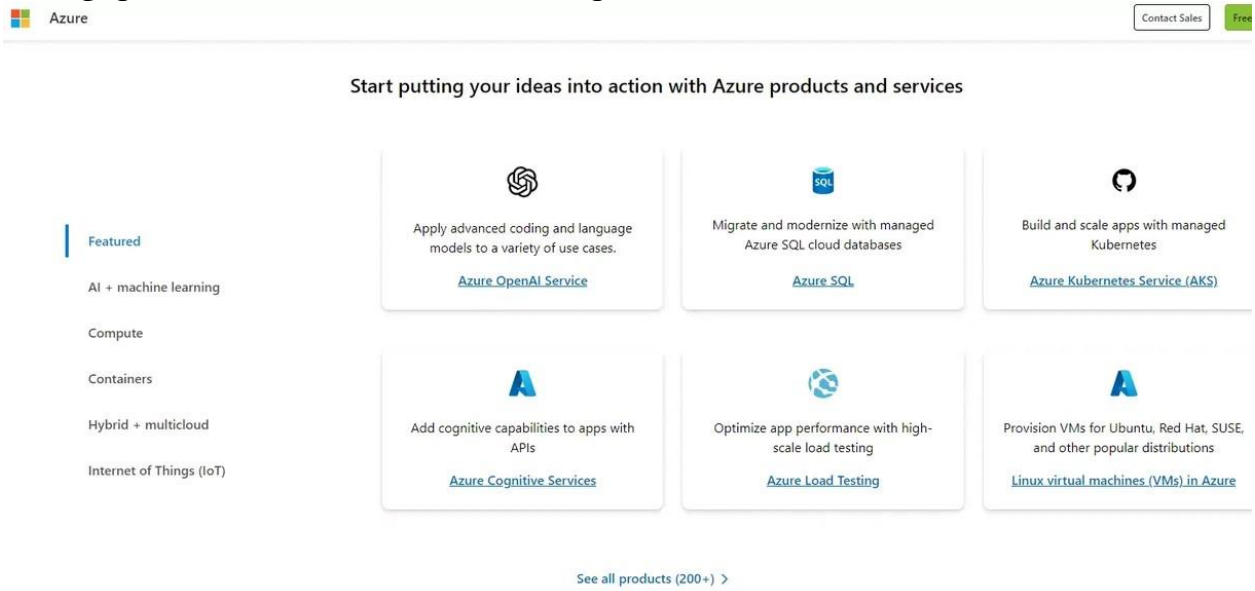


Рисунок 1.3 – Azure

Google Cloud Platform або GCP – це хмарна служба, яку пропонує Google. Він також розпочав свій шлях у 2010 році і наразі пропонує своїм клієнтам понад 100 послуг, у тому числі обчислення, мережі та великі дані. Такі популярні сервіси, як Google Workspace, Chrome OS і корпоративний Android, також є частиною Google Cloud.

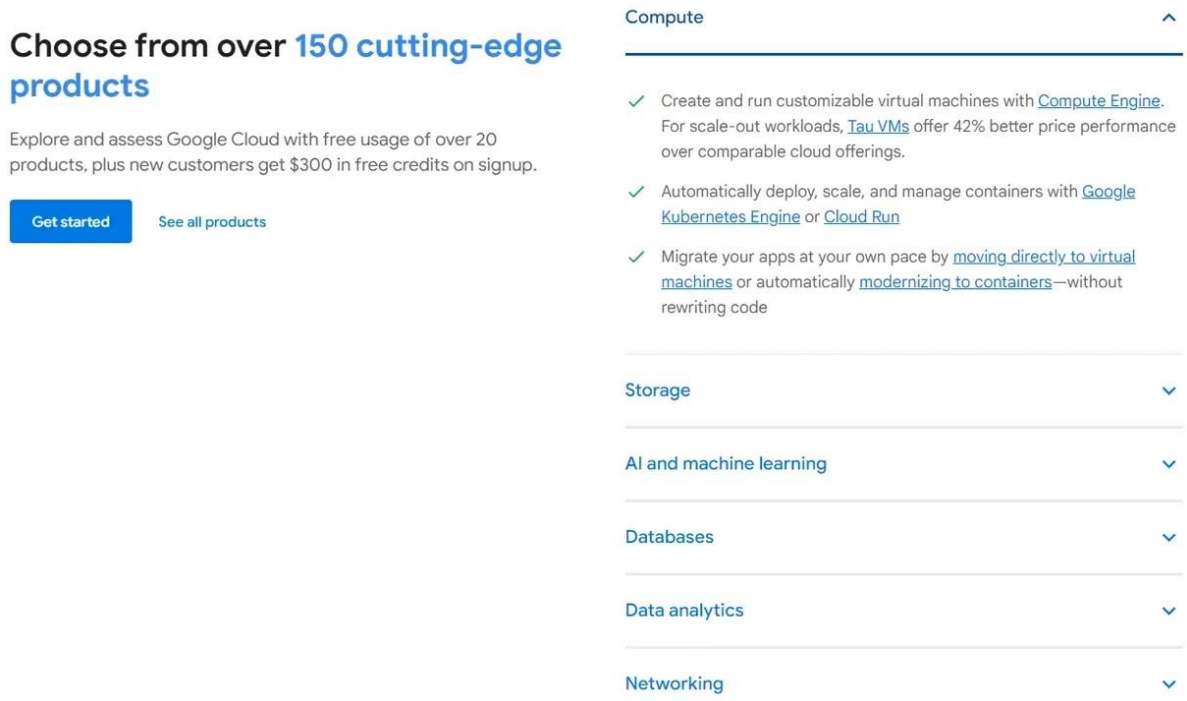


Рисунок 1.4 – Google Cloud Platform

Відповідно до цього звіту Statista, на кінець четвертого кварталу 2022 року Amazon AWS займає найбільшу частку ринку хмарних обчислень з величезною часткою ринку в 32%. Microsoft Azure займає друге місце з часткою 23%, а Google Cloud вдається отримати третю позицію з часткою ринку 10%.[2]

Таблиця 1.1, 1.2, 1.3 – Порівняння AWS, Azure і Google Cloud

AWS	
плюси	**мінуси**
• Більшість доступних послуг, від мережі до робототехніки • Найбільш зрілий • Вважається золотим стандартом у хмарній надійності та безпеці • Більше обчислювальної потужності порівняно з Azure та GCP • Усі основні постачальники програмного забезпечення роблять свої програми доступними на AWS	• Необхідно придбати підтримку Dev/Enterprise • Може вразити новачків великою кількістю послуг і опцій • Порівняно обмежені можливості для гібридної хмари

MICROSOFT AZURE	
плюси	**мінуси**
• Проста інтеграція та міграція для існуючих служб Microsoft • Доступно багато послуг, у тому числі найкращі у своєму класі AI, ML і аналітичні служби • Відносно дешевше для більшості послуг порівняно з AWS і GCP • Чудова підтримка гібридних хмарних стратегій	• Менше пропозицій послуг порівняно з AWS • Особливо орієнтований на корпоративних клієнтів

GCP	
плюси	**мінуси**
• Чудово працює з іншими службами та продуктами Google • Відмінна підтримка контейнерних робочих навантажень • Глобальна оптоволоконна мережа	• Обмежені послуги порівняно з AWS і Azure • Обмежена підтримка корпоративних випадків використання

Ціна хмарної платформи залежить від багатьох факторів: вимоги замовника, використання, використані послуги.

В AWS пропонується тарифний план із оплатою за використання, що означає, що треба сплатити лише за послуги, які використовуються. Це не передбачає жодних довгострокових контрактів чи складного ліцензування. Навіть можна отримати знижку залежно від обсягу, яка дозволить платити менше, коли використовується більше.

Microsoft Azure також пропонує конкурентоспроможні тарифні плани з оплатою за використання відповідно до бюджету та вимог організації. Можна будь-коли скасувати плани та постійно стежити за споживанням хмари та тенденціями витрат.

Як і інші постачальники хмарних послуг, Google Cloud також дозволяє платити лише за те, що використовується. Він дотримується прозорого та інноваційного підходу до ціноутворення, який допомагає заощадити гроші.[1]

Висновок: обираємо для проекту AWS тому, що вони займають найбільшу частину ринку, а також мають багато матеріалів для опанування їх технологій.

1.2 PostgreSQL vs MySQL vs MongoDB

У сучасному світі великих даних MySQL є однією з найвідоміших технологій баз даних. Її часто називають найбільш широко використовуваною базою даних. Відповідно до нещодавнього опитування розробників MySQL є найбільш використовуваним рішенням для зберігання даних . Його вибрали 55,6 відсотка респондентів.

MySQL розроблено як відкритий вихідний код, тому має величезну й активну спільноту. Його створено з використанням мов програмування C і C++. Він сумісний з різними операційними системами, включаючи Linux, OS X, Solaris і Windows.[3]

Переваги MySQL:

- З відкритим кодом і безкоштовно.

MySQL є відкритим вихідним кодом, і, звісно, програми з відкритим кодом із активною спільнотою мають перевагу постійного розвитку та оновлення. Також СУБД має версію для спільноти, яка доступна для безкоштовного встановлення. Однак у безкоштовній версії не так багато функцій, як у платній, але вона все одно ефективна для невеликих проектів.

- Простий і гнучкий.

Архітектура та операції MySQL досить прості. Завдяки кращій безпеці він забезпечує комплексну роботу навіть для великих проектів. Налагодження також можна виконати швидше.

- Продуктивність.

Найновіша версія MySQL була створена швидко та ефективно. Навіть для веб-сайтів електронної комерції база даних є досить перспективною завдяки високій продуктивності та хорошому кешу пам’яті. Він може розглядати кілька (до мільйона) запитів одночасно.

- Широка сумісність.

Незважаючи на свою популярність як бази даних для веб-додатків, MySQL створено для роботи з широким спектром технологій і систем. RDBMS сумісна з різними обчислювальними платформами, включаючи операційні системи на базі Unix, такі як Linux, Mac OS і Windows.

Недоліки MySQL:

- Обмежена діяльність з відкритим кодом.

Хоча MySQL є визнаною базою даних з відкритим кодом, на практиці це вже не так. З моменту переходу під контроль Oracle MySQL тепер має закриті модулі. Розробники з відкритим вихідним кодом розчаровуються в проекті через передбачувані камені спотикання Oracle у розробці та відмову оприлюднювати тестові випадки на дефекти та оновлення безпеки.

- Стандарти SQL дотримуються не повністю.

MySQL базується на програмі Structured Query Language (SQL), яка має свої унікальні правила та стандарти. MySQL не повністю відповідає стандартам SQL. Він радше має свою унікальну структуру та розширення.[6]

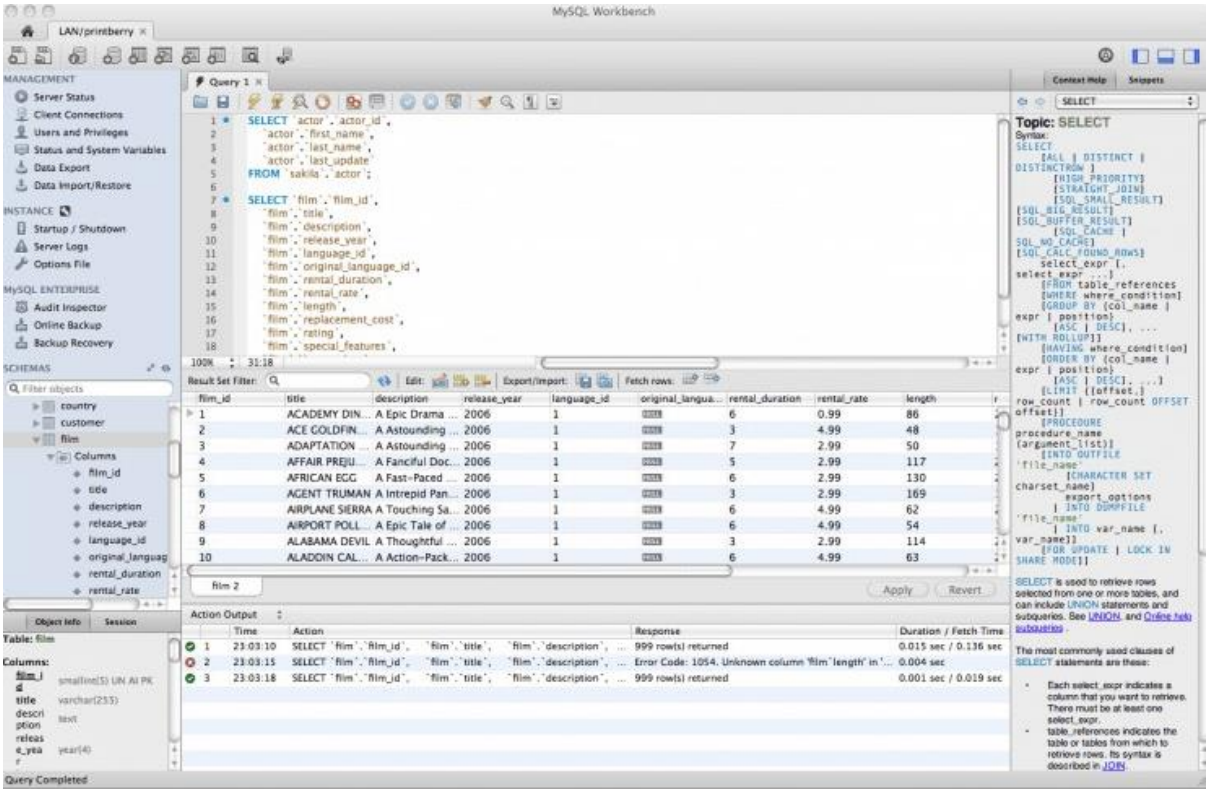


Рисунок 1.5 – середовище СУБД MySQL(Workbench)

PostgreSQL — це ще одна популярна СУБД, яка багато в чому схожа з MySQL. У нещодавньому опитуванні її було обрано другою за популярністю СУБД після MySQL. Розробники також вибрали її другою за зручністю СУБД після Redis і MongoDB.[5]

PostgreSQL — це реляційна система керування базами даних (СУБД), яка створює складніші структури даних шляхом поєднання визначених об’єктів і табличних процедур. Його метою є покращення дотримання та розширення стандартів. Як наслідок, він може впоратися з будь-яким робочим навантаженням, будь то для однієї машини чи складної програми.

Він був розроблений як СУБД з відкритим кодом. Ця СУБД була створена з використанням мови програмування C і сумісна з Microsoft, iOS, Android, NetBSD, Linux та різними іншими платформами.

Переваги PostgreSQL:

- Покращена масштабованість.

PostgreSQL оптимізовано для швидкої та постійної масштабованості, і ця якість дає йому перевагу над MySQL. Завдяки високим можливостям масштабування PostgreSQL можна розгортати у великих корпораціях і для проектів з високими вимогами.

- Повністю відкритий.

PostgreSQL, мабуть, має найкращу та найактивнішу ідеологію відкритого коду. Він пропонує організаційну ефективність і безмежні можливості для зростання. Користувачі PostgreSQL також можуть брати активну участь у спільноті, публікуючи та ділячись помилками та проблемами.

- Широка підтримка спеціальних даних.

Відомо, що PostgreSQL має хорошу робочу підтримку для модулів даних noSQL. За замовчуванням він підтримує широкий спектр типів даних, включаючи JSON, XML, H-Store тощо.

- Ефективна співпраця з іншими інструментами.

Існує широкий набір засобів керування базами даних, які доповнюють СУБД. На жаль, багато СУБД мають обмежені можливості інтеграції. PostgreSQL вирішує цю проблему шляхом легкої та ефективної інтеграції з багатьма іншими незалежними інструментами.[4]

Недоліки PostgreSQL:

- Низька швидкість.

Люди, які використовують Postgres, стикаються з різними проблемами продуктивності, а також проблемами відновлення резервної копії. Часто ви матимете повільний запит і побачите, що продуктивність середовища вашої бази даних погіршилася. Через структуру реляційної бази даних Postgres повинен починати з першого рядка, а потім сканувати всю таблицю, щоб визначити відповідні дані під час пошуку запиту. Як наслідок, це повільніше, особливо коли в рядках і стовпцях таблиці багато даних із багатьма змінними для порівняння.

- Неповна документація.

Незважаючи на те, що PostgreSQL має велику спільноту та пропонує чудову допомогу, документація залишається нерегулярною та неповною. Через розрізненість спільноти PostgreSQL документація для всіх функцій Postgre не відповідає однаковим стандартам.

- Немає інструментів звітності чи аудиту.

Існує більший шанс, що інженери пропустять помилку PostgreSQL через відсутність інструментів аудиту. Використання цього засобу вимагає обережності.

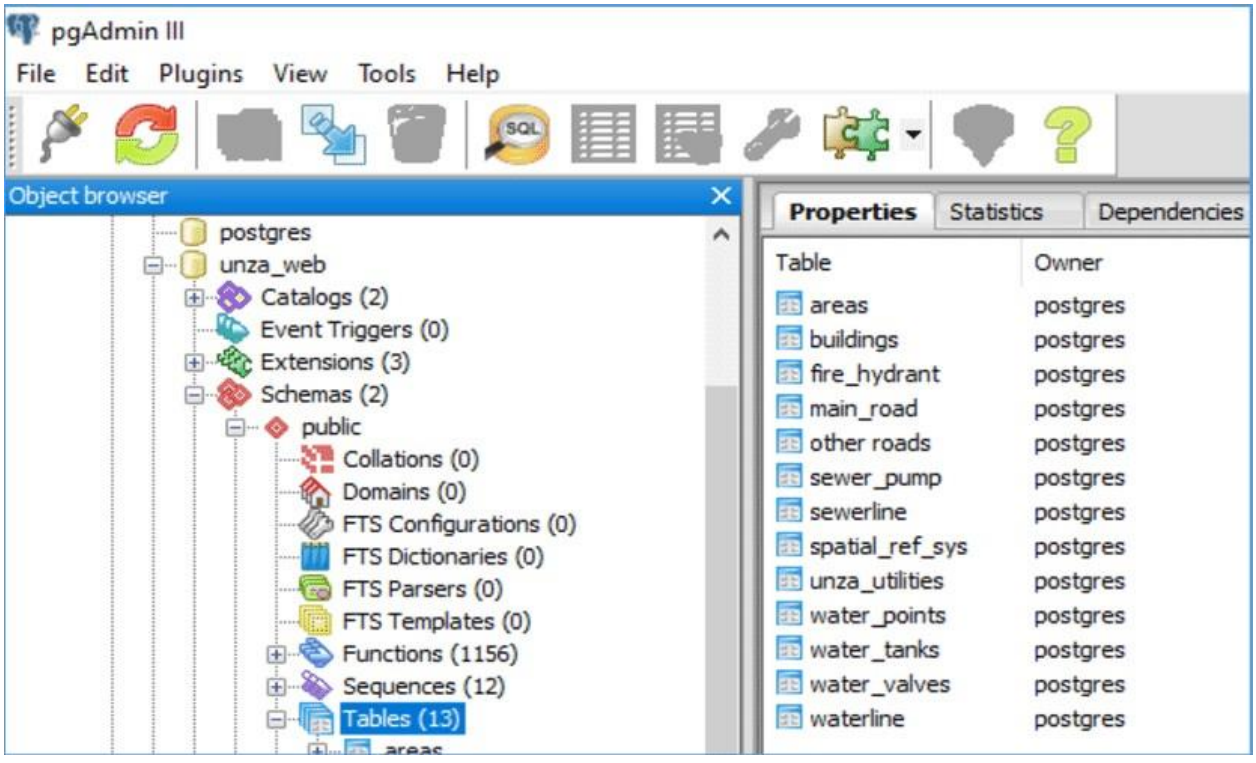


Рисунок 1.6 – середовище СУБД PostgreSQL(PgAdmin)

MongoDB — це нереляційна система керування зберіганням даних, яка обробляє та зберігає великі обсяги та різні типи даних, використовуючи гнучкі документи, а не таблиці та рядки. MongoDB не вимагає реляційної системи зберігання даних, оскільки це рішення NoSQL; тому він пропонує розширюваний формат зберігання даних, який дозволяє легко керувати декількома типами даних.

MongoDB розробляє та керує MongoDB.Inc, яка була вперше випущена в лютому 2009 року. Вона широко використовується, оскільки забезпечує гнучкість завдяки підтримці всіх популярних мов програмування.[5]

Переваги MongoDB:

- Оперативна ефективність.

Швидкі та прості операції є однією з переваг MongoDB, отриманих завдяки своїй природі Non SQL. Дані можна миттєво вводити, зберігати та отримувати з бази даних без необхідності подальшої перевірки. Він надає пріоритет використанню оперативної пам’яті, тому можна швидко отримати доступ до записів без шкоди для якості даних.

- Підтримка кількох мов.

Багатомовна підтримка MongoDB є однією з його найкращих функцій. MongoDB було оновлено кількома версіями та все ще розробляється, з підтримкою драйверів для відомих мов програмування, таких як Python, PHP, Ruby, C++, Scala, JavaScript тощо.

- Взаємодія з різними форматами даних.

MongoDB добре працює з різними програмами адміністрування баз даних, включаючи SQL і NoSQL.

Недоліки MongoDB:

- Потрібен великий обсяг пам'яті.

Операції MongoDB вимагають багато пам’яті для ефективності. Ця СУБД також запам’ятовує всі імена ключів для кожної пари значень, сприяючи використанню пам’яті.

- Інтерпретація на різні мови запитів є складним процесом.

Оскільки MongoDB не був розроблений для виконання реляційних моделей даних із самого початку, продуктивність може постраждати в таких ситуаціях. Крім того, перетворення запитів SQL у MongoDB потребує додаткових кроків для використання механізму, що потенційно може спричинити затримки.

- Вкладання документів для більш ніж 100 рівнів неможливе.

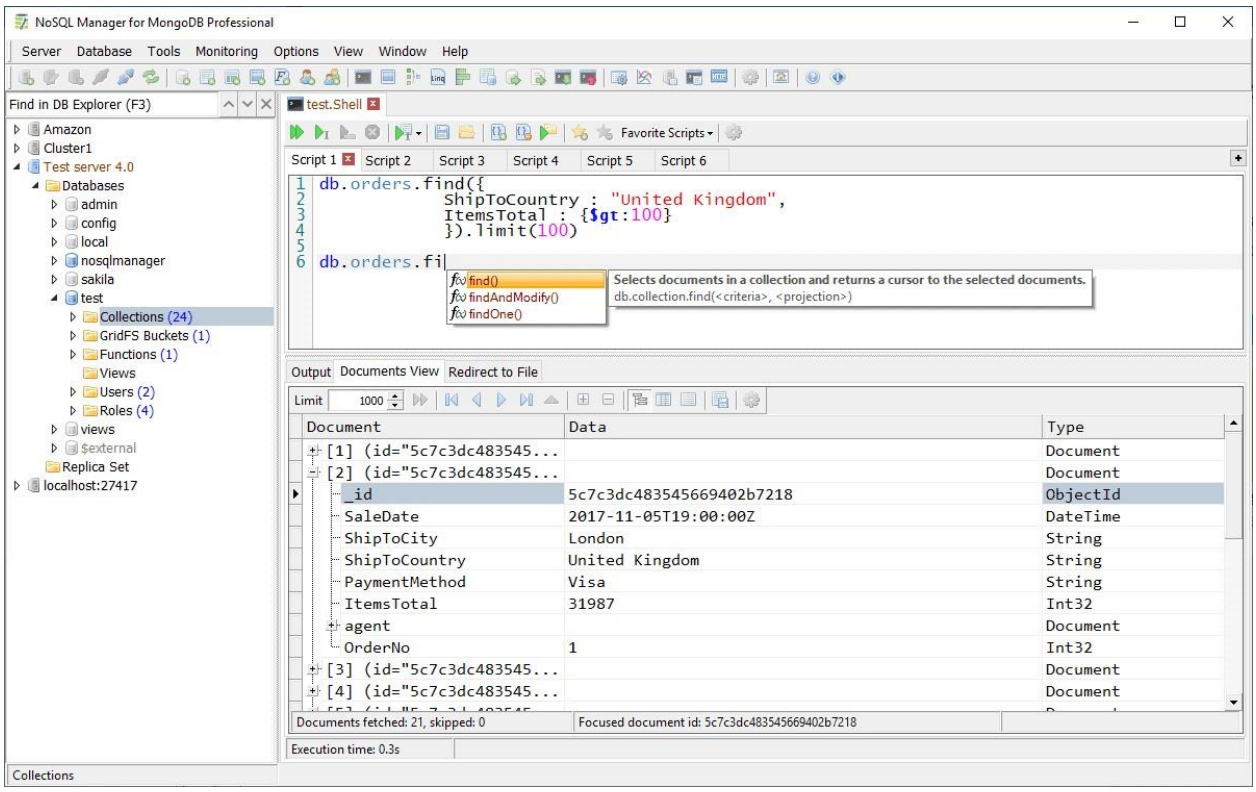


Рисунок 1.7 – середовище СУБД MongoDB (NoSQL)

Висновки за розділом

Для нашого проекту обираємо PostgreSQL тому, що це є сучасним рішенням, яке суміщує, як переваги реляційних баз даних, так і підтримку JSON. Для керування базою даних використовуємо PgAdmin. Розроблена база даних має в своєму складі таблиці: користувачі, проекти, категорії, статуси, результати голосувань та бюджет.

РОЗДІЛ 2. ПРОЕКТУВАННЯ БАЗИ ДАНИХ

2.1 Вибір СУБД та іншого програмного забезпечення

Для реалізації вищерозглянутої системи зручно використовувати реляційну СУБД, наприклад PostgreSQL.

PostgreSQL не просто реляційна, а об'єктно-реляційна СУБД. Це дає йому деякі переваги над іншими базами даних SQL з відкритим вихідним кодом, такими як SQLite, MS SQL, MySQL, MariaDB і Firebird.

Фундаментальна характеристика об'єктно-реляційної бази даних - це підтримка об'єктів користувача та їх поведінки, включаючи типи даних, функції, операції, домени та індекси. Це робить обрану СУБД неймовірно гнучкою та надійною. Серед іншого, вона може створювати, зберігати та витягувати складні структури даних.[6]

Існує великий список типів даних, які підтримує PostgreSQL. Крім числових, з плаваючою точкою, текстових, булевих та інших очікуваних типів даних (а також безлічі їх варіацій), PostgreSQL може похвалитися підтримкою uuid, грошового, перерахованого, геометричного, бінарного типів, мережевих адрес, бітових рядків, текстового пошуку, xml, масивів, композитних типів та діапазонів, а також деяких внутрішніх типів для ідентифікації об'єктів та розташування логів. MySQL, MariaDB і Firebird теж мають деякі з цих типів даних, але тільки PostgreSQL підтримує їх усі.

Оскільки PostgreSQL є об'єктно-реляційною базою даних, масиви значень можуть зберігатися для більшості існуючих типів даних. Зробити це можна шляхом додавання квадратних дужок до специфікації типу даних стовпця або за допомогою виразу ARRAY. Розмір масиву може бути заданий, але це необов'язково. MySQL, MariaDB і Firebird не мають таких можливостей. Щоб зберігати такі масиви значень у традиційних реляційних базах даних, доведеться використовувати обхідний шлях та створювати окрему таблицю з рядками для кожного з значень масиву.

Підтримка JSON у PostgreSQL дозволяє вам перейти до зберігання schema-less даних у SQL базі даних. Це може бути корисним, коли структура даних потребує певної гнучкості: наприклад, якщо в процесі розробки структура все ще змінюється або невідомо, які поля міститиме об'єкт даних.

PostgreSQL має безліч можливостей. Створений з використанням об'єктно-реляційної моделі, він підтримує складні структури та широкий спектр вбудованих та визначених користувачем типів даних. Він забезпечує розширену ємність даних і має багато засобів для збереження цілісності даних.

Для кожного з розглянутих блоків потрібно розробити таблиці у складі бази даних.

Створимо базу даних «Social_projects» у середовищі PgAdmin.

PgAdmin - це платформа з відкритим вихідним кодом для адміністрування та розробки для PostgreSQL і пов'язаних з нею систем управління базами даних. Платформа написана на Python та jQuery і підтримує всі функції PostgreSQL. Можливо використовувати PgAdmin для будь-яких операцій, починаючи з запису базових SQL-запитів і закінчуючи моніторингом розроблених баз даних і налаштування просунутих архітектур баз даних.

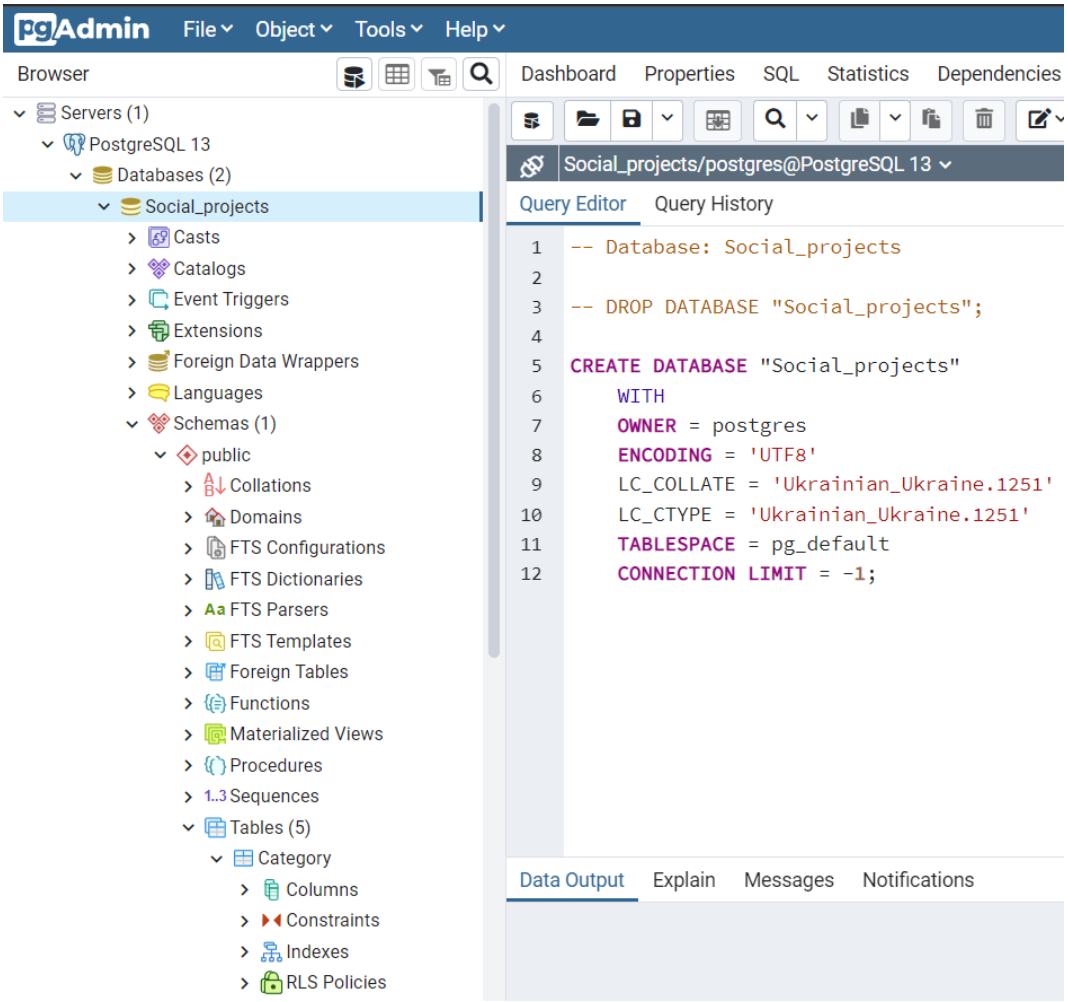


Рисунок 2.1 – Середовище PgAdmin

PostgreSQL веде свій «родовід» від некомерційної СУБД Postgres, розробленої, як і багато open-source проектів, в Каліфорнійському університеті в Берклі. До розробки Postgres, що почалася в 1986 році, мав безпосереднє відношення Майкл Стоунбрейкер, керівник більш раннього проекту Ingres, на той момент вже придбаного компанією Computer Associates. Сама назва «Postgres» розшифровувалась як Post Ingres, відповідно, при створенні Postgres були застосовані багато вже раніше зроблених напрацювань.[5]

Стоунбрейкер та його студенти розробляли нову СУБД протягом восьми років із 1986 по 1994 рік. За цей період до синтаксису було введено процедури, правила, типи користувача і багато інших компонентів.

Робота не пройшла даремно - в 1995 році розробка знову розділилася: Стоунбрейкер використав отриманий досвід у створенні комерційної СУБД

Illustra, що просувалася його власною однойменною компанією, а його студенти розробили нову версію Postgres — Postgres95, в якій мова запитів POSTQUEL — спадщина Ingres була замінена на SQL.

У цей момент розробка Postgres95 була виведена за межі університету та передана команді ентузіастів. З цього моменту СУБД отримала ім'я, під яким вона відома і розвивається зараз.

Таблиця 1.4 - На даний момент у PostgreSQL є такі обмеження:

Максимальний розмір бази даних	Немає обмежень
Максимальний розмір таблиці	32 Тбайт
Максимальний розмір запису	1,6 Тбайт
Максимальний розмір поля	1 Гбайт
Максимум записів у таблиці	Немає обмежень
Максимум полів у записі	250-1600, залежно від типів полів
Максимум індексів у таблиці	Немає обмежень

Швидкість роботи, взагалі кажучи, не є основною причиною користування реляційними СУБД. Більше того, перші реляційні бази працювали повільніше за своїх попередників. Вибір цієї технології був викликаний швидше:

- можливістю покласти підтримку цілісності даних на СУБД;
- незалежністю логічної структури даних від фізичної.

Ці особливості дозволяють сильно спростити написання застосунків, але вимагають для реалізації додаткових ресурсів.

Таким чином, перш ніж шукати відповідь на запитання «як змусити РСУБД працювати швидше?», слід відповісти на запитання «чи немає більш відповідного засобу для вирішення завдання, ніж РСУБД? Іноді використання іншого засобу потребує менше зусиль, ніж налаштування продуктивності.

За замовчуванням PostgreSQL налаштований таким чином, щоб він міг бути запущений практично на будь-якому комп'ютері і не надто заважав при цьому роботі інших додатків.[8]

Налаштування за замовчуванням підходять тільки для наступного використання: з ними можливо перевірити, чи працює установка PostgreSQL, створити тестову базу рівня записника та потренуватися писати до неї запити.

Якщо потрібно розробляти (а тим більше запускати в роботу) реальні програми, то налаштування доведеться радикально змінити. У дистрибутиві PostgreSQL, на жаль, файли з «рекомендованими» настройками не поставляються. Взагалі-то кажучи, такі файли створити дуже складно, т.к. оптимальні налаштування конкретної установки PostgreSQL будуть визначатися:

- конфігурацією комп'ютера;
- обсягом та типом даних, що зберігаються в базі;
- відношенням числа запитів на читання та запис;
- тим, чи запущені інші вимогливі до ресурсів процеси (наприклад, веб-сервер).

Розглянемо рекомендовані значення параметрів, що впливають на продуктивність СУБД. Ці параметри зазвичай встановлюються в конфігураційному файлі postgresql.conf і впливають на всі бази поточної установки.

Використовувана пам'ять

PostgreSQL не читає дані безпосередньо з диска і не пише їх одразу на диск. Дані завантажуються до загального буфера сервера, що знаходиться в пам'яті, що розділяється, серверні процеси читають і пишуть блоки в цьому буфері, а потім зміни вже скидаються на диск.

Якщо процесу потрібен доступ до таблиці, він спочатку шукає потрібні блоки у загальному буфері. Якщо блоки присутні, то він може продовжувати роботу, якщо ні - робиться системний виклик для їх завантаження. Завантажуватися блоки можуть як із файлового кеша ОС, так і з диска, і ця операція може виявитися дуже дорогою.

Якщо обсяг буфера недостатній для зберігання часто використовуваних робочих даних, то вони постійно писатимуться і читатимуться з кешу ОС або з диска, що вкрай негативно позначиться на продуктивності.

У той же час не слід встановлювати це значення занадто великим: це НЕ вся пам'ять, яка потрібна для роботи PostgreSQL, це тільки розмір розділеної між процесами PostgreSQL пам'яті, яка необхідна для виконання активних операцій. Вона повинна займати меншу частину оперативної пам'яті комп'ютера, оскільки PostgreSQL покладається на те, що операційна система кешує файли, і не старається дублювати цю роботу. Крім того, чим більше пам'яті буде віддано під буфер, тим менше залишиться операційній системі та іншим застосункам.

На жаль, щоб знати точну кількість shared_buffers, потрібно врахувати кількість оперативної пам'яті комп'ютера, розмір бази даних, число з'єднань і складність запитів, так що краще скористаємося кількома простими правилами налаштування.

На виділених серверах корисним об'ємом для shared_buffers буде значення 1/4 пам'яті у системі. Якщо у вас великі активні порції бази даних, складні запити, велика кількість одночасних з'єднань, тривалі транзакції, доступний великий обсяг оперативної пам'яти або більше процесорів, то

можна піднімати це значення та моніторити результат, щоб не призвести до «деградації» (падіння) продуктивності. Виділивши дуже багато пам'яті для бази даних, ми можемо отримати погіршення продуктивності, оскільки PostgreSQL також використовує кеш операційної системи (збільшення даного параметра більше 40% оперативної пам'яті може давати «нульовий» приріст продуктивності).

Для тонкого налаштування параметра встановіть для нього велике значення і протестуйте базу при звичайному навантаженні. Перевіряйте використання пам'яті, що розділяється, за допомогою ipcs або інших утиліт (наприклад, free або vmstat). Рекомендоване значення параметра буде приблизно в 1,2 –2 рази більше, ніж максимум використаної пам'яті. Зверніть увагу, що пам'ять під буфер виділяється при запуску сервера, та її обсяг при роботі не змінюється. Потрібно врахувати також, що налаштування ядра операційної системи можуть не дати виділити великий обсяг пам'яті (для версії PostgreSQL <9.3).

Також слід пам'ятати, що на 32 бітній системі (Linux) кожен процес лімітований в 4 ГБ адресного простору, де хоча б 1 ГБ зарезервований ядром. Це означає, що незалежно, скільки на машині пам'яті, кожен PostgreSQL інстанс зможе звернутися максимум до 3 ГБ пам'яті. Отже максимум для shared_buffers в такій системі — 2–2.5 ГБ.[9]

На Windows, великі значення для shared_buffers не настільки ефективні, як на Linux системах, і остаточно найкращі результати можна буде отримати, якщо тримати це значення відносно невелике (від 64 МБ до 512 МБ) і використовувати кеш системи замість нього.

Пам'ять для сортування результатів запиту: work_mem

work_mem параметр визначає максимальну кількість операційної пам'яті, яке може виділити одна операція сортування, агрегації та ін. Ця пам'ять не розділяється, work_mem виділяється окремо на кожну операцію (від одного до кількох разів на один запит). Доцільне значення параметра визначається наступним чином: кількість доступної оперативної пам'яті (після того, як із загального обсягу відняли пам'ять, потрібну для інших програм, і shared_buffers) ділиться на максимальну кількість одночасних запитів помножену на середнє число операцій у запиті, які потребують пам'яті.

Якщо обсяг пам'яті недостатній для сортування деякого результату, то серверний процес буде використовувати тимчасові файли.

Об'єм пам'яті задається параметром work_mem у файлі postgresql.conf.

Одиниця виміру параметра - 1 кБ. Значення за замовчуванням - 1024.

Як початкове значення для параметра можете взяти 2–4% доступної пам'яті. Для веб-застосунків зазвичай встановлюють низькі значення work_mem, тому що запитів багато, але вони прості, зазвичай вистачає від 512 до 2048 кБ. З іншого боку, додатки для підтримки прийняття рішень з сотнями рядків у кожному запиті та десятками мільйонів стовпців у таблицях фактів часто вимагають work_mem порядку 500 МБ. Для баз даних, які використовуються і так, і так, цей параметр можна встановлювати для кожного

запиту індивідуально. Наприклад, при пам'яті 1-4 ГБ рекомендується встановлювати 32-128 МБ.

Максимальна кількість клієнтів: max_connections

Параметр max_connections встановлює максимальну кількість клієнтів, які можуть підключитися до PostgreSQL. Бо для кожного клієнта потрібно виділяти пам'ять (work_mem), то цей параметр передбачає максимально можливе використання пам'яті для всіх клієнтів. Як правило, PostgreSQL може підтримувати декілька сотень підключень, але створення нового є дорогою операцією.

Тому, якщо потрібні тисячі підключень, краще використовувати пул підключень (окрема програма або бібліотека для продукту, що використовує основу).

Пам'ять для роботи команди VACUUM: maintenance_work_mem

Цей параметр ставить об’єм пам'яті, використовуваний командами VACUUM, ANALYZE, CREATE INDEX, та додавання зовнішніх ключів. Щоб операції виконувались максимально швидко, потрібно встановлювати цей параметр тим вище, чим більший розмір таблиць у вашій базі даних. Непогано б встановлювати його значення від 50 до 75% розміру найбільшої таблиці або індексу або, якщо точно визначити неможливо - можна від 32 до 256 МБ. Слід встановлювати більше значення, ніж для work_mem. Занадто великі значення не рекомендовані. Наприклад, при пам'яті 1-4 ГБ рекомендується встановлювати 128-512 МБ.

Великі сторінки: huge_pages

У PostgreSQL, починаючи з версії 9.4, з'явилася підтримка великих сторінок. В ОС Linux робота з пам'яттю ґрунтується на зверненні до сторінок, розмір яких дорівнює 4кБ (насправді залежить від платформи, можна перевірити через getconf (PAGE_SIZE). Так ось, коли обсяг пам'яті більший за кілька десятків, а то й сотні гігабайт, керувати нею стає складніше, збільшуються накладні витрати на адресацію пам'яті та підтримання сторінкових таблиць. Для полегшення життя і були придумані великі сторінки, розмір яких може бути 2MB, а то й 1GB. За рахунок використання великих сторінок можна отримати відчутний приріст швидкості роботи та збільшення чутливості у додатках, які активно працюють із пам'яттю.

Взагалі запустити PostgreSQL з підтримкою великих сторінок можливо було й раніше, за допомогою libhugetlbfs. Однак тепер є вбудована підтримка. Отже, щоб налаштувати та запустити PostgreSQL за допомогою великих сторінок, для початку слід переконатися, що ядро підтримує великі сторінки. Перевіряємо конфіг ядра щодо наявності опцій CONFIG_HUGETLBFS та CONFIG_HUGETLB_PAGE.

```
Line 1  $ grep HUGETLB /boot/config-$(uname -r)
    -   CONFIG_CGROUP_HUGETLB=y
    -   CONFIG_HUGETLBFS=y
    -   CONFIG_HUGETLB_PAGE=y
```

У разі відсутності цих опцій, нічого не запрацює і ядро потрібно перезбирати. Очевидно, що знадобиться PostgreSQL версії не нижче 9.4.

За підтримку великих сторінок відповідає параметру huge_page, який може приймати три значення:

off - не використовувати великі сторінки,

on -використовувати великі сторінки,

try — спробувати використати великі сторінки та у разі недоступності відкотитися на використання звичайних сторінок.

Значення try використовується за умовчанням та є безпечним варіантом. У випадку on, PostgreSQL не запуститься, якщо великі сторінки не визначені у системі (або їх недостатньо).

Після перезапуску бази з параметром try потрібно включити підтримку великих сторінок у системі (за замовчуванням вони задіяні).

Розрахунок сторінок приблизний і тут слід спиратися на те, скільки приблизно пам'яті можна виділити під потреби СУБД.

Значення вимірюється в сторінках розміром 2Mb, якщо потрібно виділити 16GB, це буде 8000 сторінок.[11]

Офіційна документація пропонує спиратися на значення VmPeak зі status файлу, який розміщений в /proc/PID/ директорії, відповідної номеру процесу postmaster. VmPeak як слід з назви, це пікове значення використання віртуальної пам'яті. Цей варіант дозволяє визначити мінімальну планку, від якої слід відштовхуватися, але такий спосіб визначення теж носить випадковий характер.

Включаємо підтримку huge pages у системі:

```
Line 1  $ head -1 /var/lib/pgsql/9.5/data/postmaster.pid
    -   3076
    -   $ grep ^VmPeak /proc/3076/status
    -   VmPeak:  4742563 kB
    5   $ echo $((4742563 / 2048 + 1))
    -   2316
    -   $ echo 'vm.nr_hugepages = 2316' >> /etc/sysctl.d/30-
            postgresql.conf
    -   $ sysctl -p --system
```

В ОС Linux є також система з менеджменту пам'яті під назвою "Transparent HugePages", яка включена за замовчуванням. Вона може викликати проблему при роботі з huge pages для PostgreSQL, тому рекомендується вимикати цей механізм.

Відключаємо Transparent HugePages:

```
$ echo never > /sys/kernel/mm/transparent_hugepage/defrag
$ echo never > /sys/kernel/mm/transparent_hugepage/enabled
```

Після цього перезапускаємо PostgreSQL і дивимося використання великих сторінок:

```
$ grep ^HugePages /proc/meminfo
HugePages_Total:    2316
HugePages_Free:     2301
HugePages_Rsvd:      128
HugePages_Surp:        0
```

Інші налаштування:

- temp_buffers — буфер під тимчасові об'єкти, в основному для тимчасових таблиць. Можна встановити близько 16 МБ;
- max_prepared_transactions - кількість транзакцій, що одночасно підготовлюються (PREPARE TRANSACTION). Можна залишити по замовчуваню - 5;
- vacuum_cost_delay - якщо таблиці великі, і виробляється багато одночасних операцій запису, може стати в нагоді функція, яка зменшує витрати на I/O для VACUUM, розтягуючи його за часом. Щоб увімкнути цю функціональність, потрібно підняти значення vacuum_cost_delay вище 0. Використовуйте розумну затримку від 50 до 200 мс. Для більш тонкого налаштування потрібно підвищити vacuum_cost_page_hit і знизити vacuum_cost_page_limit. Це послабить вплив VACUUM, збільшивши час його виконання. У тестах із паралельними транзакціями було отримано, що при значеннях delay - 200, page_hit - 6 і limit -100 вплив VACUUM зменшився більш ніж на 80%, але його тривалість збільшилася втричі;
- max_stack_depth — спеціальний стек для сервера, який в ідеалі повинен збігатися з розміром стека, виставленим у ядрі ОС. Установка більшого значення, ніж у ядрі, може призвести до помилок. Рекомендується встановлювати 2-4 МБ;
- max_files_per_process — максимальна кількість файлів, що відкриваються процесом і його підпроцесами в один момент часу. Потрібно зменшити даний параметр, якщо в процесі роботи спостерігається повідомлення «Too many open files».

2.2 Розробка структури бази даних(таблиць та зв’язків між ними)

Створюємо таблиці БД для підтримки соціальних проектів:

data_project	Таблиця властивостей проекту
project_name	- назва проекту
project_id	- номер проекту
creation_date	- дата створення
total_budget	- загальний бюджет
category_id	- категорія проекту
status_id	- номер статусу

user_data	Таблиця особистих даних користувача
user_name	- ім’я користувача
passport_id	- номер паспорта
registration_date	- дата регістрації
e-mail	- особиста пошта
date_of_birth	- дата народження

voices	Таблиця результатів голосування користувачів
project_id	- номер проекту
project_id	- категорія проекту
passport_id	- номер паспорта
voting_date	- дата голосування за проект

category	Таблиця категорій проекту
category_name	- назва категорії
category_id	- номер категорії

status	Таблиця статусу проекту
status_name	- назва статусу
status_id	- номер статусу

budget	Таблиця бюджету проекту
budget_id	- номер бюджету
project_id	- номер проекту
allocation	- призначення коштів
implementation	- відмітка про виконання

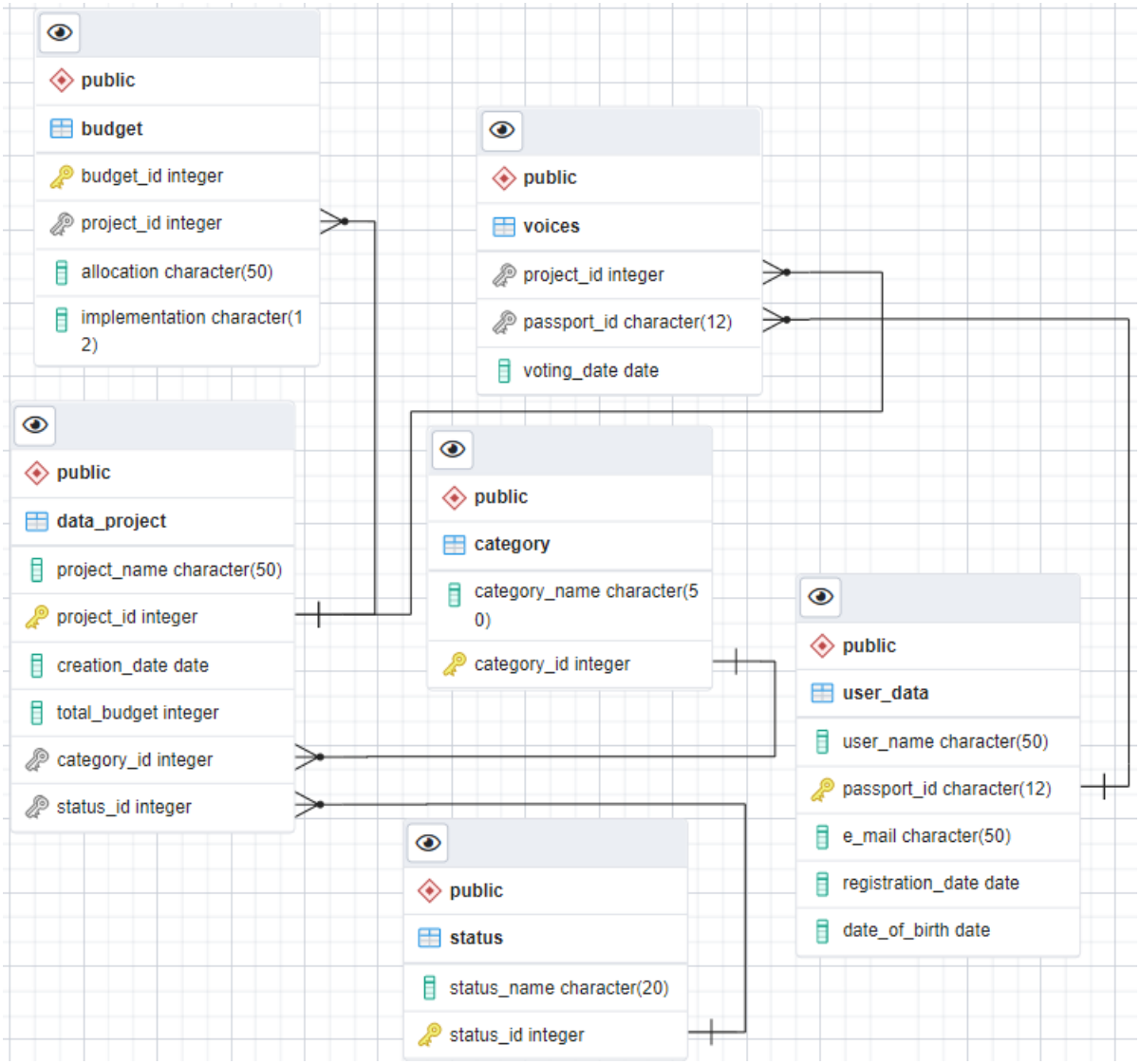


Рисунок 2.2 - Схема бази даних

Запити для створення відповідних таблиць:

Таблиця характеристик проекту

```
CREATE TABLE IF NOT EXISTS public.data_project
(
    project_name character(50) COLLATE pg_catalog."default",
    project_id integer NOT NULL,
    creation_date date,
    total_budget integer,
    category_id integer NOT NULL,
    status_id integer NOT NULL,
    CONSTRAINT data_project_pkey PRIMARY KEY (project_id),
    CONSTRAINT category_id FOREIGN KEY (category_id)
        REFERENCES public.category (category_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT status_id FOREIGN KEY (status_id)
        REFERENCES public.status (status_id) MATCH SIMPLE
        ON UPDATE NO ACTION
```

```
        ON DELETE NO ACTION
)
```

Таблиця даних користувача

```
REATE TABLE IF NOT EXISTS public.user_data
(
    user_name character(50) COLLATE pg_catalog."default",
    passport_id character(12) COLLATE pg_catalog."default" NOT NULL,
    e_mail character(50) COLLATE pg_catalog."default" NOT NULL,
    registration_date date,
    date_of_birth date,
    CONSTRAINT user_data_pkey PRIMARY KEY (passport_id)
)
```

Таблиця голосування користувачів

```
CREATE TABLE IF NOT EXISTS public.voices
(
    project_id integer NOT NULL,
    passport_id character(12) COLLATE pg_catalog."default" NOT NULL,
    voting_date date,
    CONSTRAINT passport_id FOREIGN KEY (passport_id)
        REFERENCES public.user_data (passport_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT project_id FOREIGN KEY (project_id)
        REFERENCES public.data_project (project_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)
```

Категорія проекту

```
CREATE TABLE IF NOT EXISTS public.category
(
    category_name character(50) COLLATE pg_catalog."default",
    category_id integer NOT NULL,
    CONSTRAINT category_pkey PRIMARY KEY (category_id)
)
```

Статус проекту

```
CREATE TABLE IF NOT EXISTS public.status
(
    status_name character(20) COLLATE pg_catalog."default",
    status_id integer NOT NULL,
    CONSTRAINT status_pkey PRIMARY KEY (status_id)
)
```

2.3. Розробка запитів до БД та її наповнення

Внесемо інформацію до створених таблиць:

Запит на перегляд інформації у таблиці властивостей проекту та результат його виконання:

Data Output | Explain | Messages | Notifications

	Name_project character (50)	Number_project [PK] integer	Creation_date date	Budget integer	Category character (50)
1	Шкільний коридор для веселих перерв	67	2021-09-01	378659	8
2	Спорт для людей літнього віку. Час-дій	141	2021-09-04	293020	3
3	Волейбольно-баскетбольний майданчик КНВК № 81	31	2021-08-26	489996	8
4	Здорова дитина - майбутнє України	113	2021-09-03	489417	10
5	Улюблений Саксаганський: сквер з широким серцем	108	2021-09-03	489300	6

Запит на перегляд інформації у таблиці особистих даних користувача та результат його виконання:

Data Output | Explain | Messages | Notifications

	Name character (50)	Registration_date date	Passport_ID [PK] integer	E-mail character (50)	Date_of_birth date
1	Шевченко Ігор	2018-11-01	276337	igorsheva44@bigmir.net	2000-12-26
2	Макаревич Андрій	2019-03-10	236455979	makarevich.a23@gmail.com	2001-07-21
3	Послушной Костянтин	2020-11-07	753922567	poslushnoi456@gmail.com	1998-09-17
4	Сидоренко Максим	2021-11-09	342553968	sydorenkom546@gmail.com	2001-04-13

Запит на перегляд інформації у таблиці голосування користувачів та результат його виконання:

Data Output | Explain | Messages | Notifications

	Number_project integer	Category integer	Passport_ID integer	Voting_date date
1	31	8	276337	2021-10-29
2	141	3	276337	2021-10-29
3	67	8	276337	2021-10-29
4	113	10	276337	2021-10-29
5	108	6	276337	2021-10-29

Запит на перегляд інформації у таблиці категорії проекту та результат його виконання:

Data Output | Explain | Messages | Notifications

	Name_project character (50)	Number_project [PK] integer	Creation_date date	Budget integer	Category character (50)
2	Спорт для людей ...	141	2021-09-04	293020	3
3	Волейбольно-баск...	31	2021-08-26	489996	8
4	Здорова дитина - ...	113	2021-09-03	489417	10
5	Улюблений Сакса...	108	2021-09-03	489300	6
6	Двор счастливого...	62	2021-09-01	626816	11
7	Школа починаеть...	77	2021-09-02	489947	8
8	Фестиваль інтеле...	161	2021-09-05	258100	1
9	Вшанування геть...	168	2021-09-05	265100	2
10	Комфортні хвилин...	32	2021-08-26	299994	5
11	У здоровому тілі –...	146	2021-09-04	269029	3
12	На крилах успіху ...	166	2021-09-05	495868	8
13	Реабілітаційний ц...	156	2021-09-04	299000	2
14	Безпечний сквер ...	68	2021-09-01	277700	5
15	"Пильне око"- сист...	114	2021-09-03	227360	5
16	Фестиваль "Від Те...	17	2021-08-18	486403	4
17	Ukrainian open air f...	164	2021-09-05	480300	4
18	Kids HUB на 96 му ...	160	2021-09-05	489900	7
19	"Павляндія" Дитяч...	111	2021-09-03	499955	7
20	"STEM-світ" - майд...	131	2021-09-03	1371709	9

Запит на перегляд інформації у таблиці статус проекту та результат його виконання:

Data Output | Explain | Messages

	Name character (20)	Number integer
1	Всі проєкти	1
2	Відхилений	2
3	На голосуванні	3
4	Відхилений остаточно	4

Запит для вибірки проектів з оздоровлення:

Query Editor | Query History

```
1  SELECT "Name", "Number"
2      FROM public."Category"
3      where "Name" like 'Благоустрій%';
```

Data Output | Explain | Messages | Notifications

	Name character (50)	Number [PK] integer
1	Благоустрій мікро...	6
2	Благоустрій мікро...	7

Запит для вибірки проектів з закладів середньої освіти:

Query Editor | Query History

```
1  SELECT "Name", "Number"
2      FROM public."Category"
3      where "Name" like 'Заклади%';
```

Data Output | Explain | Messages | Notifications

	Name character (50)	Number [PK] integer
1	Заклади загально...	9
2	Заклади загально...	8

Запит для вибірки дошкільних проектів:

Query Editor | Query History

```
1  SELECT "Name", "Number"
2      FROM public."Category"
3      where "Name" like 'Дошкільні%';
```

Data Output | Explain | Messages | Notifications

	Name character (50)	Number [PK] integer
1	Дошкільні та поза...	10
2	Дошкільні та поза...	11

Запити для копіювання інформації між таблицями через текстовий файл:

```
1)    copy public."data_project" to 'C:/data_project.txt'
with (format csv);

2)    copy public."user_data" to 'C:/user_data.txt'
with (format csv);

3)    copy public."voices" to 'C:/voices.txt'
with (format csv);

4)    copy public."Status" to 'C:/Status.txt'
with (format csv);

5)    copy public."Category" to 'C:/Category.txt'
with (format csv);
```

Запит для перегляду осіб, які проголосували за всі проекти:

Query Editor | Query History

```
1  SELECT data_project."Name_project",  "Name", "Voting_date"
2      FROM public.voices, public.data_project, public.user_data
3      where user_data."Passport_ID"=voices."Passport_ID"
4      and  data_project."Number_project"=voices."Number_project";
5
```

Data Output | Explain | Messages | Notifications

	Name_project character (50)	Name character (50)	Voting_date date
1	Волейбольно-баск...	Шевченко Ігор ...	2021-10-29
2	Спорт для людей ...	Шевченко Ігор ...	2021-10-29
3	Шкільний коридо...	Шевченко Ігор ...	2021-10-29
4	Здорова дитина - ...	Шевченко Ігор ...	2021-10-29
5	Улюблений Сакса...	Шевченко Ігор ...	2021-10-29

Запит для визначення кількості відданних голосів:

Query Editor | Query History

```
1  SELECT count ("Name")
2      FROM public.voices, public.data_project, public.user_data
3      where user_data."Passport_ID"=voices."Passport_ID"
4      and  data_project."Number_project"=voices."Number_project" and voices."Voting_date"='2021-10-29';
5
```

Data Output | Explain | Messages | Notifications

	count bigint
1	5

Запит для визначення кількості голосів відданних за кожний проект:

Query Editor | Query History

```
SELECT count ("Name_project") , "Name_project"
    FROM public.voices, public.data_project, public.user_data
    where user_data."Passport_ID"=voices."Passport_ID"
    and  data_project."Number_project"=voices."Number_project" and voices."Voting_date"='2021-10-29'
    group by data_project."Name_project";
```

Data Output | Explain | Messages | Notifications

	count bigint	Name_project character (50)
1	1	Волейбольно-баск...
2	1	Здорова дитина - ...
3	1	Спорт для людей ...
4	1	Улюблений Сакса...
5	1	Шкільний коридо...

Висновки за розділом

Для реалізації бази даних для інформаційної системи обрано реляційну СУБД PostgreSQL. Також для адміністрування та розробки БД використано платформу з відкритим вихідним кодом PgAdmin. Спроектована база даних складається з 6 таблиць. Розроблено схему бази даних. Більшість зв'язків, які використовуються – один до багатьох. Наведено запити для створення та заповнення таблиць, а також приклади запитів для роботи з розробленою БД.

РОЗДІЛ 3. ВИБІР ЗАСОБІВ ДЛЯ СТВОРЕННЯ САЙТУ ТА ЙОГО РЕАЛІЗАЦІЯ

3.1 Огляд технології ASP.NET MVC

Багато сайтів і порталів насправді є веб-додатками. Це поштові ресурси, сторінки соцмереж, пошукові інструменти, системи інтернет-банкінгу, інтернет-магазини, онлайнові редактори, ігри та багато іншого. Ключова відмінність веб-додатків від веб-сайтів у тому, що вони відібражають не статичні дані як документи в Інтернеті, а динамічні програми.

Веб-сайт – це набір пов'язаних веб-сторінок, який містить зображення, текст, аудіо, відео тощо. Він може складатися з однієї або декількох сторінок. Веб-сайт надає візуальний і текстовий вміст, який користувачі можуть переглядати та читати.[10]

Веб-додаток – це частина програмного забезпечення, до якої можна отримати доступ у браузері. Він потребує аутентифікації. Веб-додаток використовує комбінацію сценаріїв на стороні сервера та сценаріїв на стороні клієнта для представлення інформації. Для керування запитами від користувачів потрібен сервер. Приклад: Google Apps, Amazon, YouTube.

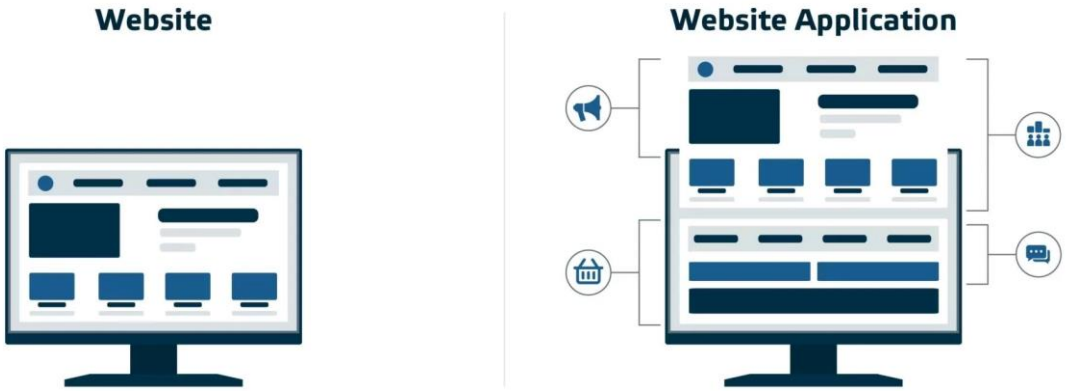


Рисунок 3.1 – Порівняння веб-сайту і веб-додатку

Веб-розробка — це процес створення веб-сайтів і веб-додатків за допомогою різноманітних мов програмування та технологій. HTML, CSS і PHP є одними з найпоширеніших мов для веб-розробки.

HTML і CSS є основою веб-розробки. HTML використовується для структурування вмісту веб-сторінки, тоді як CSS використовується для стилізації та компонування вмісту.

PHP — це серверна мова сценаріїв, яка використовується для створення динамічних веб-сторінок і веб-додатків. Він зазвичай використовується в поєднанні з HTML і CSS для створення динамічних веб-сайтів.

CGI (Common Gateway Interface) - це перша фонова технологія, яка використовується для створення динамічних веб-сайтів.

Коли програма CGI працює, клієнт спочатку надсилає запит до програми CGI на сервері, а коли сервер отримує запит клієнта, відкривається новий процес для виконання програми CGI для обробки запиту клієнта. Програма CGI нарешті передає результати виконання (HTML-код сторінки) назад клієнту.

Оскільки програма CGI відкриватиме новий процес кожного разу, коли клієнт відповідає, сервер відкриватиме кілька процесів, коли кілька користувачів одночасно надсилатимуть запити CGI, таким чином збільшуючи навантаження на сервер і роблячи його неефективним. Це причина, чому додатки CGI стають менш популярними в Інтернеті останніми роками з появою нових фонових технологій. Режим CGI не підходить для програм із великим трафіком.

ASP (Active Server pages) — це технологія динамічної веб-сторінки на платформі Microsoft. Концепція ASP, запропонована Microsoft, досягла великих успіхів у технології проектування інтерактивних веб-сторінок. Він використовує трирівневу обчислювальну структуру, яка розділяє веб-сервер (логічний рівень), клієнтський браузер (рівень представлення) і сервер бази даних (рівень даних) з хорошою розширюваністю.

Коли ASP виконується, IIS (Internet Information Services) викликає його для виконання коду ASP, вбудованого в HTML, і результат надсилається клієнту разом із вихідним HTML.

ASP може працювати лише на платформах Windows. У той час як надійність Unix і відкритий вихідний код Linux широко використовуються на веб-сервері, навпаки, кореляція платформи ASP значно обмежує його застосування.

Google Sites – це веб-платформа, створена для створення веб-сайту від Google. Він схожий на інші платформи, такі як Wix або WordPress, але безкоштовний. Він був започаткований компанією під назвою JotSpot, а пізніше куплений і розроблений Google для нової платформи сайту Google. Цей веб-сайт не лише дозволяє створювати прості веб-сайти, але й безкоштовно розміщувати їх на домені сайту.[12]

Фреймворки стали важливою частиною веб-розробки, оскільки стандарти веб-додатків постійно ускладнюються.

Django — сучасний високорівневий відкритий Python-фреймворк для розробки вебсистем. Сайт на Django складається з однієї або декількох частин, які бажано робити модульними. Це одна з істотних архітектурних відмінностей Django від деяких інших фреймворків (наприклад Ruby on Rails).

Особливості архітектури Django полягає в тому, що він надає ряд засобів, які допомагають у швидкій розробці веб-сайтів інформаційного характеру. Розробнику не потрібно створювати контролери та сторінки для адміністративної частини сайту, в цьому фреймворку є вбудований модуль для керування вмістом, який можна включити до будь-якого сайту, зробленого на Django, і який може керувати відразу декількома сайтами на одному сервері.

Адміністративний модуль Django дозволяє створювати, змінювати і вилучати будь-які об'єкти наповнення сайту, протоколюючі всі дії, а також надає інтерфейс для управління користувачами і групами (з призначенням прав).

ASP.NET — це платформа веб-розробки, яка надає модель програмування, комплексну інфраструктуру програмного забезпечення та різноманітні служби, необхідні для створення надійних веб-додатків для ПК, а також мобільних пристроїв.

ASP.NET працює на основі протоколу HTTP та використовує команди та політики HTTP для встановлення двостороннього зв’язку та співпраці між браузером і сервером.

ASP.NET є частиною платформи Microsoft .Net. Програми ASP.NET — це скомпільовані коди, написані з використанням розширюваних і багаторазово використовуваних компонентів або об’єктів, наявних у структурі .Net. Ці коди можуть використовувати всю ієрархію класів у .Net framework.

Коди програми ASP.NET можна писати будь-якою з таких мов: C#, Visual Basic.Net, Jscript, J#.

ASP.NET використовується для створення інтерактивних веб-програм, керованих даними, через Інтернет. Він складається з великої кількості елементів керування, таких як текстові поля, кнопки та мітки для складання, налаштування та керування кодом для створення HTML-сторінок.

За останні кілька років веб-сайти змінили прості HTML-сторінки з невеликою кількістю CSS на неймовірно складні програми, над якими одночасно працюють тисячі розробників. Щоб працювати з цими складними веб-додатками, розробники використовують різні шаблони дизайну, роблячи код менш складним. Найпопулярнішим із цих шаблонів є MVC, також відомий як контролер перегляду моделі.

MVC створив Trygve Reenskaug. Основною метою цього шаблону проектування було вирішення проблеми користувачів, які контролюють великий і складний набір даних, розділивши велику програму на окремі розділи, кожен з яких має власне призначення.

Структура Model-View-Controller (MVC) — це шаблон архітектури/проектування, який розділяє програму на три основні логічні компоненти Model, View і Controller . Кожен архітектурний компонент створено для обробки конкретних аспектів розробки програми. Він ізолює бізнес-логіку та рівень презентації один від одного. Традиційно використовувався для настільних графічних інтерфейсів користувача (GUI). Сьогодні MVC є однією з найбільш часто використовуваних галузевих стандартних структур веб-розробки для створення масштабованих і розширюваних проектів. Також використовується для розробки мобільних додатків.

Особливості MVC:

- Забезпечує чітке розділення бізнес-логіки, логіки Ul і логіки введення.
- Пропонує повний контроль над HTML і URL-адресами, що полегшує розробку архітектури веб-додатків.
- Це потужний компонент відображення URL-адрес, за допомогою якого можна створювати програми, які мають зрозумілі та доступні для пошуку URL-адреси.
- Підтримує тестову розробку (TDD).

Структура MVC включає наступні 3 компоненти: контролер, модель, перегляд

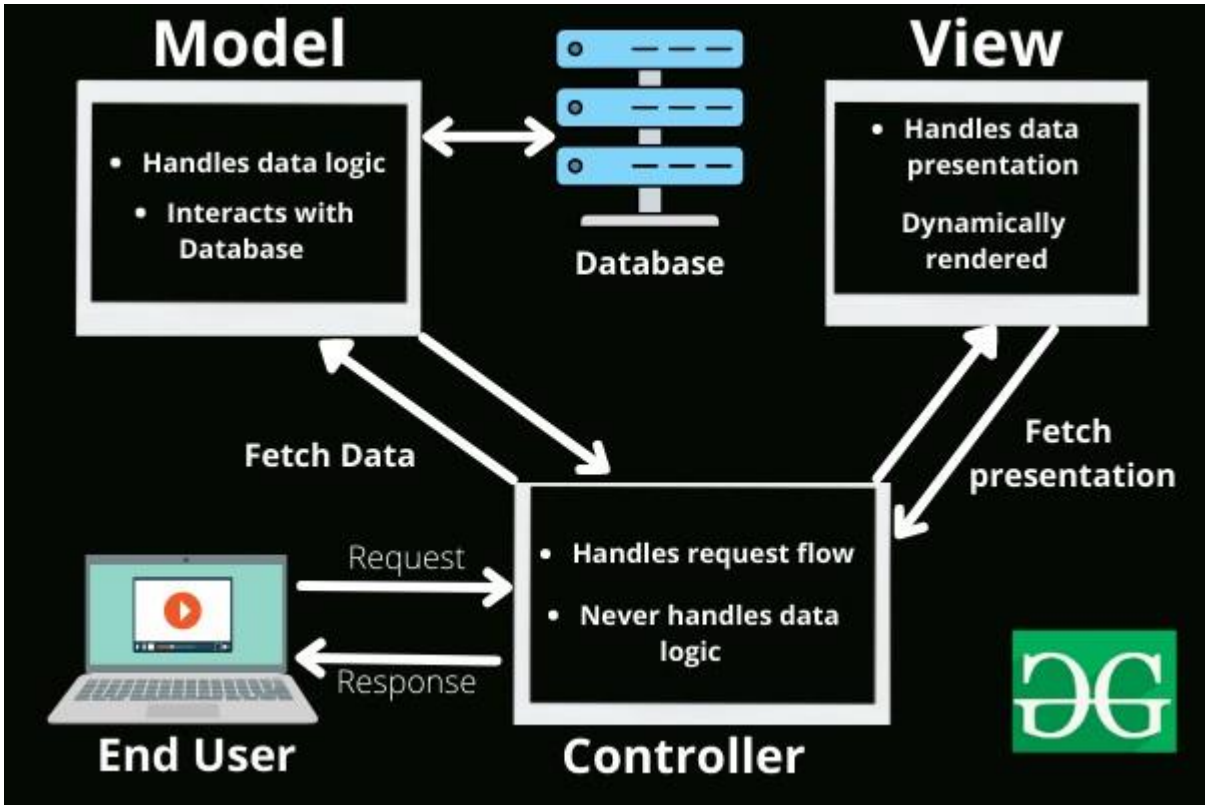


Рисунок 3.2 – Дизайн архітектури MVC

Контролер — це компонент, який забезпечує взаємозв'язок між представленнями та моделлю, тому він діє як посередник. Контролеру не потрібно турбуватися про обробку логіки даних, він просто повідомляє моделі, що робити. Він обробляє всю бізнес-логіку та вхідні запити, маніпулює даними за допомогою компонента Model і взаємодіє з View для відтворення кінцевого результату.[13]

Компонент View використовується для всієї логіки інтерфейсу користувача програми. Він створює інтерфейс користувача для користувача. Представлення створюються на основі даних, зібраних компонентом моделі, але ці дані беруться не безпосередньо, а через контролер. Він взаємодіє лише з контролером.

Компонент Model відповідає всій логіці даних, з якою працює користувач. Це можуть бути дані, які передаються між компонентами View і

Controller, або будь-які інші дані, пов’язані з бізнес-логікою. Також може додавати або отримувати дані з бази даних. Він відповідає на запит контролера, оскільки контролер не може самостійно взаємодіяти з базою даних. Модель взаємодіє з базою даних і повертає необхідні дані контролеру.
Переваги MVC:

- Код легко підтримувати і розширювати.
- Компонент моделі MVC можна протестувати окремо.
- Компоненти MVC можна розробляти одночасно.
- Зменшує складність, розділяючи програму на три блоки.
- Підтримка тестової розробки (TDD).
- Добре працює для веб-програм, які підтримуються великими командами веб-дизайнерів і розробників.
- Ця архітектура допомагає тестувати компоненти незалежно, оскільки всі класи та об’єкти незалежні один від одного.
- Зручність пошукової оптимізації (SEO) .

Недоліки MVC:

- Важко читати, змінювати, тестувати та повторно використовувати
- Не підходить для створення невеликих програм.
- Навігація в структурі може бути складною, оскільки вводить нові рівні абстракції, що вимагає від користувачів адаптації до критеріїв декомпозиції MVC.
- Підвищена складність і неефективність даних.

Елементи проекту MVC:
/Areas (області) - це спосіб розбиття великих додатків на дрібні порції.
/Dependencies - елемент залежностей надає докладні відомості про всі пакети, від яких залежить проект.
/Components - тут визначено класи компонентів уявлень, які використовуються для відображення самодостатніх засобів.
/Controllers - сюди розміщуються класи контролері. Вони можуть розміщуватися будь-де, тому що всі вони компілюються в ту саму збірку.
/Data - тут визначено класи контексту бази даних.
/Migrations - тут зберігаються деталі схем баз даних, щоб можна було оновлювати розгорнуті бази даних.
/Models - сюди розміщуються класи моделей уявлень та моделей предметної області. Класи моделей можуть визначатися будь-де в поточному проекті або навіть в окремому проекті.
/Views - тут зберігаються уявлення та часткові уявлення, зазвичай згруповані в папки з іменами контролерів, до яких вони належать.
/Views/Shared - тут зберігаються компонування та уявлення, які не є специфічними для контролерів.
/Views/_Viewimports.cshtml - застосовується для вказівки просторів імен, які будуть включені у файли уявлень Razor.
/Views/ViewStart.cshtml - дозволяє вказати стандартне компонування для механізму візуалізації Razor.

/bower.json - Цей файл за замовчуванням прихований. Він містить список пакетів.
/project.json - У цьому файлі вказані базові конфігураційні параметри для проекту, у тому числі пакети NuGet, які він використовує .
/Program.cs - клас конфігурує платформу, на якій розміщується програма
/Startup . cs - клас конфігурує сам додаток.
/wwwroot - Сюди розміщується статичний вміст, такий як файли CSS та зображень. Крім того, саме сюди диспетчер пакетів Bower встановлює пакети JavaScript та CSS.[14]

3.2 Розробка веб-додатку

У якості програмного середовища для реалізації веб-додатку, який розробляється обираємо MS Visual Studio 2022. Для розробки використаємо мову програмування C#.

VS підтримує багато інтегрованиї модулів для реалізації та підтримки різноманітних технологій.

В якості шаблону для проекту оберемо ASP.NET Core на базі архітектури MVC:

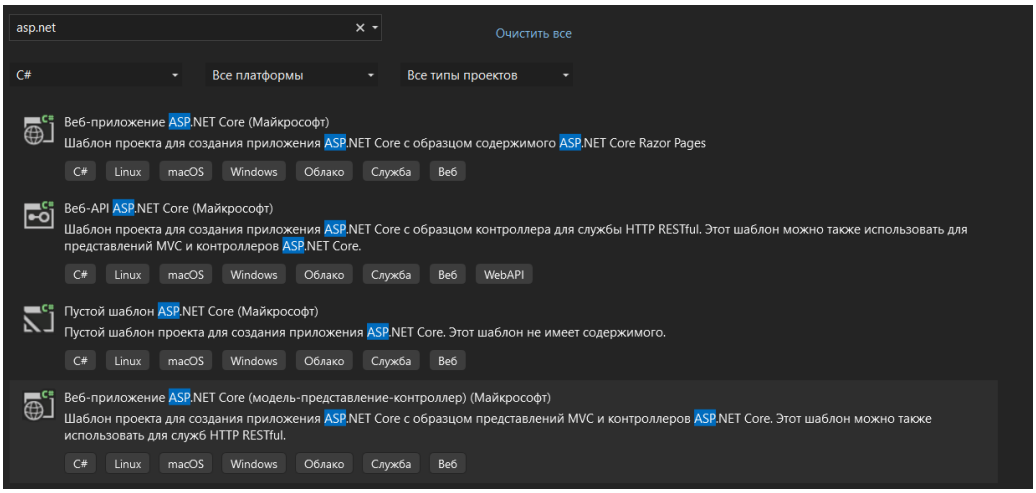


Рис 3.3 – Вікно вибору шаблону проекту

При налаштуванні проекту оберемо налаштування індивідуальних облікових записів, щоб автоматично згенерувалася частина, яка відповідає за ідентифікацію користувачів:

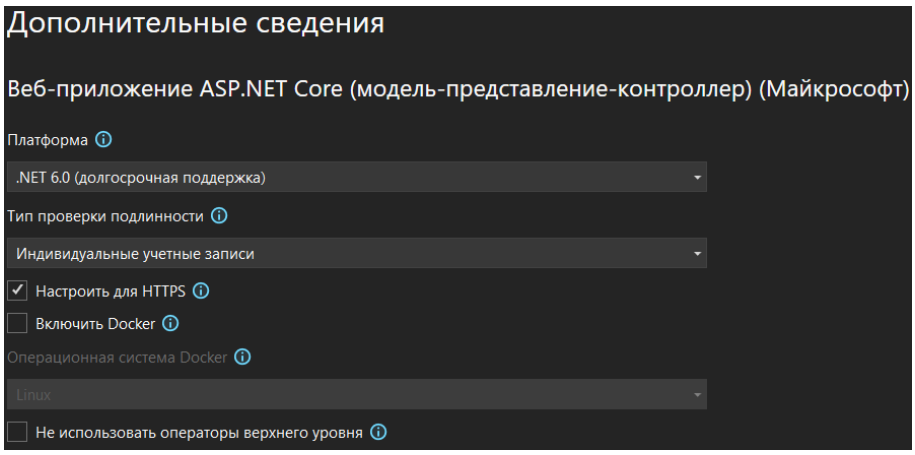


Рисунок 3.4 – Налаштування перевірки аутентичності користувачів

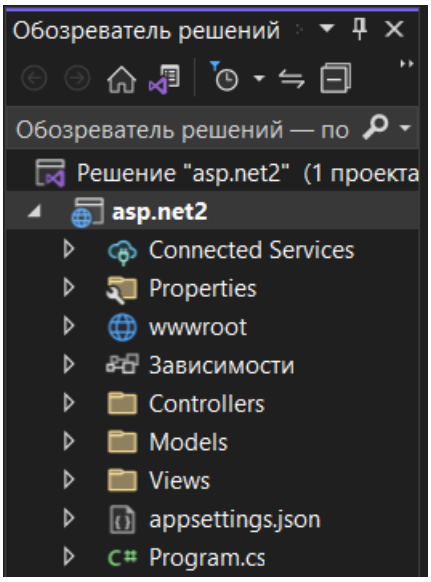


Рисунок 3.5 – Початкова структура файлів та папок проекту ASP.NET Core MVC

Поняття маршрутів

На додаток до моделей, уявлень та контролерів у додатках MVC застосовується система маршрутизації ASP.NET, яка визначає, як URL відображаються на контролери та дії. Маршрут - це правило, яке використовується для вирішення того, як обробляти запит. Коли середовище Visual Studio створює проект MVC, воно додає ряд стандартних маршрутів, що виступають як початкові. Можна запитувати будь-яку з наступних URL-адрес, і вони будуть спрямовані на дію Index класу HomeController.

Розробимо моделі для об’єктів, інформацію про які буде збирати, обробляти та передавати користувачам наш додаток.

Клас користувачі:

```
public class UsersModel
{
[Key]
public string User_name { get; set; }
public string E_mail { get; set; }
```

```
public string Passport_id { get; set; }
public string Registration_date { get; set; }
public string Date_of_birth { get; set; }
}
```

Клас категорії:

```
public class Category
{
[Key]
public string Category_name { get; set; }
public string Category_id { get; set; }
}
```

Клас проекти:

```
public class Project
{
[Key]
public string Project_name { get; set; }
public string Project_id { get; set; }
public string Creation_date { get; set; }
public string Total_budget { get; set; }
public string Category_id { get; set; }
public string Status_id { get; set; }
}
```

Уявлення, пов'язані з контролером Home, містяться в папці на ім'я Views/Home. Уявлення, які не є специфічними для окремого контролера, зберігаються в папці під назвою Views/Shared. Середовище Visual Studio створює папки Home та Shared автоматично, коли використовується шаблон Web Application, і в рамках початкової підготовки проекту містить у них кілька уявлень-заповнювачів.Розробимо представлення для форм, за допомогою яких збиратиметься інформація.
Форма для додавання нової категорії:

```
@model asp.net2.Models.Category
@{
Layout = null;
}
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width" />
<title>Реєстрація</title>
<link rel="stylesheet" href="/css/styles.css" />
<link rel="stylesheet" href="/lib/bootstrap/dist/css/bootstrap.css" />
</head>
<body>
<div class="panel panel-success text-center">
```

```
<div class="panel-heading text-center"><h4>Додавання нової
категорії</h4></div>
<div class="panel-body">
<form class="p-a-1" asp-action="AddCategory" method="post">
<div asp-validation-swnmary="All"></div>
<div class="form-group">
<label asp-for="Category_name">назва_категорії:</label>
<input class="form-control" asp-for="Category_name">
</div>
<div class="form-group">
<label asp-for="Category_id">номер_категорії:</label>
<input class="form-control" asp-for="Category_id" />
</div>
<br />
<button class="btn btn-primary" type="submit">Додати</button>
<a class="nav-link" asp-action="ListResponses1">Натисніть, щоб
побачити список категорій</a>
<a class="btn btn-primary" asp-action="Index2">Category</a>
</form>
</div>
</div>
</body>
</html>
```

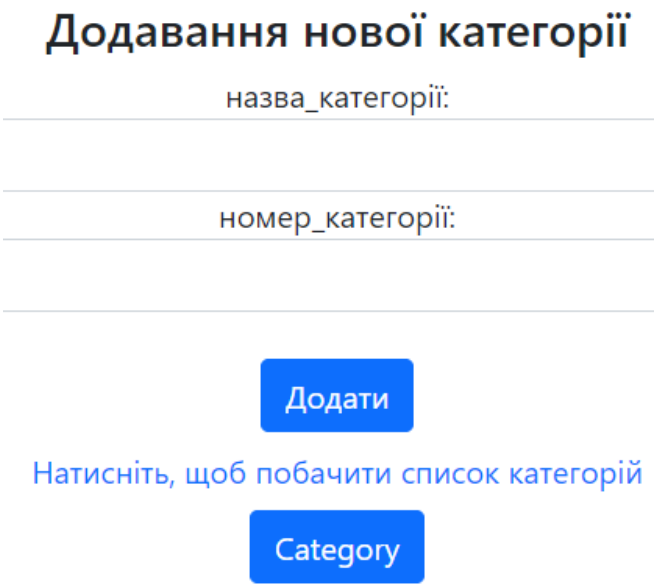


Рисунок 3.6 – Сторінка “Додавання нової категорії”

Форма для додавання нового проекту:

```
@model asp.net2.Models.Project
@{
Layout = null;
}
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width" />
<title>Проекти</title>
```

```
<link rel="stylesheet" href="/css/styles.css" />
<link rel="stylesheet" href="/lib/bootstrap/dist/css/bootstrap.css" />
</head>
<body>
<div class="panel panel-success text-center">
<div class="panel-heading text-center"><h4>Додавання нового
проекту</h4></div>
<div class="panel-body">
<form class="p-a-1" asp-action="AddProject" method="post">
<div asp-validation-swnmary="All"></div>
<div class="form-group">
<label asp-for="Project_name">назва_проекту:</label>
<input class="form-control" asp-for="Project_name">
</div>
<div class="form-group">
<label asp-for="Project_id">номер_проекту:</label>
<input class="form-control" asp-for="Project_id" />
</div>
<div class="form-group">
<label asp-for="Creation_date">дата_створення:</label>
<input class="form-control" asp-for="Creation_date" />
</div>
<div class="form-group">
<label asp-for="Total_budget">загальний_бюджет:</label>
<input class="form-control" asp-for="Total_budget" />
</div>
<div class="form-group">
<label asp-for="Category_id">номер_категорії:</label>
<input class="form-control" asp-for="Category_id" />
</div>
<div class="form-group">
<label asp-for="Status_id">статус_проекту:</label>
<input class="form-control" asp-for="Status_id" />
</div>
<br/>
<button class="btn btn-primary" type="submit">Додати</button>
<a><h4>Натисніть, щоб побачити список проектів</h4></a>
<a class="btn btn-primary" asp-action="Index3">Projects</a>
</form>
</div>
</div>
</body>
</html>
```

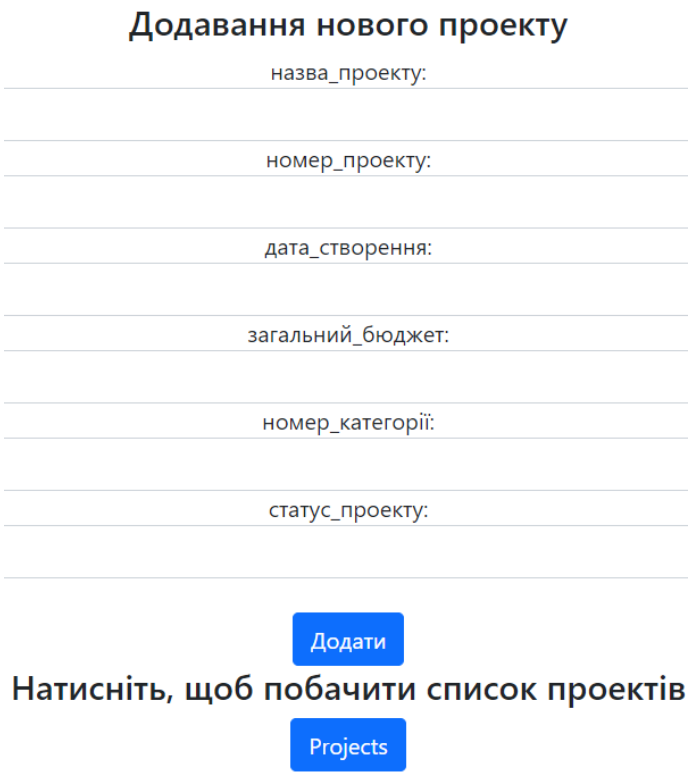


Рисунок 3.7 – Сторінка “Додавання нового проекту”

Представлення для відображення таблиці “Категорії”:

```
@model IEnumerable<asp.net2.Models.Category>

@{
    ViewData["Title"] = "Index2";
}

<h1>Таблиця категорії у БД</h1>

<p>
    <a asp-action="Create">Create New</a>
</p>
<table class="table">
<thead>
<tr>
            <th>
                @Html.DisplayNameFor(model => model.Category_name)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Category_id)
            </th>

        </tr>
    </thead>
    <tbody>
        @foreach (var item in Model)
        {
            <tr>
```

```
                <td>
                    @Html.DisplayFor(modelItem => item.Category_name)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.Category_id)
                </td>

            </tr>
        }
    </tbody>
</table>
```

Таблиця категорії у БД

Create New

Category_name	Category_id
igor	1111
igor	123345
Игорь	1234
Игорь	333
Test	1

Рисунок 3.8 – Сторінка “Відображення таблиці категорій”

Представлення для відображення таблиці “Проекти”:

```
@model IEnumerable<asp.net2.Models.Project>

@{
    ViewData["Title"] = "Index3";
}

<h1>Таблиця проектів</h1>

<p>
    <a asp-action="Create">Create New</a>
</p>
<table class="table">
    <thead>
        <tr>
            <th>
                @Html.DisplayNameFor(model => model.Project_name)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Project_id)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Creation_date)
```

```
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Total_budget)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Category_id)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Status_id)
            </th>

        </tr>
    </thead>
    <tbody>
        @foreach (var item in Model)
        {
            <tr>
                <td>
                    @Html.DisplayFor(modelItem => item.Project_name)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.Project_id)
                </td>
                <th>
                    @Html.DisplayFor(model => item.Creation_date)
                </th>
                <th>
                    @Html.DisplayFor(model => item.Total_budget)
                </th>
                <th>
                    @Html.DisplayFor(model => item.Category_id)
                </th>
                <th>
                    @Html.DisplayFor(model => item.Status_id)
                </th>

            </tr>
        }
    </tbody>
</table>
```

Таблиця проектів

Create New

Project_name	Project_id	Creation_date	Total_budget	Category_id	Status_id
Шкільний коридор для веселих перерв	67	**01.09.2021 00:00:00**	**378659**	**8**	**1**
Спорт для людей літнього віку. Час-дій	141	**04.09.2021 00:00:00**	**293020**	**3**	**1**
Волейбольно-баскетбольний майданчик КНВК № 81	31	**26.08.2021 00:00:00**	**489996**	**8**	**1**
Здорова дитина - майбутнє України	113	**03.09.2021 00:00:00**	**489417**	**10**	**1**
Улюблений Саксаганський: сквер з широким серцем	108	**03.09.2021 00:00:00**	**489300**	**6**	**1**

Рисунок 3.9 – Сторінка “Відображення таблиці проектів”

Представлення для головної сторінки додатку:

```
@{
Layout = null;
}
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width"/>
<title>Shevchenko</title>
<link rel="stylesheet" href="/lib/bootstrap/dist/css/bootstrap.css"/>
</head>
<body>
    <div class="text-center">
        <h1>Підтримка соціальних проектів!</h1>
        <h5>Web-додаток для оприлюднення пропозицій відбудови
            інфраструктури країни та контролю за їх реалізацією <br/>
            Для відбудови зруйнованої інфраструктури України після
перемоги вже збираються матеріали щодо знищеного та зіпсованого майна,
як особистого так і громадського. <br/>
            Метою цього web-ресурсу є оприлюднення інформації щодо
запропонованих проектів відновлення та відбудови, залучення
громадськості до вирішення соціальних проблем, шляхом розміщення або
підтримки проектних про пропозицій, проведення відкритого голосування
за такі проекти, а також забезпечення відкритості та прозорості
діяльності. Фінансування таких об’єктів може здійснюватися не лише
державою, але й фізичними та юридичними особами, які мають бажання
підтримати обрані рішення. <br />
        </h5>
        <a class="btn btn-primary" asp-action="Index1">Users</a>
        <a class="btn btn-primary" asp-action="Index2">Category</a>
        <a class="btn btn-primary" asp-action="Index3">Project</a>
        <a class="btn btn-primary" asp-
action="AddCategory">AddCategory</a>
        <a class="btn btn-primary" asp
action="AddProject">AddProject</a>
    </div>
</body>
</html>
```

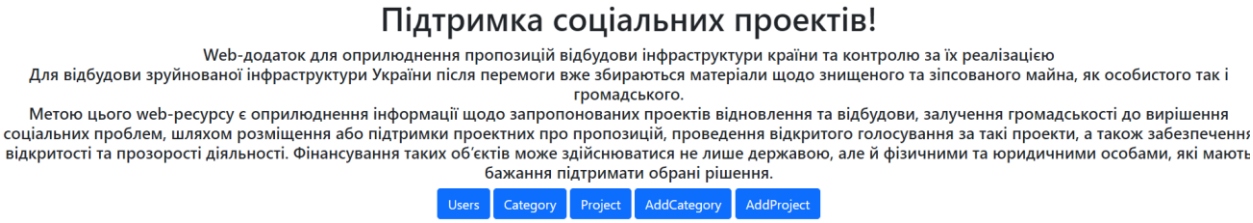


Рисунок 3.10 – Головна сторінка

Тепер налаштуємо контроллер для взаємодії складових додатку.
Виклик сторінки з формою для додання проекту:

```
[HttpGet]
        public ViewResult AddProject()
        {
            return View();
        }
        [HttpPost]
        public IActionResult AddProject(ProjectResponse
projectResponse)
        {
            NpgsqlConnection conn = new
NpgsqlConnection("Server=localhost;Port=5432;Database=social_project1;
User Id=postgres;Password=sheva2847;");
            conn.Open();
            Repository2.AddProjResponse(projectResponse);
            using (var cmd = new NpgsqlCommand($" INSERT  INTO
data_project (project_name, project_id, creation_date, total_budget,
category_id, status_id) VALUES('{projectResponse.Project_name}',
'{projectResponse.Project_id}', '{projectResponse.Creation_date}',
'{projectResponse.Total_budget}', '{projectResponse.Category_id}',
'{projectResponse.Status_id}')", conn))
            {
                cmd.ExecuteNonQueryAsync();
            }

            return RedirectToAction("Index3");
        }
```

Підключення до бази даних (таблиця Проекти):

```
[HttpGet]
        public ViewResult AddProject()
        {
            return View();
        }
        [HttpPost]
        public IActionResult AddProject(ProjectResponse
projectResponse)
        {
            NpgsqlConnection conn = new
NpgsqlConnection("Server=localhost;Port=5432;Database=social_project1;
User Id=postgres;Password=sheva2847;");
            conn.Open();
            Repository2.AddProjResponse(projectResponse);
            using (var cmd = new NpgsqlCommand($" INSERT  INTO
data_project (project_name, project_id, creation_date, total_budget,
category_id, status_id) VALUES('{projectResponse.Project_name}',
'{projectResponse.Project_id}', '{projectResponse.Creation_date}',
'{projectResponse.Total_budget}', '{projectResponse.Category_id}',
'{projectResponse.Status_id}')", conn))
```

```
            {
                cmd.ExecuteNonQueryAsync();
            }
            return RedirectToAction("Index3");
        }
```

Виклик сторінки з формою для додання категорії:

```
public IActionResult Index2()
        {
            List<Category> displaycategory = new List<Category>();
            NpgsqlConnection conn = new
NpgsqlConnection("Server=localhost;Port=5432;Database=social_project1;
User Id=postgres;Password=sheva2847;");
            conn.Open();
            NpgsqlCommand comm = new NpgsqlCommand();
            comm.Connection = conn;
            comm.CommandType = CommandType.Text;
            comm.CommandText = "select * from category1";
            NpgsqlDataReader sdr = comm.ExecuteReader();
            while (sdr.Read())
            {
                var categorylist = new Category();
                categorylist.Category_name =
sdr["category_name"].ToString();
                categorylist.Category_id =
sdr["category_id"].ToString();

                displaycategory.Add(categorylist);
            }

            return View(displaycategory);
        }
```

Підключення до бази даних (таблиця Категорії):

```
public ViewResult AddCategory()
        {
            return View();
        }
        [HttpPost]
        public IActionResult AddCategory(CategoryResponse
categoryResponse)
        {
            NpgsqlConnection conn = new
NpgsqlConnection("Server=localhost;Port=5432;Database=social_project1;
User Id=postgres;Password=sheva2847;");
            conn.Open();
            Repository1.AddCatResponse(categoryResponse);
            using (var cmd = new NpgsqlCommand($"INSERT  INTO
category1 (category_name, category_id)
```

```
VALUES('{categoryResponse.Category_name}',
'{categoryResponse.Category_id}')", conn))
            {
                cmd.ExecuteNonQueryAsync();
            }
            return RedirectToAction("Index2");
        }
```

Щоб отримати і обробити надіслані дані форми, скористаємося основною можливістю контролера, додавши другий метод дії RsvpForm(), щоб отримати своє розпорядження наступне.

- Метод, який відповідає на HTTP-запити GET. Щоразу, коли хтось клацає на засланні, браузер зазвичай видає запит GET. Ця версія дії буде відповідати за відображення спочатку порожньої форми, коли вперше відвідує Home/RsvpForm.
- Метод, який відповідає на HTTP-запити POST. За замовчуванням форми, візуалізовані за допомогою Html.BeginForm(), надсилаються браузером як запити POST. Ця версія дії буде відповідати за отримання надісланих даних та ухвалення рішення про те, що з ними робити.

Обробка запитів GET і POST окремих методах C# сприяє забезпеченню акуратності коду контролера так, як ці два методи мають різні обов'язки. Обидва методи дій викликаються через той же URL, але в залежності від виду запиту - GET або POST - інфраструктура MVC викликає відповідний метод.

Прив'язка моделі - потужний засіб, який позбавляє кропіткої і важкої праці по взаємодії з HTTP-запитами безпосередньо і дозволяє працювати з об'єктами C#, а не мати справу з індивідуальними значеннями даних, що відправляються браузером. Об'єкт CategoryResponse, який передається цьому методу дії як параметр, автоматично заповнюється даними з полів форми.

Однією з цілей програми є надання підсумкової сторінки з деталями про проект, що означає необхідність збереження відповідей. Це виконується за допомогою створеної в пам'яті колекції об'єктів. У цьому додатку такий підхід не підійде, так як дані відповідей будуть втрачатися в результаті зупинки або перезапуску програми, але він дозволяє зосередити увагу на MVC і створити програму, яка може бути легко скинута у свій початковий стан.

Клас Repository та його члени оголошені статичними, щоб полегшити збереження та вилучення даних з різних місць програми. Інфраструктура MVC пропонує складніший підхід до визначення загальної функціональності, що називається використанням залежностей.[17]

```
namespace asp.net2.Models
{
    public class Repository1
    {
        private static List<CategoryResponse> cat_responses = new
List<CategoryResponse>();
        public static IEnumerable<CategoryResponse> Cat_responses
        {
```

```
            get
            {
                return cat_responses;
            }
        }
        public static void AddCatResponse(CategoryResponse
cat_response)
        {
            cat_responses.Add(cat_response);
        }
    }

    public class Repository2
    {
        private static List<ProjectResponse> proj_responses = new
List<ProjectResponse>();

        public static IEnumerable<ProjectResponse> Proj_responses
        {
            get
            {
                return proj_responses;
            }
        }
        public static void AddProjResponse(ProjectResponse
proj_response)
        {
            proj_responses.Add(proj_response);
        }
    }
}
```

Все, що необхідно зробити з даними форми, надісланими у запиті - це передати методу Repository.AddResponse() як аргумент об'єкт GuestResponse, який був переданий методу дії, щоб відповідь могла бути збережена.

Новий метод дії називається ListResponses1(), він викликає метод View(), використовуючи властивість Repository.Responses як аргумент. Саме так метод дії надає дані строго типізованого подання. Колекція об'єктів GuestResponse1 фільтрується із застосуванням LINQ. Метод дії ListResponses1 не вказує ім'я подання, яке має застосовуватися для відображення колекції об'єктів GuestResponse1, тому буде задіяна угода про іменування та MVC ініціює пошук подання на ім'я ListResponses1.cshtml у папках Views/Home та Views/Shared .

```
@model IEnumerable<asp.net2.Models.CategoryResponse>
@{
    Layout = null;
}
<!DOCTYPE html>
<html>
```

```
<head>
    <meta name="viewport" content="width=device-width" />
    <link rel="stylesheet"
href="/lib/bootstrap/dist/css/bootstrap.css" />
    <title>Користувачі</title>
</head>
<body>
    <h2>Список зареєстрованих користувачів</h2>
    <table class="panel-body table table-sm table-striped table-
bordered">
        <thead>
            <tr>
                <th>Назва категорії</th>
                <th>Номер категорії</th>
            </tr>
        </thead>
        <tbody>
            @foreach (asp.net2.Models.CategoryResponse r in Model)
            {
                <tr>
                    <td>@r.Category_name</td>
                    <td>@r.Category_id</td>
                </tr>
            }
        </tbody>
    </table>
</body>
</html>
```

Файли уявлень Razor мають розширення cshtml, тому що містять суміш коду C# та елементів HTML. Це можна помітити в лістингу 2.18, де використовується цикл foreach для обробки всіх об'єктів GuestResponse, які метод дії передає уявленню із застосуванням методу View(). На відміну від нормального циклу foreach мови C# тіло циклу foreach з Razor містить елементи HTML, що додаються до відповіді, яка буде відправлена назад браузеру. У цьому поданні для кожного об'єкта GuestResponse генерується елемент tr, який містить елементи td, заповнені значеннями властивостей об'єкта.

Базові засоби Bootstrap працюють за рахунок застосування класів до елементів, які відповідають селекторам CSS, визначеним усередині доданих до папки wwwroot/lib/bootstrap файлів.

У розмітку було додано елемент link, атрибут href якого завантажує файл bootstrap.css та з папки wwwroot/lib/bootstrap/dist/css . За згодою пакети CSS та JavaScript від незалежних постачальників встановлюються в папку wwwroot/lib.

Після імпортування таблиць стилів Bootstrap залишилося стилізувати елементи. Клас text-center центрує вміст елемента та його дочірніх елементів. Клас btn стилізує елемент button, input або у вигляді симпатичної кнопки. а клас btn -primary вказує діапазон кольорів цієї кнопки.[17]

3.3 Розгортання веб-додатку у хмарному середовищі AWS

Для розгортання додатку доцільно використати сучасні засоби, які надає AWS (Amazon Web Services). Розглянемо основні компоненти, які можуть використовуватися при створенні та розгортанні веб-додатку.

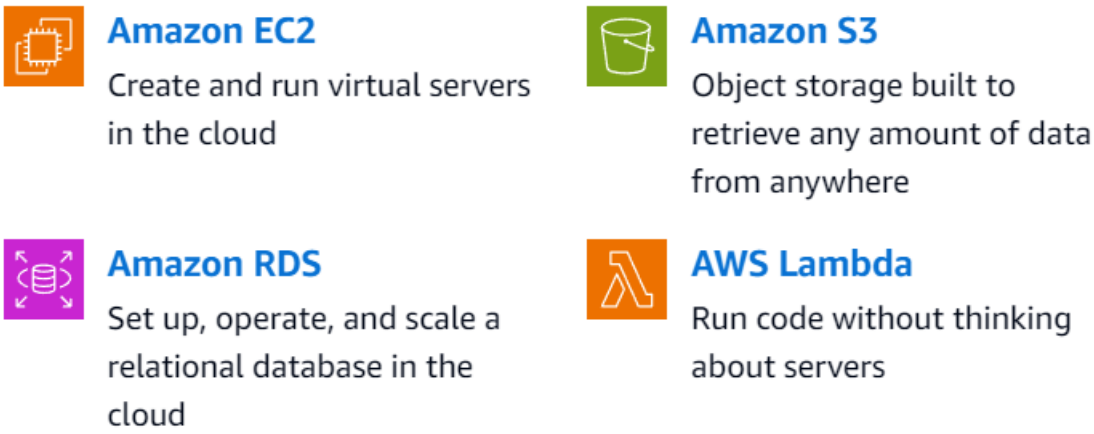


Рисунок 3.11 – Базові компоненти AWS

Amazon Elastic Compute Cloud (Amazon EC2) забезпечує масштабовану обчислювальну потужність за запитом у хмарі Amazon Web Services (AWS). Використання Amazon EC2 зменшує витрати на апаратне забезпечення, завдяки чому прискорується розробка та розгортання програми. Можна використовувати Amazon EC2, щоб запускати стільки віртуальних серверів, скільки потрібно, налаштовувати безпеку та мережу, а також керувати сховищем. Збільшити потужність для виконання важких обчислювальних завдань, таких як щомісячні чи річні процеси або стрибки відвідуваності веб-сайту. Коли використання зменшиться, є можливість знову зменшити ємність.

Коли запускається примірник, визначається апаратне забезпечення головного комп’ютера, який використовується для даного примірника. Кожен тип екземпляра пропонує різні можливості обчислення, пам’яті та зберігання та згрупований у сімейство екземплярів на основі цих можливостей.[15]

Amazon EC2 виділяє деякі ресурси головного комп’ютера, наприклад, ЦП, пам’ять і сховище екземплярів, для певного екземпляра. Amazon EC2 спільно використовує інші ресурси головного комп’ютера, наприклад мережу та дискову підсистему, серед екземплярів. Якщо кожен екземпляр на головному комп’ютері намагається використовувати якомога більше одного з цих спільних ресурсів, кожен отримує рівну частку цього ресурсу. Однак, коли ресурс використовується недостатньо, примірник може споживати більшу частку цього ресурсу, поки він доступний.

Кожен тип екземпляра забезпечує вищу або нижчу мінімальну продуктивність спільного ресурсу. Наприклад, типи екземплярів із високою продуктивністю введення-виведення мають більший розподіл спільних ресурсів. Виділення більшої частки спільних ресурсів також зменшує дисперсію продуктивності введення-виведення. Для більшості програм помірної продуктивності вводу-виводу більш ніж достатньо. Однак для

додатків, які вимагають більшої або стабільнішої продуктивності введення-виведення, потрібен тип екземпляра з вищою продуктивністю введення-виведення.

Amazon EC2 надає різноманітні типи екземплярів, тому мажна обрати тип, який найкраще відповідає вимогам. Типи екземплярів називаються на основі їх сімейства, покоління, сімейства процесорів, додаткових можливостей і розміру. Перша позиція назви типу екземпляра вказує на сімейство екземплярів. Друга позиція вказує на створення екземпляра. Третя позиція вказує на сімейство процесорів. Літери, що залишилися перед крапкою, позначають додаткові можливості, наприклад томи сховища примірників. Після крапки вказується розмір екземпляра, наприклад small або 4xlarge, або metal для екземплярів без використання металу.

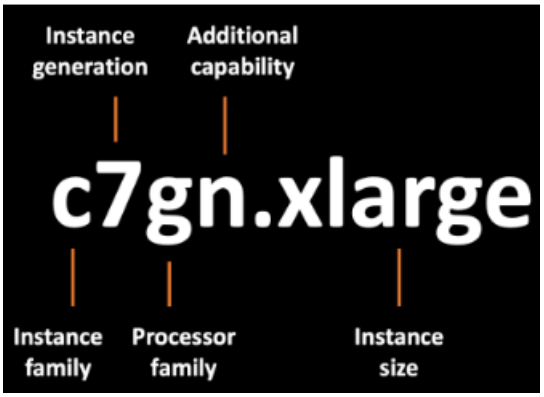


Рисунок 3.12 – Правила іменування інстансів

Сімейства екземплярів:

- **C** – оптимізоване обчислення;
- **D** – Щільне зберігання;
- **F** – FPGA;
- **G** – інтенсивна графіка;
- **Hpc** – високопродуктивне обчислення;
- **I** – оптимізоване зберігання;
- **Im** – оптимізоване сховище із співвідношенням віртуального ЦП до пам’яті один до чотирьох;
- **Є** – оптимізоване зберігання зі співвідношенням віртуального процесора до пам’яті один до шести;
- **Inf** – AWS Inferentia;
- **M** – загального призначення;
- **Mac** – macOS;
- **P** – прискорений GPU;
- **R** – оптимізовано пам’ять;
- **T** – стійка продуктивність;
- **Trn** – AWS Trainium;
- **U** – висока пам'ять;
- **VT** – перекодування відео;

- **X** – Інтенсивна пам’ять.

Сімейства процесорів:

- **a** – процесори AMD;
- **g** – процесори AWS Graviton;
- **i** – процесори Intel.

Додаткові можливості:

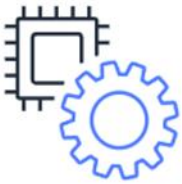
- **d** – томи сховища примірників;
- **n** – оптимізовано мережу та EBS;
- **e** – додаткове сховище або пам’ять;
- **z** – висока продуктивність;
- **q** – прискорювачі висновків Qualcomm;
- **flex** – екземпляр Flex.

За типами Інстанси поділяються на:

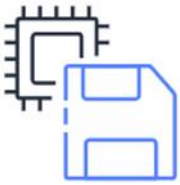
- загального призначення;
- оптимізовані для обичислювальних задач;
- оптимізовані для памяті;
- для прискорених обчислень;
- оптимізовані для зберігання;
- оптимізовані для високопродуктивних обчислень.

3.3.1 Огляд та вибір інстансів EC2

General-Purpose
Ideal for business critical applications, small and mid-sized databases, web tier applications, and more.



Compute Optimized
Ideal for high performance computing, batch processing, video encoding, and more.



Memory Optimized
Ideal for high performance databases, distributed web scale in-memory caches, real time big data analytics, and more.

Accelerated Computing
Ideal for machine learning, graphic intensive applications, gaming, and more.

Storage Optimized
Ideal for NoSQL databases, data warehousing, distributed file systems, and more.

Рисунок 3.13, 3.14 – Типи інстансів EC2

Екземпляри Amazon EC2, які працюють на процесорах Intel, можуть містити такі функції. Не всі з наведених нижче функцій процесора підтримуються всіма типами примірників.

Нові інструкції Intel AES (AES-NI) — набір інструкцій шифрування Intel AES-NI вдосконалює оригінальний алгоритм Advanced Encryption Standard (AES), щоб забезпечити швидший захист даних і більшу безпеку. Усі примірники EC2 поточного покоління підтримують цю функцію процесора.

- Intel Advanced Vector Extensions (Intel AVX, Intel AVX2 і Intel AVX-512) — Intel AVX і Intel AVX2 є 256-розрядними, а Intel AVX-512 — це 512-розрядне розширення набору інструкцій, призначене для програм із плаваючою комою (FP).) інтенсивний. Інструкції Intel AVX покращують продуктивність таких програм, як обробка зображень і аудіо/відео, наукове моделювання, фінансова аналітика, а також 3D-моделювання й аналіз. Ці функції доступні лише в екземплярах, запущених за допомогою HVM AMI.
- Технологія Intel Turbo Boost — процесори з технологією Intel Turbo Boost автоматично запускають ядра швидше, ніж базова робоча частота.
- Intel Deep Learning Boost (Intel DL Boost) — прискорює випадки глибокого навчання ШІ. Масштабовані процесори Intel Xeon 2-го покоління розширюють Intel AVX-512 новою інструкцією векторної нейронної мережі (VNNI/INT8), яка значно підвищує продуктивність логічного висновку в порівнянні з процесорами Intel Xeon Scalable попереднього покоління (з FP32) для розпізнавання/сегментації зображень, виявлення об’єктів, розпізнавання мовлення, переклад мови, системи рекомендацій, навчання з підкріпленням тощо. VNNI може бути не сумісний з усіма дистрибутивами Linux.

Наступні екземпляри підтримують VNNI: M5n, R5n, M5dn, M5zn, R5b, R5dn, D3, D3en, та C6i. C5i C5d екземпляри підтримують VNNI лише для екземплярів 12xlarge, 24xlargei .metal

Плутанина може виникнути через галузеві угоди про найменування 64-розрядних ЦП. Виробник мікросхем Advanced Micro Devices (AMD) представив першу комерційно успішну 64-розрядну архітектуру на основі набору інструкцій Intel x86. Отже, архітектуру широко називають AMD64 незалежно від виробника чіпа. Windows і кілька дистрибутивів Linux дотримуються цієї практики. Це пояснює, чому інформація про внутрішню систему екземпляра, що працює під керуванням Ubuntu або Windows, відображає архітектуру ЦП як AMD64, навіть якщо екземпляри працюють на обладнанні Intel.[16]

Рівень безкоштовного користування дозволяє впродовж 12 місяців використовувати 750 год/міс інстанси t2.micro або t3.micro з Linux, RHEL або SLES в залежності від регіону.

Інстанси Amazon EC2 T2 – це інстанси з продуктивністю, що підвищується, які забезпечують базовий рівень продуктивності ЦПУ з можливістю його підвищення.

Безлімітні інстанси T2 можуть підтримувати високу продуктивність ЦПУ до того часу, поки цього вимагає робоче навантаження. Більшість робочих навантажень загального призначення безлімітні інстанси T2 забезпечують достатню продуктивність без додаткової плати. Якщо потрібно виділити більше ресурсів ЦПУ на інстанс протягом тривалого часу, це можна зробити з оплатою за фіксованим тарифом: 5 центів за годину роботи віртуального ЦПУ.

Базовий рівень продуктивності та можливість його перевищення визначаються кредитами ЦПУ. Інстанси T2 регулярно одержують певну кількість кредитів, яка залежить від розміру інстансу. Вони накопичують кредити ЦПУ під час простою та споживають їх в активному стані. Інстанси T2 добре підходять для різних робочих навантажень загального призначення, включаючи мікросервіси, інтерактивні програми з низькою затримкою, малі та середні бази даних, віртуальні робочі столи, середовища розробки, збирання та тестування, репозиторії коду та прототипи продуктів.

Можливості:

- Процесор Intel Xeon Scalable із частотою до 3,3 ГГц (Haswell E5-2676 v3 або Broadwell E5-2686 v4);
- Високочастотні процесори Intel Xeon;
- Постійний базовий рівень продуктивності та ЦП з можливістю підвищення продуктивності (регулюється кредитами ЦП);
- Економічний тип інстансів загального призначення. Доступний на рівні безкоштовного користування;
- Збалансоване співвідношення обчислювальних, мережевих ресурсів та пам'яті.

Instance	vCPU*	CPU Credits / hour	Mem (GiB)	Storage	Network Performance
t2.nano	1	3	0.5	EBS-Only	Low
t2.micro	1	6	1	EBS-Only	Low to Moderate
t2.small	1	12	2	EBS-Only	Low to Moderate
t2.medium	2	24	4	EBS-Only	Low to Moderate
t2.large	2	36	8	EBS-Only	Low to Moderate
t2.xlarge	4	54	16	EBS-Only	Moderate
t2.2xlarge	8	81	32	EBS-Only	Moderate

Рисунок 3.15 – Інстанси t2

Інстанси Amazon EC2 T3 - це нове покоління універсальних інстансів з продуктивністю, що підвищується, які надають продуктивність ЦП базового рівня з можливістю підвищити її в будь-який момент на будь-який необхідний період часу. Інстанси T3 пропонують збалансоване поєднання

обчислювальних, мережевих ресурсів та пам'яті. Вони створені для додатків із помірним використанням ЦПУ з періодичними піками навантажень.

Коли робоче навантаження знаходиться нижче за базове порогове значення, інстанси T3 накопичують кредити ЦПУ. Один кредит ЦПУ надає інстансу T3 можливість за необхідності на хвилину підвищити продуктивність ядра ЦПУ до максимальної. У безлімітному режимі інстанси T3 можуть будь-якої миті підвищити продуктивність на будь-який необхідний період часу.

Можливості:

- Процесор Intel Xeon Scalable із тактовою частотою 3,1 ГГц (Skylake 8175M або Cascade Lake 8259CL);
- Постійний базовий рівень продуктивності та ЦП з можливістю підвищення продуктивності (регулюється кредитами ЦП);
- Безлімітний режим за промовчанням для забезпечення продуктивності в пікові періоди та можливість вибрати стандартний режим для прогнозованої щомісячної вартості;
- Робота на базі системи AWS Nitro, яка є поєднанням виділеного обладнання та компактного гіпервізора.

Instance	vCPU*	CPU Credits/hour	Mem (GiB)	Storage	Network Performance (Gbps)
t3.nano	2	6	0.5	EBS-Only	Up to 5
t3.micro	2	12	1	EBS-Only	Up to 5
t3.small	2	24	2	EBS-Only	Up to 5
t3.medium	2	24	4	EBS-Only	Up to 5
t3.large	2	36	8	EBS-Only	Up to 5
t3.xlarge	4	96	16	EBS-Only	Up to 5
t3.2xlarge	8	192	32	EBS-Only	Up to 5

Рисунок 3.16 – Інстанси t3

Усі віртуальні ЦПУ в інстансах Amazon EC2 на базі Graviton – це ядра процесора AWS Graviton.

Всі віртуальні ЦПУ в інстансах Amazon EC2, що не базуються на Graviton, є потоком процесора на основі архітектури x86, за винятком інстансів M7a, T2 та m3.medium.

AVX, AVX2 та покращена мережна конфігурація доступні лише на інстансах, запущених з використанням образів HVM AMI.

3.3.2 Огляд та вибір бакетів S3

Amazon Simple Storage Service (Amazon S3) — це служба зберігання об’єктів, яка пропонує найкращі в галузі масштабованість, доступність даних, безпеку та продуктивність. Клієнти будь-якого розміру та галузі можуть використовувати Amazon S3 для зберігання та захисту будь-якої кількості

даних для різноманітних випадків використання, таких як озера даних, веб-сайти, мобільні програми, резервне копіювання та відновлення, архівування, корпоративні програми, пристрої IoT та великі дані. аналітика. Amazon S3 надає функції керування, щоб можна було оптимізувати, упорядковувати та налаштовувати доступ до своїх даних відповідно до конкретних бізнес-вимог, організаційних вимог і вимог відповідності.[12]

Класи зберігання

Amazon S3 пропонує низку класів зберігання, розроблених для різних випадків використання. Наприклад, ви можете зберігати критично важливі виробничі дані в S3 Standard або S3 Express One Zone для частого доступу, заощаджувати кошти, зберігаючи рідко доступні дані в S3 Standard-IA або S3 One Zone-IA, і архівувати дані з найменшими витратами в S3 Glacier Instant Retrieval, S3 Glacier Flexible Retrieval і S3 Glacier Deep Archive.

Amazon S3 Express One Zone — це високопродуктивне однозонне сховище класу Amazon S3, яке спеціально створено для надання узгодженого доступу до даних із однозначною цифрою в мілісекундах для найбільш чутливих до затримок програм. S3 Express One Zone — це клас зберігання хмарних об’єктів з найнижчою затримкою, доступний на сьогодні, зі швидкістю доступу до даних до 10 разів швидшою, а витрати на запит на 50% нижчі, ніж S3 Standard. S3 Express One Zone — це перший клас зберігання S3, де ви можете вибрати одну зону доступності з можливістю розміщення об’єктного сховища разом із обчислювальними ресурсами, що забезпечує найвищу можливу швидкість доступу. Крім того, для подальшого збільшення швидкості доступу та підтримки сотень тисяч запитів на секунду дані зберігаються в новому типі відра: відро каталогу Amazon S3.

Amazon S3 — це служба зберігання об’єктів, яка зберігає дані як об’єкти в сегментах. Об’єкт – це файл і будь-які метадані, які описують файл. Bucket - це контейнер для предметів.

Щоб зберігати свої дані в Amazon S3, ви спочатку створюєте сегмент і вказуєте ім’я сегмента та регіон AWS. Потім ви завантажуєте свої дані в це відро як об’єкти в Amazon S3. Кожен об’єкт має ключ (або назву ключа), який є унікальним ідентифікатором об’єкта в сегменті.

S3 надає функції, які можна налаштувати для підтримки конкретного випадку використання. Наприклад, ви можете використовувати керування версіями S3, щоб зберігати кілька версій об’єкта в одному сегменті, що дозволяє відновлювати об’єкти, які випадково видалено або перезаписано.

Відра та об’єкти в них є приватними, і до них можна отримати доступ, лише якщо ви явно надасте дозволи на доступ. Для керування доступом можна використовувати політики сегментів, політики AWS Identity and Access Management (IAM), списки контролю доступу (ACL) і точки доступу S3.

Amazon S3 забезпечує надійну послідовність читання після запису для запитів PUT і DELETE об’єктів у вашому сегменті Amazon S3 у всіх регіонах AWS. Така поведінка стосується як запису в нові об’єкти, так і запитів PUT, які перезаписують існуючі об’єкти, і запитів DELETE. Крім того, операції

читання в Amazon S3 Select, списках керування доступом Amazon S3 (ACL), тегах об’єктів Amazon S3 і метаданих об’єктів (наприклад, об’єкт HEAD) є дуже узгодженими.[9]

Оновлення одного ключа є атомарними. Наприклад, якщо ви робите запит PUT до існуючого ключа з одного потоку та виконуєте запит GET для того самого ключа з другого потоку одночасно, ви отримаєте або старі дані, або нові дані, але ніколи не часткові або пошкоджені дані.

Amazon S3 забезпечує високу доступність завдяки тиражуванню даних на кількох серверах у центрах обробки даних AWS. Якщо запит PUT виконано успішно, ваші дані безпечно зберігаються. Будь-яке читання (запит GET або LIST), ініційоване після отримання успішної відповіді PUT, поверне дані, записані запитом PUT. Ось приклади такої поведінки:

- Процес записує новий об’єкт в Amazon S3 і негайно перераховує ключі в своєму сегменті. У списку з’явиться новий об’єкт.
- Процес замінює існуючий об'єкт і негайно намагається його прочитати. Amazon S3 повертає нові дані.
- Процес видаляє існуючий об’єкт і негайно намагається його прочитати. Amazon S3 не повертає жодних даних, оскільки об’єкт було видалено.
- Процес видаляє існуючий об’єкт і негайно перераховує ключі в його сегменті. Об'єкт не відображається в списку.

Одночасні програми

У цьому розділі наведено приклади очікуваної поведінки Amazon S3, коли кілька клієнтів записують в ті самі елементи.

У цьому прикладі і W1 (запис 1), і W2 (запис 2) закінчуються перед початком R1 (читання 1) і R2 (читання 2). Оскільки S3 дуже узгоджений, R1 і R2 обидва повертають color = ruby.

Рисунок 3.17 – Приклад поведінки Amazon S3

У наступному прикладі W2 не закінчується до початку R1. Тому R1 може повернути color = ruby або color = garnet. Однак, оскільки W1 і W2 закінчуються до початку R2, R2 повертається color = garnet.

Рисунок 3.18 – Приклад поведінки Amazon S3

В останньому прикладі W2 починається до того, як W1 отримає підтвердження. Тому ці записи вважаються одночасними. Amazon S3 внутрішньо використовує семантику last-writer-wins, щоб визначити, який запис має пріоритет. Однак порядок, у якому Amazon S3 отримує запити, і порядок, у якому програми отримують підтвердження, неможливо передбачити через різні фактори, наприклад затримку мережі. Наприклад, W2 може бути ініційований примірником Amazon EC2 у тому самому регіоні, тоді як W1 може бути ініційований хостом, який знаходиться далі. Найкращий спосіб визначити остаточне значення — виконати читання після підтвердження обох записів.

Рисунок 3.19 – Приклад поведінки Amazon S3

Фрагмент програми підключення до bucket:

```
/// <summary>
        /// Shows how to create a new Amazon S3 bucket.
        /// </summary>
        /// <param name="client">An initialized Amazon S3 client
object.</param>
        /// <param name="bucketName">The name of the bucket to
create.</param>
        /// <returns>A boolean value representing the success or
failure of
        /// the bucket creation process.</returns>
```

```
        public static async Task<bool> CreateBucketAsync(IAmazonS3
client, string bucketName)
        {
            try
            {
                var request = new PutBucketRequest
                {
                    BucketName = bucketName,
                    UseClientRegion = true,
                };

                var response = await client.PutBucketAsync(request);
                return response.HttpStatusCode ==
System.Net.HttpStatusCode.OK;
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error creating bucket:
'{ex.Message}'");
                return false;
            }
          }
```

Можна використовувати Amazon S3 для розміщення статичного веб-сайту. На статичному веб-сайті окремі веб-сторінки містять статичний вміст. Вони також можуть містити сценарії на стороні клієнта.

Навпаки, динамічний веб-сайт покладається на обробку на стороні сервера, включаючи сценарії на стороні сервера, такі як PHP, JSP або ASP.NET. Amazon S3 не підтримує сценарії на стороні сервера, але AWS має інші ресурси для розміщення динамічних веб-сайтів.

Використання AWS SDK для .NET

AWS SDK для .NET надає API для сегментів Amazon S3 і операцій з об’єктами. Для операцій з об’єктами, крім надання API для завантаження об’єктів за одну операцію, SDK надає API для завантаження великих об’єктів частинами.

API низького рівня

API низького рівня відповідають основним операціям Amazon S3 REST, включаючи операції створення, оновлення та видалення, які застосовуються до сегментів і об’єктів. Коли ви завантажуєте великі об’єкти за допомогою низькорівневого API багатокомпонентного завантаження, це забезпечує більший контроль. Наприклад, це дозволяє призупиняти та відновлювати багатокомпонентні завантаження, змінювати розміри частин під час завантаження або починати завантаження, коли ви заздалегідь не знаєте розмір даних. Якщо у вас немає цих вимог, використовуйте API високого рівня для завантаження об’єктів.[11]

API високого рівня

Для завантаження об’єктів SDK забезпечує вищий рівень абстракції, надаючи клас TransferUtility. API високого рівня — це простіший API, у якому лише за кілька рядків коду ви можете завантажувати файли та потоки на Amazon S3. Вам слід використовувати цей API для завантаження даних, якщо вам не потрібно контролювати завантаження, як описано в попередньому розділі API низького рівня.

Для меншого розміру даних TransferUtilityAPI завантажує дані за одну операцію. Однак TransferUtility перемикається на використання API багатокомпонентного завантаження, коли розмір даних досягає певного порогу. За замовчуванням він використовує кілька потоків для одночасного завантаження частин. Якщо завантаження частини не вдається, API повторює невдале завантаження частини щонайбільше три рази. Однак ці параметри можна налаштувати.

Організація .NET API

Під час написання програм Amazon S3 за допомогою AWS SDK для .NET ви використовуєте AWSSDK.dll. Наступні простори імен у цій збірці забезпечують багатокомпонентний API завантаження:

- Amazon.S3.Transfer — надає API високого рівня для завантаження ваших даних частинами.

Він містить TransferUtility клас, який дозволяє вказати файл, каталог або потік для завантаження ваших даних. Він також містить класи TransferUtilityUploadRequestта TransferUtilityUploadDirectoryRequest для налаштування додаткових параметрів, таких як кількість одночасних потоків, розмір частини, метадані об’єкта, клас зберігання (STANDARD, REDUCED_REDUNDANCY) і список контролю доступу до об’єктів (ACL).

- Amazon.S3 — забезпечує реалізацію API низького рівня. Він надає методи, які відповідають багатокомпонентному API завантаження Amazon S3 REST (див. Використання REST API).
- Amazon.S3.Model — надає класи API низького рівня для створення запитів і обробки відповідей. Наприклад, він надає класи InitiateMultipartUploadRequestand InitiateMultipartUploadResponse, які можна використовувати під час завантаження кількох частин, і класи UploadPartRequestand UploadPartResponse під час завантаження частин.
- Amazon.S3.Encryption— Надає AmazonS3EncryptionClient.
- Amazon.S3.Util — надає різні класи корисності, такі як AmazonS3Utili BucketRegionDetector.

Підключення для статичного сайту:

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
```

```
{
    class WebsiteConfigTest
    {
        private const string bucketName = "*** bucket name ***";
        private const string indexDocumentSuffix = "*** index object
key ***"; // For example, index.html.
        private const string errorDocument = "*** error object key
***"; // For example, error.html.
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
RegionEndpoint.USWest2;
        private static IAmazonS3 client;
        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            AddWebsiteConfigurationAsync(bucketName,
indexDocumentSuffix, errorDocument).Wait();
        }

        static async Task AddWebsiteConfigurationAsync(string
bucketName,
                                                string
indexDocumentSuffix,
                                                string errorDocument)
        {
            try
            {
                // 1. Put the website configuration.
                PutBucketWebsiteRequest putRequest = new
PutBucketWebsiteRequest()
                {
                    BucketName = bucketName,
                    WebsiteConfiguration = new WebsiteConfiguration()
                    {
                        IndexDocumentSuffix = indexDocumentSuffix,
                        ErrorDocument = errorDocument
                    }
                };
                PutBucketWebsiteResponse response = await
client.PutBucketWebsiteAsync(putRequest);

                // 2. Get the website configuration.
                GetBucketWebsiteRequest getRequest = new
GetBucketWebsiteRequest()
                {
                    BucketName = bucketName
                };
                GetBucketWebsiteResponse getResponse = await
client.GetBucketWebsiteAsync(getRequest);
                Console.WriteLine("Index document: {0}",
getResponse.WebsiteConfiguration.IndexDocumentSuffix);
```

```
                Console.WriteLine("Error document: {0}",
getResponse.WebsiteConfiguration.ErrorDocument);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server.
Message:'{0}' when writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server.
Message:'{0}' when writing an object", e.Message);
            }
        }
    }
{
```

Рівень безкоштовного користування дозволяє впродовж 12 місяців використовувати 5 ГБ стандартного сховища, 20000 запросів Get, 2000 запросів Put.

Ви платите за зберігання об’єктів у ваших відрах S3. Плата залежить від розміру ваших об’єктів, тривалості зберігання об’єктів протягом місяця та класу зберігання — S3 Standard, S3 Intelligent-Tiering, S3 Standard-Infrequent Access, S3 One Zone-Infrequent Access, S3 Express One Zone, S3 Glacier Instant Retrieval, S3 Glacier Flexible Retrieval (раніше S3 Glacier) і S3 Glacier Deep Archive. Ви сплачуєте щомісячну плату за моніторинг і автоматизацію за об’єкт, який зберігається в класі зберігання S3 Intelligent-Tiering, щоб контролювати шаблони доступу та переміщувати об’єкти між рівнями доступу. У S3 Intelligent-Tiering не стягується плата за пошук і не стягується додаткова плата за рівень доступу, коли об’єкти переміщуються між рівнями доступу.[6]

За використання PUT, COPY або правил життєвого циклу для переміщення даних у будь-який клас сховища S3 стягується плата за прийом. Перед переміщенням об’єктів у будь-який клас сховища враховуйте витрати на прийом або перехід. Є можливість оцінити витрати за допомогою калькулятора цін AWS.

3.3.3 Огляд та вибір засобів RDS для PostgreSQL

Amazon Relational Database Service (Amazon RDS) – це набір керованих сервісів, який спрощує налаштування, використання та масштабування бази даних у хмарі. Підтримується сім ядер: версія Amazon Aurora, сумісна з MySQL, версія Amazon Aurora, сумісна з PostgreSQL, MySQL, MariaDB, PostgreSQL, Oracle та SQL Server.

Рисунок 3.20 - Amazon RDS

В рамках рівня безкоштовного користування нові клієнти AWS можуть розпочати роботу із сервісом Amazon RDS безкоштовно. Рівень безкоштовного користування Amazon RDS включає наступний обсяг послуг, що надаються щомісяця протягом одного року.

- Використання Amazon RDS за місяць: 750 годин в окремих базах даних інстансів в одній зоні доступності. При використанні кількох інстансів дані про це збираються на основі їх типів. (Доступні двигуни: MySQL, MariaDB, PostgreSQL або SQL Server - тільки версія SQL Server Express.)
- Універсальний том SSD (gp2), сховище на місяць: 20 ГБ;
- Сховище автоматичного резервного копіювання баз даних на місяць: 20 ГБ.

БД PostgreSQL стала однією з найпопулярніших реляційних баз даних з відкритим вихідним кодом серед розробників великих компаній та стартапів. На її основі працює безліч мобільних рішень та додатків для бізнесу. Amazon RDS спрощує налаштування, експлуатацію та масштабування розгортання PostgreSQL у хмарі.[14]

Amazon RDS дозволяє всього за кілька хвилин здійснювати масштабоване та недороге розгортання PostgreSQL з можливістю налаштування обсягу апаратних ресурсів. Amazon RDS бере на себе складні та трудомісткі завдання з адміністрування, такі як встановлення та оновлення ПЗ PostgreSQL, управління сховищем, реплікація з метою забезпечення високої доступності та швидкості читання, а також резервне копіювання для аварійного відновлення.

Amazon RDS для PostgreSQL забезпечує доступ до можливостей відомого ядра БД PostgreSQL. Це означає, що код, програми та інструменти, які вже застосовуються з існуючими базами даних, можна використовувати із сервісом Amazon RDS. Amazon RDS для PostgreSQL в даний час підтримує PostgreSQL 9.6, 10, 11, 12, 13, 14 і 15. Пакет Trusted Language Extensions (TLE) для PostgreSQL дозволяє створювати високопродуктивні розширення та безпечно запускати їх в Amazon RDS, використовуючи популярні довірені мови сертифікації коду AWS.

Можливості

- Прості керовані розгортання. Сервіс Amazon RDS для БД PostgreSQL призначений для розробників та компаній, яким потрібен повний набір функціональних можливостей бази даних PostgreSQL або потрібно здійснити

перенесення існуючих програм та інструментів, які використовують базу даних PostgreSQL. Amazon RDS для PostgreSQL надає безпосередній доступ до стандартного ПЗ баз даних PostgreSQL, запущеному на інстансі БД Amazon RDS, що належить клієнту. Це забезпечує ефективну роботу програм.

- Попередньо налаштовані параметри. Для розгортань PostgreSQL в Amazon RDS попередньо налаштований точний набір параметрів та установок, що відповідає обраному класу інстансу БД. Залишається тільки запустити інстанс PostgreSQL та підключити програму. Процес займає всього кілька хвилин і не потребує додаткового налаштування. Додаткові можливості керування надані за допомогою груп параметрів БД.
- Моніторинг та метрики. Сервіс Amazon RDS надає доступ до метриків Amazon CloudWatch для інстансів БД, що використовуються, без додаткової плати. За допомогою Консолі керування AWS можна переглядати основні робочі метрики запущеного інстансу БД, включаючи використання обчислювальних ресурсів, пам'яті та сховища, інтенсивність операцій введення виводу та підключення до інстансу БД.
- Оповіщення про події БД: сервіс Amazon RDS надає можливість отримання оповіщень Amazon SNS про розгортання інстансу БД за допомогою електронної пошти або SMS. За допомогою Консолі керування AWS або API Amazon RDS можна передплатити більш ніж 40 різних подій БД, пов'язаних з розгортаннями в Amazon RDS.

Автоматичне оновлення. З використанням сервісу Amazon RDS можна не сумніватися в тому, що при розгортанні використовується найактуальніша версія PostgreSQL з усіма встановленими виправленнями. Система управління версіями ядра БД дозволяє налаштувати необхідність застосування виправлень кожному інстансі БД та його періодичність.

Швидке і надійне сховище

- Універсальні сховища (SSD). Універсальні сховища (SSD) Amazon RDS забезпечують не менше 3 операцій введення виводу за секунду на кожен виділений гігабайт і дозволяють виходити на піковий рівень до 3000 операцій введення виводу за секунду.Можна виконати перехід із магнітного сховища на універсальне (SSD), у своїй зниження доступності ресурсів буде лише короткочасним.Щоб отримати додаткові відомості та розпочати роботу з універсальним сховищем (SSD) Amazon RDS, вивчіть цей розділ посібника користувача Amazon RDS.[9]
- Provisioned IOPS (SSD) – це можливість виділення сховища об'ємом до 64 ТБ з кількістю операцій введення-виведення на секунду до 80 000 на кожну інстанс бази даних. Фактична кількість виконуваних IOPS може відрізнятись від виділеної кількості залежно від робочого навантаження бази даних, типу інстансу та обраного ядра БД. Див. Фактори, що впливають на кількість операцій, що фактично реалізуються, вводу-виводу в секунду посібника користувача Amazon RDS.
- Можна виконати перехід зі стандартного сховища на сховища з виділеним об'ємом IOPS, що забезпечить стабільну пропускну здатність та

малу затримку операцій введення виводу. За такого переходу спостерігатиметься короткочасне зниження доступності. Під час роботи можна без простоїв виконувати незалежне масштабування сховища та кількості IOPS (з кроком у 1000 операцій). Кількість IOPS можна масштабувати, збільшуючи або зменшуючи, наприклад, залежно від сезонних змін до трафіку додатків.

- Amazon DevOps Guru для RDS. Amazon DevOps Guru – це сервіс хмарних операцій на основі машинного навчання, який допомагає підвищити доступність програми. Amazon DevOps Guru для RDS дозволяє використовувати аналітичні дані, отримані за допомогою машинного навчання, для швидкого виявлення та діагностики проблем, пов'язаних із продуктивністю реляційної бази даних. Завдяки цьому сервісу час усунення таких проблем скорочується з кількох днів до кількох хвилин. Розробники та фахівці DevOps можуть за допомогою DevOps Guru для RDS автоматично діагностувати основну причину проблем із продуктивністю та отримувати обґрунтовані рекомендації для усунення цих проблем, не звертаючись за допомогою до експертів з баз даних.

Резервне копіювання та відновлення

- Автоматичне резервне копіювання. Увімкнена за замовчуванням в Amazon RDS можливість автоматичного резервного копіювання дозволяє відновлювати інстанс БД на час. Amazon RDS виконує резервне копіювання бази даних та логів транзакцій та зберігає їх протягом зазначеного користувачем періоду. Протягом цього періоду стан інстансу БД можна відновити будь-якої миті часу, до останніх п'яти хвилин використання. Термін зберігання автоматично створених резервних копій може становити до 35 днів.
- Знімки БД є резервними копіями інстансу БД, ініційовані користувачем. Ці повні резервні копії бази даних зберігаються в Amazon RDS безстроково, якщо ви не видалите їх навмисно. За бажання можна будь-якої миті створити новий інстанс БД зі знімка стану БД. Можна також копіювати знімки стану БД в інші регіони AWS з метою перенесення в інше географічне розташування або аварійного відновлення.

Зручність масштабування

- Клас інстансу БД. За допомогою API Amazon RDS або кількох клацань мишею в Консолі керування AWS можна масштабувати обчислювальні ресурси та ресурси пам'яті для існуючих розгортань, зменшуючи або збільшуючи їх обсяг. Масштабування зазвичай виконується за кілька хвилин.
- Обсяг сховища та кількість операцій введення виводу на секунду: у міру збільшення вимог до об'єму сховища виділяти додатковий обсяг можна під час роботи і без простоїв. Якщо ви використовуєте сховища RDS Provisioned IOPS, ви можете масштабувати пропускну здатність інстансу бази даних, задаючи кількість операцій введення виведення в секунду в діапазоні від 1000 до 80 000 з кроком 1000 та (або) об'єм сховища в діапазоні від 100 ГБ до 64 ТБ.

Реплікація

- Розгортання кількох зон доступності. Це варіант для розгортання інстансів робочої бази даних, який підвищує доступність і при цьому захищає останні оновлення бази даних від незапланованих відключень. При створенні або перекладі існуючого інстансу БД на використання кількох зон доступності Amazon RDS автоматично створює резервну копію в іншій зоні доступності (з незалежною інфраструктурою з фізично ізольованим розташуванням) та забезпечує керування нею. Оновлення бази даних для первинних та резервних ресурсів виконуються паралельно, щоб уникнути затримки реплікації. При проведенні запланованого технічного обслуговування, у разі виходу з ладу або відмови зони доступності Amazon RDS виконує автоматичне аварійне перемикання на резервний інстанс, що дозволяє продовжити роботу з базою даних відразу після завершення перемикання без втручання адміністратора. До перемикання прямий доступ до резервного інстансу недоступний, інстанс може бути використаний виконання операцій читання.
- Репліки читання. Ця можливість реплікації забезпечує просте еластичне масштабування з можливістю перевищення ємності одного інстансу бази даних виконання робочих навантажень із великою кількістю операцій читання. Для заданого інстансу БД можна створити одну або кілька реплік в межах одного регіону AWS для обслуговування трафіку додатків з великою кількістю операцій з читання різних копій, що дозволяє збільшити загальну пропускну здатність таких операцій. Для поширення змін, внесених у вихідний інстанс БД, за репліками читання сервіс Amazon RDS використовує вбудовані можливості реплікації PostgreSQL. Необхідно враховувати, що репліки читання використовують стандартну реплікацію PostgreSQL, тому можуть оновлюватися з відставанням від оригінальних інстансів бази даних.

Ізоляція та безпека

- Amazon RDS підтримує шифрування в базах даних PostgreSQL за допомогою ключів, керованих за допомогою AWS Key Management Service (KMS). В інстансі БД із шифруванням Amazon RDS шифруються всі дані, що знаходяться в базовому сховищі, а також автоматичні резервні копії, репліки читання та знімки стану.
- За допомогою Amazon Virtual Private Cloud (VPC) можна ізолювати інстанси БД в особистій віртуальній мережі та підключитися до існуючої IT-інфраструктури через з'єднання VPN, зашифроване за допомогою стандартного для галузі VPN підключення IPSec. Крім того, Amazon RDS дозволяє налаштувати параметри брандмауера та керувати мережевим доступом до інстансів БД.

Підтримувані можливості PostgreSQL

- PostGIS – це засіб просторового розширення баз даних для об’єктно реляційних БД PostgreSQL. Він забезпечує підтримку географічних об'єктів, завдяки чому можна запускати запити, розміщені в SQL. Розширення підтримки мов: PostgreSQL дозволяє за допомогою розширення завантажити

в базу даних процедурні мови. У PostgreSQL є чотири розширення підтримки мов для Perl, pgSQL, Tcl і JavaScript (з використанням ядра JavaScript V8).

- Словарі повнотекстового пошуку. PostgreSQL підтримує можливість повнотекстового пошуку, що дозволяє визначати документи на природних мовах, що відповідають умовам запиту, і при необхідності сортувати їх за релевантністю. Словарі не тільки підвищують якість пошуку, виконують нормалізацію та видаляють стоп слова, але й підвищують продуктивність виконання запитів.
- HStore, типи даних JSON. PostgreSQL підтримує тип даних JSON і дві функції JSON. Це дозволяє повертати дані JSON безпосередньо з бази даних сервера. PostgreSQL включає в себе розширення, яке інтегрує тип даних hstore для зберігання наборів за «ключ значення» в одному значенні PostgreSQL.
- Pg_stat_statements. Це розширення дозволяє відстежувати статистику виконання будь-яких заявок SQL, виконаних на момент (наприклад, ідентифікатор користувача), точно визначати, які запити виконуються, а також визначати загальне затрачене час.
- Упаковщики зовнішніх даних. Розширення postgres_fdw забезпечує доступ до даних, що зберігаються на інших серверах PostgreSQL, і дозволяє змінити ці дані точно так само, як дані на інстансі БД PostgreSQL в Amazon RDS.[10]

Мовні розширення для PostgreSQL

Trusted Language Extensions (TLE) для PostgreSQL – це комплект розробника та проект із відкритим вихідним кодом, який дозволяє швидко створювати високопродуктивні розширення та безпечно запускати їх в Amazon RDS без необхідності отримання сертифіката AWS на код. Розробники можуть використовувати популярні довірені мови, наприклад JavaScript, PL/pgSQL, Perl і SQL, для безпечного напису розширення коду. Назначення TLE – передбачити доступ до небезпечних ресурсів і обмежити дефекти розширення одним підключенням до бази даних. Адміністратори бази даних отримують можливість точного онлайн-контролю, що дозволяє вирішити, кому дозволено встановити розширення, і можуть створити модель дозволів для їх запуску. Решення TLE доступні для клієнтів Amazon RDS без додаткової плати.

Розгортання Amazon RDS без переривання в обслуговуванні

Розгортання Amazon RDS без переривання в обслуговуванні дозволяє оновлювати бази даних в Amazon RDS для PostgreSQL безпечніше, швидше, швидше і без втрати даних. За кілька кроків розвертання без переривання в обслуговуванні створюють проміжне середовище, яке є дзеркальним відображенням виробничого середовища, і синхронізує ці дві середовища за допомогою логічної реплікації. Ви можете вносити зміни, наприклад оновлювати основні та додаткові версії, модифікувати схеми та змінювати значення параметрів без зниження продуктивності вашої робочої навантаження.

Під час просування промежуткової середовища розвертання без переривання в обслуговуванні блокують запис як у середовищі з поточною версією додатка, так і в середовищі з новою його версією до завершення переключення. Розвертання без переривання в обслуговуванні використовують обмеження переключень, які відключають просування, якщо воно вище максимально допустимого часу простого, знаходять помилки реплікації, перевіряють стан інстанцій і багато іншого.[14]

Робота з Amazon RDS для PostgreSQL

Amazon RDS дозволяє використовувати Консоль управління AWS або простий набір API веб-сервісів для створення, видалення та зміни інстанцій реляційної бази даних (інстанцій БД). Додатково можна контролювати доступ і безпеку інстансів (інстанцій) БД і керувати резервним копіюванням БД і створенням знімків її стану. Повний список доступних API Amazon RDS див. в руководстве по API Amazon RDS. Нижче перераховані деякі часто використовувані API та їх можливості.

- CreateDBInstance. Дозволяє створити новий інстанс БД, вказавши необхідні параметри: ядро PostgreSQL, клас інстансу БД, об’єм хранилища, версію ядра БД (необов’язково) і політику зберігання резервних копій. Крім того, можна вказати, чи слід використовувати розвертання у багатьох зонах доступності для цього екземпляра БД. Цей виклик API надає всі необхідні параметри доступу до діючої БД PostgreSQL з попередньо встановленим ПО та вказаним обсягом ресурсів.
- ModifyDBInstance. Дозволяє змінювати налаштування працюючого інстанса БД. За допомогою цього виклику API можна масштабувати доступні для інстанси ресурси BD у відповідь на зміну навантаження на базу даних або змінювати порядок резервного копіювання та обслуговування за своїм переглядом, а також перейти на розвертання інстанси в кількох зонах доступності або, навпаки, повернутися до використаного одного AZ . Цей інструмент також можна використовувати в якості додаткових засобів оновленої версії PostgreSQL для інстанси BD, що дозволяє підтримувати сумісність з конкретними версіями баз даних, тестувати нові версії з контрольним додатком перед робочим розвертанням, а також виконувати оновлення на необхідних умовах і в задані терміни.
- DeleteDBInstance. Удаляє працюючий інстанс БД. Amazon RDS дозволяє в будь-який момент встановити роботу інстансу БД і платити тільки за фактично використані ресурси.
- Створити знімок DBS. Створює знімок стану інстанса БД. Дозволяє в будь-який момент відновити інстанси БД за створеними користувачами знімками стану. Таким чином можна відновити навіть видалений попередній інстанс БД.
- RestoreDBInstanceToPointInTime. Створює нові інстанси БД з резервної копії на момент часу. Дозволяє вивести відновлення стану на будь-який момент у межах зазначеного періоду зберігання резервних копій; як правило,

відновлення можливо ввести до останніх п'яти хвилин використання бази даних.

- CreateDBInstanceReadReplica. Створює інстанс БД, що працює в якості репліки чтення вихідного інстансу БД.

Міграція в Amazon RDS для PostgreSQL

Якщо програма вже використовує базу даних PostgreSQL, імпортувати дані в Amazon RDS досить просто. Для перенесення даних в Amazon RDS потрібно:

- створити інстанс БД із затребуваними вичислювальними ресурсами, ємністю хранилища та правилами доступу;
- створити копію даних для імпорту за допомогою функції pg_dump;
- за допомогою psql створити базу даних на інстансі БД і завантажити дані;
- оновити рядок підключення бази даних у файлі конфігурації додатка.

Інстанси БД Amazon RDS для PostgreSQL, що використовують версію PostgreSQL 9.3.5 і новіші, підтримують роль реплікації сесії. Імпортувати дані на екземпляри БД Amazon RDS для PostgreSQL з мінімальним часом просто можна також, використовуючи данну роль і запущені за допомогою інструментів реплікації подій з відкритим вихідним кодом, наприклад Londiste.

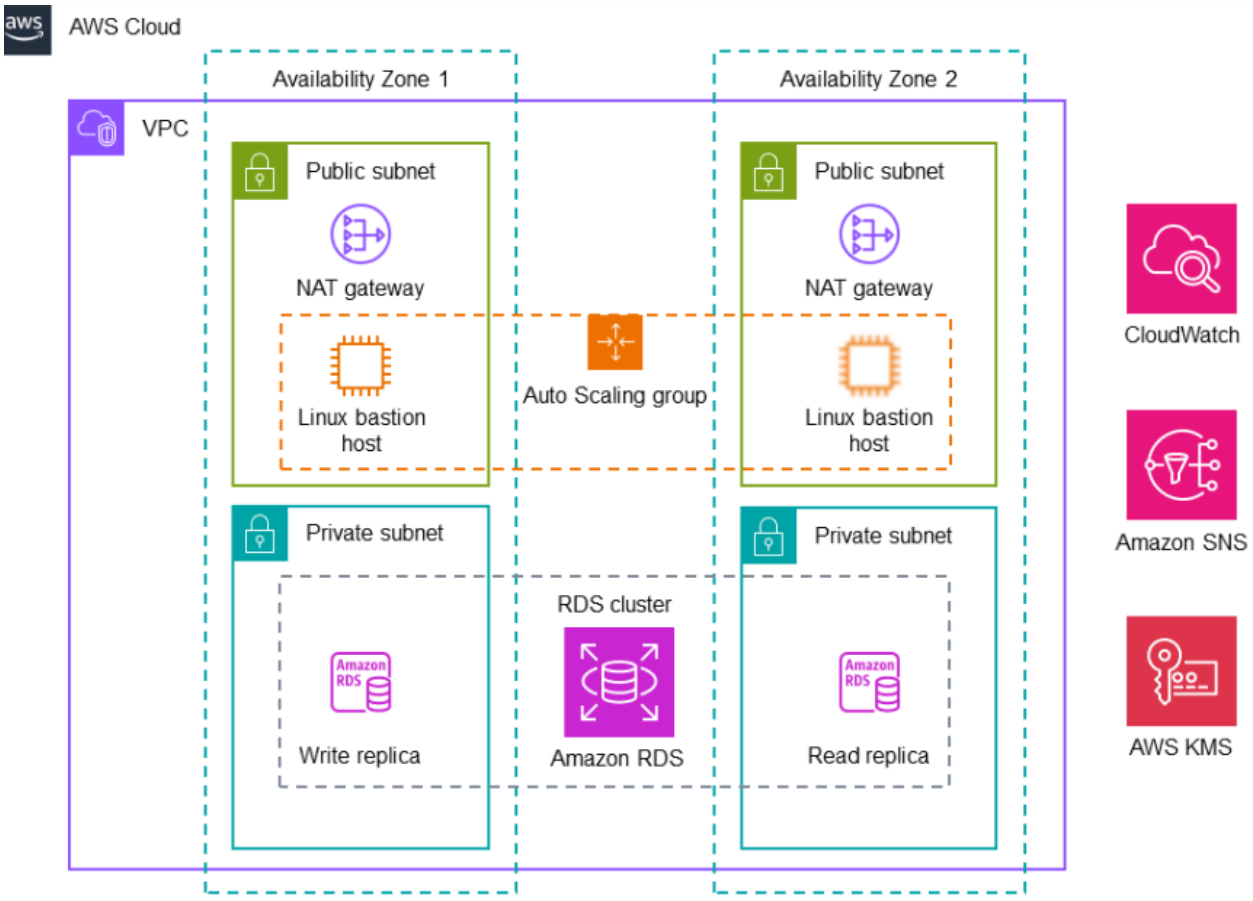


Рисунок 3.21 – Приклад розгортання Amazon RDS для PostgreSQL

Це рішення встановлює наступне:

- Високодоступна архітектура, яка охоплює дві зони доступності.*
- Віртуальна приватна хмара (VPC), налаштована з загальнодоступними та приватними підмережами відповідно до найкращих практик AWS, щоб надати вам власну віртуальну мережу на AWS.
- У публічних підмережах:
  1) Керовані шлюзи NAT для забезпечення вихідного доступу до Інтернету для ресурсів у приватних підмережах.*
  2) Додатковий бастіонний хост Linux у групі автоматичного масштабування, щоб дозволити вхідний доступ SSH (Secure Shell) до примірників Amazon Elastic Compute Cloud (Amazon EC2) у загальнодоступних і приватних підмережах.
- У приватних підмережах:
  1) Кластер Amazon RDS, який містить репліку для запису в одній зоні доступності та репліку для читання в іншій.
- Метрики CloudWatch для моніторингу екземпляра бази даних і журнали CloudWatch для зберігання журналів бази даних.
- Тема Amazon Simple Notification Service (Amazon SNS) для надсилання сповіщень про тривогу CloudWatch і подій Amazon RDS.
- Служба керування ключами AWS (AWS KMS) для шифрування даних, що зберігаються в екземплярі бази даних.[6]

Висновки за розділом

Веб-додаток розроблявся на основі технології ASP.NET MVC.

ASP.NET - це платформа веб-розробки, яка надає модель програмування, комплексну інфраструктуру програмного забезпечення та різноманітні служби, необхідні для створення надійних веб-додатків для ПК, а також мобільних пристроїв. Для розгортання проекту у хмарному середовищі використано сучасні засоби Amazon Web Services.

Розглянуто сервіси RDS для PostgreSQL. Amazon Relational Database Service (Amazon RDS) – це набір керованих сервісів, який спрощує налаштування, використання та масштабування бази даних у хмарі.

РОЗДІЛ 4. ДОСЛІДЖЕННЯ ОПТИМАЛЬНОСТІ ВИБОРУ ЗАСОБІВ РОЗРОБКИ САЙТУ

4.1 Опис загальної методології досліджень. Метод суміщеного ідеалу

Метод суміщеного ідеалу призначений для виділення одного або підмножини найбільш переважних об'єктів. Характерними рисами методу є:

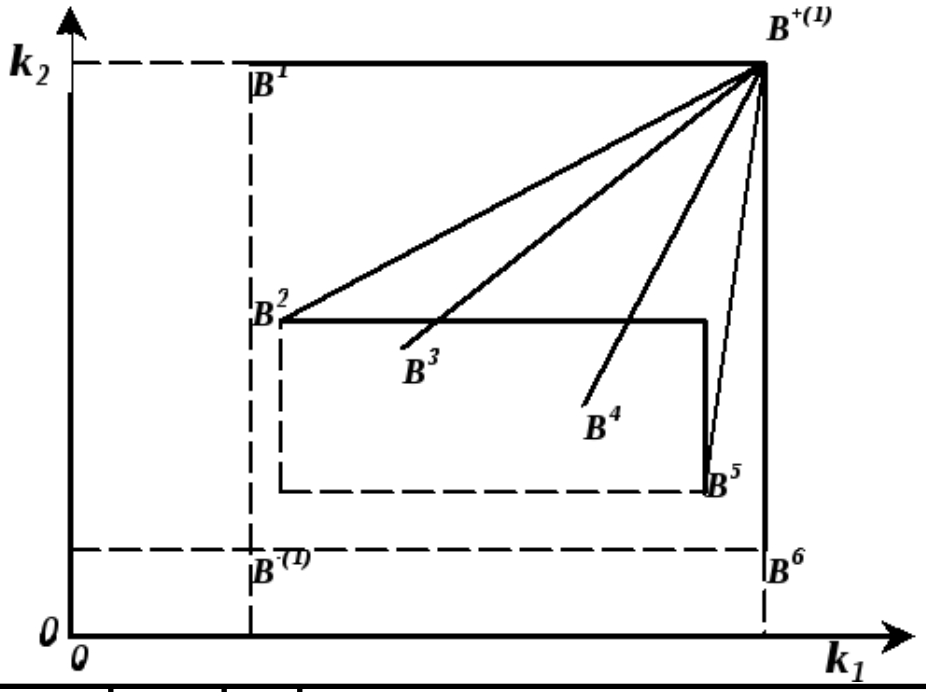
1. наявність процедури формування “ідеального” об'єкта (B^+), що служить своєрідною метою. Такий “ідеал”, зазвичай, недосяжний, але його корисно мати для розуміння своїх цілей;
2. на кожній ітерації виробляється виняток об'єктів, які не претендують на кращі, тобто не виділяються “найкращі” об'єкти, а виключаються “найгірші”.

У загальному вигляді алгоритм методу наступний (рис 2.1): спочатку виключаються домінovані об'єкти, оскільки серед них не може бути кращих.

Формується “ідеальний” об'єкт $B^{+(1)}$ з найбільш переважних значень критеріїв і “антиідеальний” з найменш переважних значень. Визначаються відстані від об'єктів з вихідної множини до “антиідеалу”, на підставі яких виділяються “найгірші” об'єкти. Серед таких об'єктів, як правило, є об'єкти, що мають одне найбільш переважне значення (об'єкти B^1 і B^6 на Рисунок 2.1).

Після виключення “гірших” об'єктів знову переходимо до етапу формування “ідеалу”, і він змінюється (на рисунку це $B^{+(2)}$), наближаючись до реальних об'єктів.

Процедура закінчується, коли залишається невелика кількість об'єктів, які і вважаються найкращими.[8]



					КНУ.РМ.123.23.19.ДВЗСС			
Змн.	Арк.	№ документа	Підпис	Дата				
Розробив		Шевченко			ДОСЛІДЖЕННЯ ВИБОРУ ЗАСОБІВ РОЗРОБКИ САЙТУ	Літера	Аркуш	Аркушів
Перевірив		Сьомочкина						
						КІ-22М		
Н.контроль		Кузнєцов						
Затвердив		Купін						

Рисунок 4.1 – Ілюстрація алгоритму методу суміщеного ідеалу

Слід зазначити, що при порівнянні реально існуючих об'єктів з “ідеалом” виникає незадоволеність, викликана недоступністю сформованого “ідеалу”. Таку незадоволеність називають конфліктом перед вирішенням.

Після вибору найкращого об'єкта виникає незадоволеність, викликана тим фактом, що обрано саме цей об'єкт, а не інший. Таку незадоволеність називають конфліктом після вирішення.

На перших ітераціях методу переважає конфлікт перед розв'язанням. На наступних ітераціях "ідеал" наближається до реальних об'єктів і конфлікт перед вирішенням зменшується. Проте конфлікт після вирішення може зростати.

Розглянемо докладно алгоритм методу, блок-схему якого наведено на Рисунок2.2.

Нехай вихідна множина об'єктів включає n-об'єктів. Усі критерії $k_j(j=1,\ldots, m)$ вимірюються за шкалою інтервалів або відносин.

На першому етапі формується “ідеальний” об'єкт $(k_1^+,\ldots, k_m^+)$, де k_j^+-максимальне за перевагою значення критерію серед усіх об'єктів, тобто $k_j^+ = \max_i k_j^i$, якщо перевага об'єкта зростає при збільшенні k_j, або $k_j^+ = \min_i k_j^i$, якщо перевага об'єкта зростає при зменшенні критерію. Якщо “ідеал” належить безлічі об'єктів, то він і буде найкращим. Але оскільки багатокритеріальна задача зазвичай вирішується на множині ефективних об'єктів, то “ідеальний” об'єкт не буде належати вихідній множині.

На цьому ж етапі формується “найгірший” об'єкт $(k_1^-,\ldots, k_m^-)$ із найменш переважних значень.

На другому етапі здійснюється перехід від фізичних одиниць виміру критеріїв до відносних відповідно до виразу:

$$d_j^i = (k_j^i - k_j^-)/(k_j^+ - k_j^-)$$

У відносних одиницях всі критерії будуть змінюватися в інтервалі [0;1], при цьому, чим менше d_j^i, тим ближче об'єкт за критерієм k_j до “антиідеального”.

Перші два етапи виконуються автоматично. На третьому етапі, виходячи зі своїх суджень про важливість критеріїв, потрібно задати ваги критеріїв W_j (j = 1,...,m).

Якщо виникають ускладнення, можна скористатися інформаційним підходом до визначення важливості критеріїв. На четвертому етапі розраховуються відстані об'єктів до “антиідеалу”. В якості метрики використовується наступний вираз:

$$L_i^p = \left[\sum_{j=1}^{m} W_j (d_j^i)^p\right]^{-p}$$

Використовуючи різні p, можна отримати різні метрики. Так, при p = 1, отримаємо адитивний оператор, а при p→∞ формула матиме вигляд

$$L_i^{\infty} = \max_i \left| W_j d_j^i \right|$$

Чим більше значення L_i^p, тим далі об'єкт від “антиідеалу” і ближче до “ідеального”.

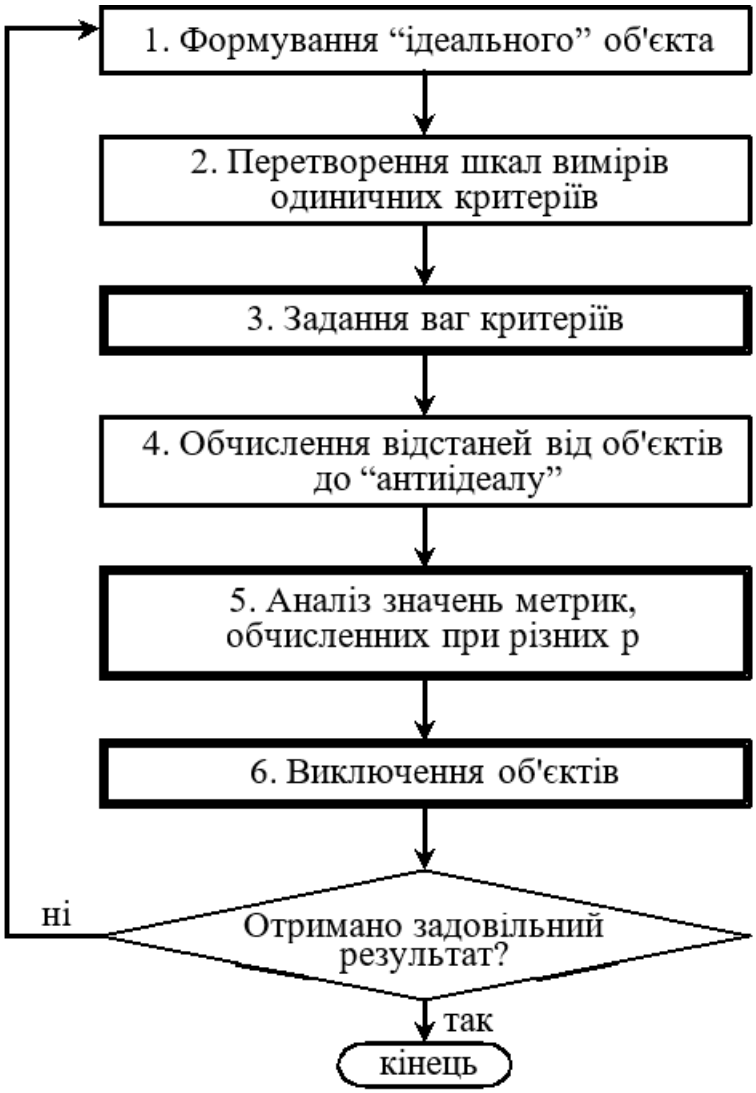


Рис .4.2 – Блок-схема методу суміщеного ідеалу

Слід зазначити, що як метрика для порівняння об'єкта з “ідеальним” можна використовувати й інші оператори агрегування.

На наступному, п'ятому етапі, задаючи різні значення p, визначаються різні метрики для порівняння з “ідеальним”. При кожному p, тобто для кожної метрики, всі об'єкти впорядковуються близько до “ідеалу” за величиною L_i^p. Змінюючи p, можна досліджувати вплив різних метрик на впорядкування об'єктів.[9]

Далі, на шостому етапі приймається рішення про виключення об'єктів, які не претендують на найкращий. Вочевидь, що це об'єкти, які за різних метриках (різних p) перебувають у кінці впорядкованих рядів. Якщо незалежно від обраної метрики об'єкт далекий від “ідеалу”, то є підстави виключити його.

Після виключення об'єктів починається наступна ітерація з формування “ідеального” об'єкта вже на підмножині об'єктів, що залишилися.

Закінчується процедура, коли після чергового виключення залишилося невелика кількість об'єктів, які будуть найкращими.

Слід зазначити, що з кожної ітерації доцільно аналізувати розкид критеріїв. Справа в тому, що серед об'єктів, що виключаються, як правило, є об'єкти, що включають максимальні і мінімальні значення критеріїв. Тим самим на кожній ітерації зменшується область зміни критеріїв і суттєво змінюється їхній розкид. Тоді, використовуючи інформаційний підхід, можна виділити неінформативні критерії та з метою спрощення завдання виключити такі критерії.

4.2 Опис методики проведених досліджень

1. Сформуємо декілька об’єктів, як варіанти можливої реалізації веб-додатку

У якості критеріїв оцінки проектів оберемо критерії, які максимізуються(к-сть сторінок сайту, к-сть зареєстрованих користівачів, к-сть проектів), та критерії, які мінімузуються(тривалість розробки, вартість). Дані наведено у Таблиця1.1

Таблиця 1.1 – Варіанти реалізації веб-додатку

	A	B	C	D	E	F	G	H	I
1	критерії	варіант 1	варіант 2	варіант 3	варіант 4	варіант 5	варіант 6	варіант 7	варіант 8
2	к-сть сторінок	1	2	4	5	5	8	8	10
3	к-сть зареєстрованих користувачів	100	500	200	200	500	400	800	1000
4	к-сть проектів	20	100	50	50	100	100	150	200
5	тривалість розробки(год)	100	200	400	300	300	200	300	400
6	вартість($)	4000	6000	5000	8000	9000	10000	12000	15000
7	авторизація користувача через E-mail або соцмережі	0	1	0	1	1	1	1	1
8	можливість переключення мови	0	1	0	1	0	1	1	1
9	інтеграція гугланалітики	0	0	0	1	0	1	1	1

2. Виконаємо процедуру формування ідеального об’єкта з найбільш переважних значень критеріїв.

Таблиця 1.2 – Показники ідеального об’єкта

	A	J
1	критерії	ідеальний об'єкт
2	к-сть сторінок	10
3	к-сть зареєстрованих користувачів	1000
4	к-сть проектів	200
5	тривалість розробки(год)	100
6	вартість($)	4000
7	авторизація користувача через E-mail або соцмережі	1
8	можливість переключення мови	1
9	інтеграція гугланалітики	1

Та найгіршого об’єкта з найменш переважних значень.

Таблиця 1.3 – Показники антиідеального об’єкта

	A	K
1	критерії	найгірший об'єкт
2	к-сть сторінок	1
3	к-сть зареєстрованих користувачів	100
4	к-сть проектів	20
5	тривалість розробки(год)	400
6	вартість($)	15000
7	авторизація користувача через E-mail або соцмережі	0
8	можливість переключення мови	0
9	інтеграція гугланалітики	0

3. Здійснюємо перехід від фізичних одиниць виміру критеріїв до відносних відповідно до виразу:

$$d_{i,j} = \frac{k_j^+ - k_{i,j}}{k_j^+ - k_j^-}$$

Таблиця 1.4 – Відносні значення критеріїв(розрахунок)

критерії	варіант 1	варіант 2	варіант 3	варіант 4	варіант 5	варіант 6	варіант 7	варіант 8
к-сть сторінок	=($J2-B2)/($J2-$K2)	=($J2-C2)/($J2-$K2)	=($J2-D2)/($J2-$K2)	=($J2-E2)/($J2-$K2)	=($J2-F2)/($J2-$K2)	=($J2-G2)/($J2-$K2)	=($J2-H2)/($J2-$K2)	=($J2-I2)/($J2-$K2)
к-сть зареєстрованих користувачів	=($J3-B3)/($J3-$K3)	=($J3-C3)/($J3-$K3)	=($J3-D3)/($J3-$K3)	=($J3-E3)/($J3-$K3)	=($J3-F3)/($J3-$K3)	=($J3-G3)/($J3-$K3)	=($J3-H3)/($J3-$K3)	=($J3-I3)/($J3-$K3)
к-сть проектів	=($J4-B4)/($J4-$K4)	=($J4-C4)/($J4-$K4)	=($J4-D4)/($J4-$K4)	=($J4-E4)/($J4-$K4)	=($J4-F4)/($J4-$K4)	=($J4-G4)/($J4-$K4)	=($J4-H4)/($J4-$K4)	=($J4-I4)/($J4-$K4)
тривалість розробки(год)	=($J5-B5)/($J5-$K5)	=($J5-C5)/($J5-$K5)	=($J5-D5)/($J5-$K5)	=($J5-E5)/($J5-$K5)	=($J5-F5)/($J5-$K5)	=($J5-G5)/($J5-$K5)	=($J5-H5)/($J5-$K5)	=($J5-I5)/($J5-$K5)
вартість($)	=($J6-B6)/($J6-$K6)	=($J6-C6)/($J6-$K6)	=($J6-D6)/($J6-$K6)	=($J6-E6)/($J6-$K6)	=($J6-F6)/($J6-$K6)	=($J6-G6)/($J6-$K6)	=($J6-H6)/($J6-$K6)	=($J6-I6)/($J6-$K6)
авторизація користувача через E-mail або соцмережі	=($J7-B7)/($J7-$K7)	=($J7-C7)/($J7-$K7)	=($J7-D7)/($J7-$K7)	=($J7-E7)/($J7-$K7)	=($J7-F7)/($J7-$K7)	=($J7-G7)/($J7-$K7)	=($J7-H7)/($J7-$K7)	=($J7-I7)/($J7-$K7)
можливість переключення мови	=($J8-B8)/($J8-$K8)	=($J8-C8)/($J8-$K8)	=($J8-D8)/($J8-$K8)	=($J8-E8)/($J8-$K8)	=($J8-F8)/($J8-$K8)	=($J8-G8)/($J8-$K8)	=($J8-H8)/($J8-$K8)	=($J8-I8)/($J8-$K8)
інтеграція гугланалітики	=($J9-B9)/($J9-$K9)	=($J9-C9)/($J9-$K9)	=($J9-D9)/($J9-$K9)	=($J9-E9)/($J9-$K9)	=($J9-F9)/($J9-$K9)	=($J9-G9)/($J9-$K9)	=($J9-H9)/($J9-$K9)	=($J9-I9)/($J9-$K9)

Таблиця 1.5 – Відносні значення критеріїв(значення)

критерії	варіант 1	варіант 2	варіант 3	варіант 4	варіант 5	варіант 6	варіант 7	варіант 8
к-сть сторінок	1	0,88889	0,666667	0,555556	0,555556	0,22222	0,222222	0
к-сть зареєстрованих кор	1	0,55556	0,888889	0,888889	0,555556	0,66667	0,222222	0
к-сть проектів	1	0,55556	0,833333	0,833333	0,555556	0,55556	0,277778	0
тривалість розробки(год)	0	0,33333	1	0,666667	0,666667	0,33333	0,666667	1
вартість($)	0	0,18182	0,090909	0,363636	0,454545	0,54545	0,727273	1
авторизація користувача через E-mail або соцмережі	1	0	1	0	0	0	0	0
можливість переключення мови	1	0	1	0	1	0	0	0
інтеграція гугланалітики	1	1	1	0	1	0	0	0

4. Задамо вагу критеріїв враховуючи їх важливість.

Таблиця 1.6 – Вага критеріїв

критерії	вага критерію W
к-сть сторінок	2
к-сть зареєстрованих користувачів	2
к-сть проектів	4
тривалість розробки(год)	1
вартість($)	8
авторизація користувача через E-mail або соцмережі	1
можливість переключення мови	1
інтеграція гугланалітики	2

5. Розрахуємо метрику, прийнявши що p=1:

$$L_i^p = \left[\sum_{j=1}^{m} W_j (d_j^i)^p \right]^{\frac{1}{p}}$$

Таблиця 1.7 – Розрахунок метрик для різних варіантів

критерії	варіант 1
метрика L	=B13*$B26+B14*$B27+B15*$B28+B16*$B29+B17*$B30+B18*$B31+B19*$B32+B20*$B33

Таблиця 1.8 – Значення метрик для різних варіантів

критерії	варіант 1	варіант 2	варіант 3	варіант 4	варіант 5	варіант 6	варіант 7	варіант 8
метрика L	12	8,89899	12,17172	9,79798	11,74747	8,69697	8,484848	9

6. Побудуємо графічне представлення отриманих результатів для критеріїв тривалості розробки та вартості.

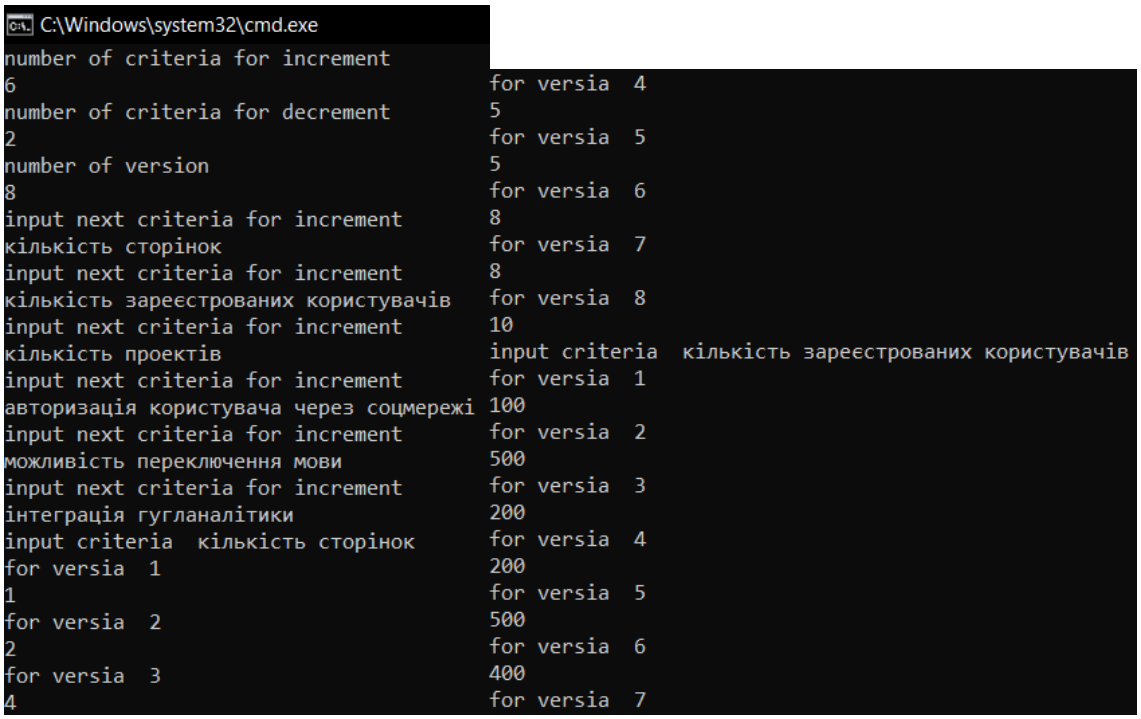
7. Проаналізуємо отримані результати. Як видно з Таблиця 1.8, найгіршим варіантом є 3, а найкращим є 7 варіант тому, що він має найменшу відстань до ідеального об’єкта.

4.3 Розробка програмного забезпечення для реалізації методу суміщеного ідеалу

Для здійснення автоматизованого багатокритеріального експертного вибору створимо власну програмну реалізацію вищезазначеного методу. У

якості програмного середовища обрано об’єктно-орієнтовану мову програмування Python. Приклад такої програми наведено в додатках.

Вихідними даними для програми служать: кількість і назви критеріїв порівняння, кількість і назви варіантів порівняння, фактичні значення критеріїв для кожного варіанту й коефіцієнти відносної важливості для кожного критерію порівняння. У результаті обчислень програма визначає кращий варіант, виводить його назву на екран та записує у текстовий документ research.txt.



```
800
for versia  8
1000
input criteria  кількість проектів
for versia  1
20
for versia  2
100
for versia  3
50
for versia  4
50
for versia  5
100
for versia  6
100
for versia  7
150
for versia  8
200
input criteria  авторизація користувача через соцмережі
for versia  1
0
for versia  2
1
for versia  3
0
for versia  4
1
for versia  5
1
for versia  6
1
for versia  7
1
for versia  8
1
input criteria  можливість переключення мови
for versia  1
0
for versia  2
1
for versia  3
0
for versia  4
1
for versia  5
0
for versia  6
1
for versia  7
1
for versia  8
1
input criteria  інтеграція гугланалітики
for versia  1
0
for versia  2
0
for versia  3
0
for versia  4
1
for versia  5
0
for versia  6
1
for versia  7
1
for versia  8
1
input next criteria for decrement
тривалість розробки
input next criteria for decrement
вартість
input criteria  тривалість розробки
for versia  1
100
for versia  2
200
for versia  3
400
for versia  4
300
for versia  5
300
for versia  6
200
for versia  7
300
for versia  8
400
input criteria  вартість
for versia  1
4000
for versia  2
6000
for versia  3
5000
for versia  4
8000
for versia  5
9000
for versia  6
10000
for versia  7
12000
for versia  8
15000
кількість сторінок [1.0, 2.0, 4.0, 5.0, 5.0, 8.0, 8.0, 10.0] ideal =  10.0 antiideal =  1.0
кількість зареєстрованих користувачів [100.0, 500.0, 200.0, 200.0, 500.0, 400.0, 800.0, 1000.0] ideal =  1000.0 antiideal =  100.0
кількість проектів [20.0, 100.0, 50.0, 50.0, 100.0, 100.0, 150.0, 200.0] ideal =  200.0 antiideal =  20.0
авторизація користувача через соцмережі [0.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0] ideal =  1.0 antiideal =  0.0
можливість переключення мови [0.0, 1.0, 0.0, 1.0, 0.0, 1.0, 1.0, 1.0] ideal =  1.0 antiideal =  0.0
інтеграція гугланалітики [0.0, 0.0, 0.0, 1.0, 0.0, 1.0, 1.0, 1.0] ideal =  1.0 antiideal =  0.0
тривалість розробки [100.0, 200.0, 400.0, 300.0, 300.0, 200.0, 300.0, 400.0] ideal =  100.0 antiideal =  400.0
вартість [4000.0, 6000.0, 5000.0, 8000.0, 9000.0, 10000.0, 12000.0, 15000.0] ideal =  4000.0 antiideal =  15000.0
relative criteria values:
кількість сторінок [1.0, 0.8888888888888888, 0.6666666666666666, 0.5555555555555556, 0.5555555555555556, 0.2222222222222222, 0.2222222222222222, 0.0]
кількість зареєстрованих користувачів [1.0, 0.5555555555555556, 0.8888888888888888, 0.8888888888888888, 0.5555555555555556, 0.6666666666666666, 0.2222222222222222, 0.0]
кількість проектів [1.0, 0.5555555555555556, 0.8333333333333334, 0.8333333333333334, 0.5555555555555556, 0.5555555555555556, 0.2777777777777778, 0.0]
авторизація користувача через соцмережі [1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0]
можливість переключення мови [1.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0]
інтеграція гугланалітики [1.0, 1.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0]
тривалість розробки [-0.0, 0.3333333333333333, 1.0, 0.6666666666666666, 0.6666666666666666, 0.3333333333333333, 0.6666666666666666, 1.0]
вартість [-0.0, 0.18181818181818182, 0.09090909090909091, 0.36363636363636365, 0.45454545454545453, 0.5454545454545454, 0.7272727272727273, 1.0]
input weight for criteria  кількість сторінок
2
input weight for criteria  кількість зареєстрованих користувачів
2
input weight for criteria  кількість проектів
4
input weight for criteria  авторизація користувача через соцмережі
1
input weight for criteria  можливість переключення мови
1
input weight for criteria  інтеграція гугланалітики
2
input weight for criteria  тривалість розробки
1
input weight for criteria  вартість
8
criteria weight:
кількість сторінок 2.0
кількість зареєстрованих користувачів 2.0
кількість проектів 4.0
```

Рисунок 4.3, 4.4, 4.5, 4.6, 4.7, – Введення даних

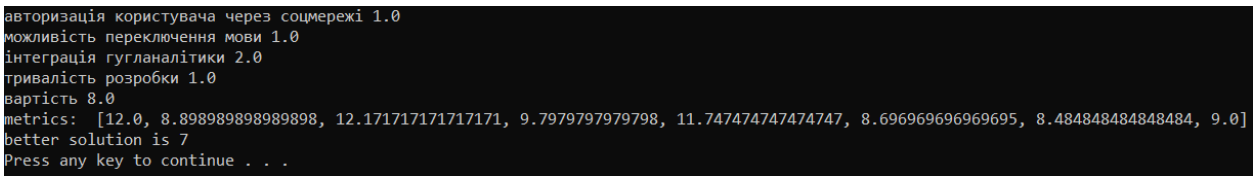


Рисунок 4.8 – Результат роботи програми

Отже, найкращий варіант – 7. Результати розрахунків засобами MsExcel та програмна реалізація у Python однакові.

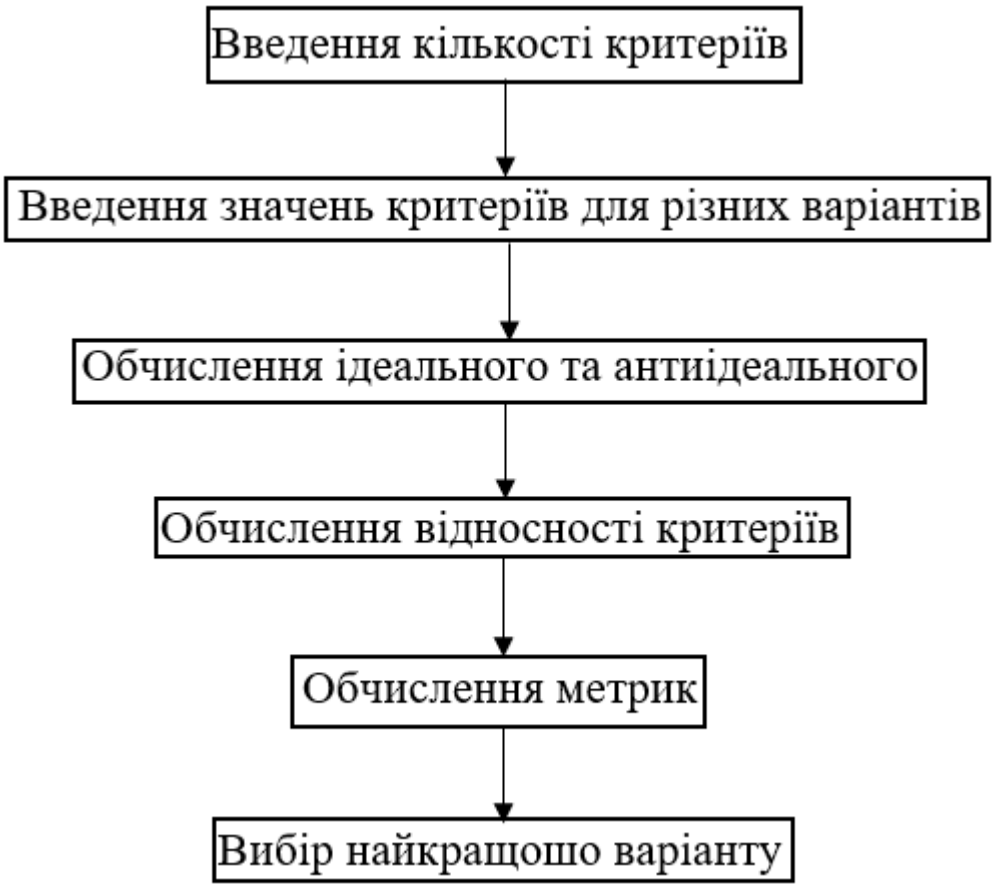


Рисунок 4.9 – Блок-схема алгоритму

4.4 Альтернативне рішення методом лінійного програмування

Лінійне програмування – розділ математичного програмування, який вивчає методи знаходження умовного екстремуму функції, на змінні якої накладені лінійні обмеження.

Особливо цікаві завдання пошуку оптимальних рішень у різнихсферах людської діяльності.

Для вирішення цих завдань насамперед необхідно створити математичну модель – формальний опис цього явища.

Будь-яка модель передбачає ідеалізацію та абстракцію: треба певною мірою спростити постановку завдання та відволіктися від її особливостей.

Велика кількість економічних, технічних та інших процесів досить добре і повно описується лінійними моделями.[9]

Усі завдання лінійного програмування поділяються на 2 види:

1) Завдання, у яких потрібно мінімізувати цільову функцію;
2) завдання, у яких потрібно максимізувати цільову функцію.

Лінійне програмування (та дослідження задачі лінійного програмування) є однією із найрозвинутиших галузей математичного програмування та теорії оптимізації.

Загальна постановка задачі лінійного програмування, та один із підходів до її розв'язання (ідея розрішаючих множників або двоїстих оцінок) вперше наведено в роботі математика Канторовича Л.В. в 1939 році.

Зада́ча лінійного програмува́ння — це задача оптимізації з лінійною цільовою функцією та допустимою множиною, обмеженою лінійними рівностями або нерівностями.

Приклади стандартних задач лінійного програмування:

- Транспортна задача;
- задача комівояжера;
- задача про призначення;
- задача оптимального виробничого планування;
- задача про суміші або задача про дієту;
- задача про оптимальний план перевезень;
- задача про оптимальне розміщення виробництва;
- задача про раціональний розкрій матеріалів;
- задача динаміки виробництва і створення запасів;
- стохастична задача комплектування станочного парку;
- задача про управління портфелем активів;
- задача про використання потужностей (завантаження обладнання);
- задача про ранець;
- задача про розподіл ресурсів.

Предметом вивчення цієї теми є лінійні екстремальні завдання, які мають цільову функцію F– лінійною.[8]

Існують різні форми запису задачі ЛП:

- векторна форма;
- матрична форма запису.[16]

Основне завдання лінійного програмування формулюється так:
Задано лінійну функцію Z:

$$z = c_1 x_1 + c_2 x_2 + \ldots + c_n x_n \to \max\ (\min) \qquad (1)$$

та система лінійних нерівностей :

(2)

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots \quad + a_{1n}x_n \le b_1, \\ a_{21}x_1 + a_{22}x_2 + \ldots \quad + a_{2n}x_n \le b_2, \\ \qquad\qquad \ldots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots \quad + a_{mn}x_n \le b_m, \\ x_j \ge 0, \quad j = 1,2,\ldots,n \end{cases}$$

Потрібно знайти такі значення, (x_1, x_2,…, x_n) при яких функція Z досягає свого екстремуму (max або min) і при цьому виконуються умови (2).

Сформулюємо цільову функцію для нашої задачі. Це буде максимальна кількість проектів, інформація про які буде зберігатися у веб-додатку, максимальна кількість зареєстрованих користувачів, та кількість сторінок сайту. Вагові коефіцієнти візьмемо із Таблиця 1.6.

Обмеження буде містити вартість проекту, яка б не перевищувала 4 тис., кількість сторінок не може бути менша однієї, а кількість користувачів не менша 50, кількість проектів не менша 20.

Розрахунок зробимо за допомогою MsExcel, використовуючи його вбудовані засоби аналізу, а саме “Пошук рішення”, який визначає оптимальне значення цільової комірки, змінюючи значення заданих параметрів.

Таблиця 1.9 – Рішення із застосуванням методів лінійного програмування

к-сть сторінок	к-сть зареєстрованих користувачів	к-сть проектів	обмеження	цільова функція
1	178,0487826	60,97560875	4000,000001	602,0000001

Обмеження:

f_x =A30*1000+B30*10+C30*20

Цільова функція:

f_x =2*A30+2*B30+4*C30

Задання параметрів пошуку рішення:

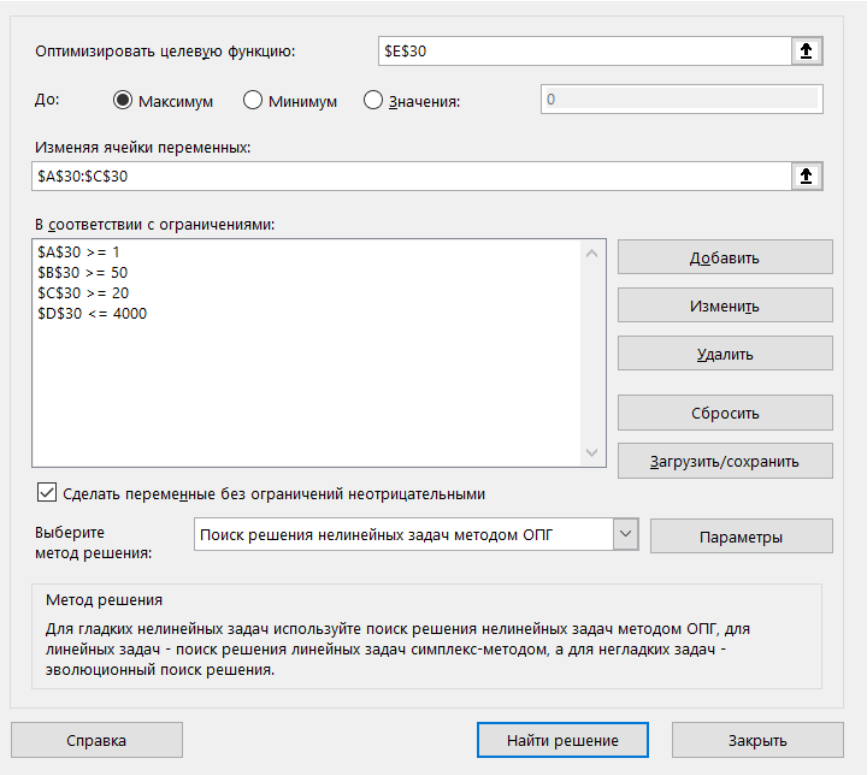


Рисунок 4.10 – Параметри пошуку рішення

Висновки за розділом

Порівнявши методи суміщеного ідеалу і метод лінійного програмування можна зазначити, що метод суміщеного ідеалу більше підходить для оціночного вибору із множини варіантів із заздалегідь заданими параметрами, а метод лінійного програмування дозволяє уточнювати значення параметрів, якщо можна їх змінювати для досягнення оптимального значення цільової функції. Лістинг програми для реалізації методу суміщеного ідеалу наведено в додатку.

ВИСНОВКИ

У кваліфікаційній роботі було розглянуто методи та засоби розробки web-додатків на прикладі системи підтримки соціальних проектів.

Виконано порівняння пропозицій сучасних постачальників хмарних сервісів та обрано Amazon Web Services (AWS), який має найбільше можливостей та займає 30% ринку хмарних послуг. Виконано порівняння найпоширеніших безкоштовних СУБД та обрано PostgreSQL, як реляційну базу даних, що відповідає сучасним стандартам та легко інтегрується з іншими засобами розробки.

Для адміністрування та розробки бази даних використано платформу з відкритим вихідним кодом PgAdmin. Розроблено схему та структуру бази даних «Social_projects», яка містить інформацію про користувачів, проекти, їх характеристики та результати підтримки проектів. Розроблено запити для створення таблиць, налаштовано зв'язки між таблицями, також наведено приклади запитів для введеня даних та пошукових запитів. Підключено створену базу даних до середовища розробки Visual Studio.

Для розробки web-додатка обрано технологію ASP.NET MVC (Model-View-Controller), розглянуто її основні принципи, інструменти, переваги та недоліки. В процесі роботи над додатком створено один контролер, п'ять моделей та вісім представлень. Розглянуто основні складові AWS для розгортання додатку – EC2, S3 та RDS для PostgreSQL. Наведено характеристики відповідних інстансів та команди підключення та налаштування обраних сервісів.

Проведено дослідження оптимальності вибору засобів розробки, виконано порівняння методів суміщеного ідеалу (розроблено відповідну програму та протестовано її роботу) і лінійного програмування (виконано розрахунки за допомогою вбудованих засобів Microsoft Excel).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Введення в структуру MVC. URL: https://www.geeksforgeeks.org/mvc-framework-introduction/
2. Васильченко І.П. та ін. Вища математика: основні означення, приклади і задачі. Навч. посібник: У двох книгах. Книга 2/ І.П. Васильченко, В.Я. Данилов, А.І. Лобанов, Є.Ю. Таран. – друге видання зі змінами. – К.: Либідь, 1994. – 280 с.
3. Типи інстансів. URL: https://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/instance-types.html
4. Модель узгодженості даних Amazon S3. URL: https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html#ConsistencyModel
5. Вовк Р.Б. Організація баз даних: практикум. - Івано-Франківськ: ІФНТУНГ, 2016. - 102 с.
6. Пасічник В. В., Резніченко В. А. Організація баз даних та знань. – К.: Видавнича група BHV, 2006. — 384 с.
7. Технології програмування та створення програмних продуктів: конспект лекцій / О. В. Алексенко. – Суми : Сумський ДУ, 2013. – 133 с.
8. Hands-on network programming with C# and .NET Core: build robust network applications with C# and .NET Core / Sean Burns. - Birmingham, UK : Packt Publishing, 2019.
9. Організація баз даних: практичний курс: Навч. посіб. для студ. / А. Ю. Берко, О. М. Верес; Нац. ун-т «Львів. політехніка». — Л., 2003. — 149 с.
10. Дудзяний І.М. Об’єктно-орієнтоване моделювання програмних систем: [навч. посіб.] / І.М. Дудзяний. – Львів, Видавничий центр ЛНУ ім. Івана Франка, 2007. – 108 с.
11. Грицунов О. В. Інформаційні системи та технології. Навчальний посібник. — Х.: ХНАМГ, 2010. — 222 с.
12. ASP.NET MVC 4 in Action / [Palermo Jeffrey, Bogard Jimmy, Hexter Eric and others]; foreword by Phil Haack. – Shelter Island: Manning Pablications Co., 2012. – 406 p.
13. Daniel S. Illustrated C# 2012. C# 5.0 presented clearly, concisely and visually / Solis Daniel. – [4th edition]. – New York: Apress, 2012. – 750 p.
14. Patrick T. Microsoft ADO.NET 4. Step by Step / Tim Patrick. – California: O’Reilly Media, 2012. – 441 p.
15. Підключення до БД. URL: https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_ConnectToPostgreSQLInstance.html

16. Adam Freeman. Pro ASP.NET MVC 5 (Expert's Voice in ASP.Net) 6th ed. Edition – 992 p.
17. Васильєв А.Ю. Робота з PostgreSQL: налаштування та масштабування. - Creative Commons Attribution-Noncommercial 4.0 International, 2017-286 с.
18. Карпова Т.С. Бази даних. Моделі, технологія, реалізація. - М.: НОУ "Інтуїт", 2016. - 403 с.
19. Карвін Б. Програмування баз даних SQL. Типові помилки та їх усунення / Б. Карвін. - М.: Рід Груп, 2012. - 336 с.
20. Сірант О.В., Коваленко Т.А. Робота із базами даних. - М.: НОУ "Інтуїт", 2016. - 150 с.

ДОДАТОК

Лістинг програми для методу суміщеного ідеалу:

```
def research():
    criteria_more = []
    criteria_less = []
    version_more = []
    version_less = []
    ideal_more = []
    ideal_less = []
    antiideal_more = []
    antiideal_less = []
    rel_more = []
    rel_less = []
    weight_more = []
    weight_less = []
    metrics = []
    number_of_criteria_more = input("number of criteria for increment \n")
    number_of_criteria_less = input("number of criteria for decrement \n")
    number_of_version = input("number of version \n")
    for i in range(int(number_of_criteria_more)):
        t = input("input next criteria for increment \n")
        criteria_more.append(t)
    for i in range(int(number_of_criteria_more)):
        version_more.append([])
        print("input criteria ", criteria_more[i])
        for j in range(int(number_of_version)):
            print("for versia ", j + 1)
            v = input()
            version_more[i].append(float(v))
    for i in range(int(number_of_criteria_less)):
        t = input("input next criteria for decrement \n")
        criteria_less.append(t)
    for i in range(int(number_of_criteria_less)):
        version_less.append([])
        print("input criteria ", criteria_less[i])
        for j in range(int(number_of_version)):
            print("for versia ", j + 1)
            v = input()
            version_less[i].append(float(v))

    for i in range(len(criteria_more)):
        ideal_more.append(max(version_more[i]))
        antiideal_more.append(min(version_more[i]))
        print(criteria_more[i], version_more[i], "ideal = ", ideal_more[i],
"antiideal = ", antiideal_more[i])
    for i in range(len(criteria_less)):
        ideal_less.append(min(version_less[i]))
        antiideal_less.append(max(version_less[i]))
        print(criteria_less[i], version_less[i], "ideal = ", ideal_less[i],
"antiideal = ", antiideal_less[i])
```

```
    print("relative criteria values:")
    for i in range(len(criteria_more)):
        rel_more.append([])
        for j in range(int(number_of_version)):
            km = (ideal_more[i] - version_more[i][j]) / (ideal_more[i] -
antiideal_more[i])
            rel_more[i].append(km)
        print(criteria_more[i], rel_more[i])
    for i in range(len(criteria_less)):
        rel_less.append([])
        for j in range(int(number_of_version)):
            kl = (ideal_less[i] - version_less[i][j]) / (ideal_less[i] -
antiideal_less[i])
            rel_less[i].append(kl)
        print(criteria_less[i], rel_less[i])

    for i in range(int(number_of_criteria_more)):
        print("input weight for criteria ", criteria_more[i])
        w = input()
        weight_more.append(float(w))
    for i in range(int(number_of_criteria_less)):
        print("input weight for criteria ", criteria_less[i])
        w = input()
        weight_less.append(float(w))
    print("criteria weight:")
    for i in range(int(number_of_criteria_more)):
        print(criteria_more[i], weight_more[i])
    for i in range(int(number_of_criteria_less)):
        print(criteria_less[i], weight_less[i])

    for j in range(int(number_of_version)):
        m = 0
        for i in range(int(number_of_criteria_more)):
            m = m + rel_more[i][j]*weight_more[i]
        for i in range(int(number_of_criteria_less)):
            m = m + rel_less[i][j]*weight_less[i]
        metrics.append(m)
    print("metrics: ", metrics)
    print("better solution is", metrics.index(min(metrics)) + 1)

    with open(r"D:\research.txt", "a") as file:
        for i in range(len(criteria_more)):
            file.write(criteria_more[i] + "  " + "".join(str(version_more[i])) + "
ideal = " + str(ideal_more[i]) + " antiideal = " + str(antiideal_more[i]) + '\n')
        for i in range(len(criteria_less)):
            file.write(criteria_less[i] + "  " + "".join(str(version_less[i])) + "
ideal = " + str(ideal_less[i]) + " antiideal = " + str(antiideal_less[i]) + '\n')
        file.write("relative criteria values:" + '\n')
        for i in range(len(criteria_more)):
            file.write(criteria_more[i] + "  " + "".join(str(rel_more[i])) + '\n')
        for i in range(len(criteria_less)):
            file.write(criteria_less[i] + "  " + "".join(str(rel_less[i])) + '\n')
        file.write("criteria weight:" + '\n')
        for i in range(int(number_of_criteria_more)):
            file.write(criteria_more[i] + "  " + str(weight_more[i]) + '\n')
        for i in range(int(number_of_criteria_less)):
            file.write(criteria_less[i] + "  " + str(weight_less[i]) + '\n')
        file.write("metrics:" + '\n' + "".join(str(metrics)) + '\n')
        file.write("better solution is version  " + str(metrics.index(min(metrics)) +
1))
    return
research()
```

Результати тестування, які записані у документ research.txt:

```
кількість сторінок  [1.0, 2.0, 4.0, 5.0, 5.0, 8.0, 8.0, 10.0] ideal = 10.0
antiideal = 1.0
кількість зареєстрованих користувачів  [100.0, 500.0, 200.0, 200.0, 500.0,
400.0, 800.0, 1000.0] ideal = 1000.0 antiideal = 100.0
кількість проектів  [20.0, 100.0, 50.0, 50.0, 100.0, 100.0, 150.0, 200.0]
ideal = 200.0 antiideal = 20.0
авторизація користувача через e-mail або соцмережі  [0.0, 1.0, 0.0, 1.0,
1.0, 1.0, 1.0, 1.0] ideal = 1.0 antiideal = 0.0
можливість переключення мови  [0.0, 1.0, 0.0, 1.0, 0.0, 1.0, 1.0, 1.0]
ideal = 1.0 antiideal = 0.0
інтеграція гугланалітики  [0.0, 0.0, 0.0, 1.0, 0.0, 1.0, 1.0, 1.0] ideal =
1.0 antiideal = 0.0
тривалість розробки (год)  [100.0, 200.0, 400.0, 300.0, 300.0, 200.0,
300.0, 400.0] ideal = 100.0 antiideal = 400.0
вартість $  [4000.0, 6000.0, 5000.0, 8000.0, 9000.0, 10000.0, 12000.0,
15000.0] ideal = 4000.0 antiideal = 15000.0
relative criteria values:
кількість сторінок  [1.0, 0.8888888888888888, 0.6666666666666666,
0.5555555555555556, 0.5555555555555556, 0.2222222222222222,
0.2222222222222222, 0.0]
кількість зареєстрованих користувачів  [1.0, 0.5555555555555556,
0.8888888888888888, 0.8888888888888888, 0.5555555555555556,
0.6666666666666666, 0.2222222222222222, 0.0]
кількість проектів  [1.0, 0.5555555555555556, 0.8333333333333334,
0.8333333333333334, 0.5555555555555556, 0.5555555555555556,
0.2777777777777778, 0.0]
авторизація користувача через e-mail або соцмережі  [1.0, 0.0, 1.0, 0.0,
0.0, 0.0, 0.0, 0.0]
можливість переключення мови  [1.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0]
інтеграція гугланалітики  [1.0, 1.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0]
тривалість розробки (год)  [-0.0, 0.3333333333333333, 1.0,
0.6666666666666666, 0.6666666666666666, 0.3333333333333333,
0.6666666666666666, 1.0]
вартість $  [-0.0, 0.18181818181818182, 0.09090909090909091,
0.36363636363636365, 0.45454545454545453, 0.5454545454545454,
0.7272727272727273, 1.0]
criteria weight:
кількість сторінок  2.0
кількість зареєстрованих користувачів  2.0
кількість проектів  4.0
авторизація користувача через e-mail або соцмережі  1.0
можливість переключення мови  1.0
інтеграція гугланалітики  2.0
тривалість розробки (год)  1.0
вартість $  8.0
metrics:
[12.0, 8.898989898989898, 12.171717171717171, 9.7979797979798,
11.747474747474747, 8.696969696969695, 8.484848484848484, 9.0]
better solution is version  7
```