МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»

Кафедра інженерії програмного забезпечення

КУРСОВА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

з дисципліни: «Бази даних»

на тему:

«База даних віртуального планетарію»

студента 2 курсу групи ІПЗ-23-1
спеціальності 121 «Інженерія програмного забезпечення»
Печериці Богдана Дмитровича
(прізвище, ім'я та по-батькові)

Керівник ст. викладач кафедри ІПЗ
Світлана КРАВЧЕНКО

Дата захисту: "____" _____ 2024 р.
Національна шкала _____
Кількість балів: _____
Оцінка: ECTS _____

Члени комісії

_____ Інна СУГОНЯК
(підпис) (прізвище та ініціали)

_____ Олексій ЧИЖМОТРЯ
(підпис) (прізвище та ініціали)

_____ Ольга КОРОТУН
(підпис) (прізвище та ініціали)

_____ Світлана КРАВЧЕНКО
(підпис) (прізвище та ініціали)

Житомир – 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
Факультет інформаційно-комп'ютерних технологій
Кафедра інженерії програмного забезпечення
Освітній рівень: бакалавр
Спеціальність 121 «Інженерія програмного забезпечення»

«ЗАТВЕРДЖУЮ»
Завідувач кафедри ІПЗ
_____ Тетяна ВАКАЛЮК
«____»_____2024 р.

ЗАВДАННЯ
НА КУРСОВУ РОБОТУ СТУДЕНТУ
Печериці Богдану Дмитровичу

1. Тема роботи: «База даних віртуального планетарію»
   керівник роботи: ст. викладач кафедри ІПЗ Світлана КРАВЧЕНКО
2. Строк подання студентом: « 30 » грудня 2024 р.
3. Вихідні дані до роботи: розробити базу даних віртуального планетарію
4. Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці)
   1. Постановка завдання
   2. Аналіз аналогічних розробок
   3. Алгоритми роботи програми
   4. Опис роботи програми
   5. Програмне дослідження

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
   1. Презентація до КР
   2.Посилання_на_репозиторій:
   https://github.com/BogdanPecheritsa/VirtualPlanetarium.git
6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посади консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1- 4	Світлана КРАВЧЕНКО	20.10.2024	20.10.2024

7. Дата видачі завдання " 20 " жовтня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів курсового проєктування	Строк виконання етапів роботи	Примітки
1	Постановка задачі	20.10.2024	Виконано
2	Пошук, огляд та аналіз аналогічних розробок	30.10.2024	Виконано
3	Формулювання технічного завдання	31.10.2024	Виконано
4	Опрацювання літературних джерел	02.11.2024	Виконано
5	Проєктування структури	05.11.2024	Виконано
6	Написання програмного коду	25.12.2024	Виконано
7	Відлагодження	28.12.2024	Виконано
8	Написання пояснювальної записки	29.12.2024	Виконано
9	Захист		

Студент _____ Печериця Богдан
(підпис) (прізвище та ініціали)

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

РЕФЕРАТ

Метою даної курсової роботи є розробка бази даних для програми віртуального планетарію, що дозволяє зберігати та обробляти дані про небесні об'єкти, такі як зірки, планети, галактики, комети та туманності. Віртуальний планетарій є програмою, що надає можливість користувачам вивчати різноманітні астрономічні об'єкти через візуалізацію на екрані комп'ютера. Оскільки дані про космічні об'єкти є великими і постійно оновлюються, необхідно створити ефективну і масштабовану базу даних для зберігання цих даних.

Розділ 1. Аналіз поставленої задачі

У цьому розділі проведено аналіз задачі створення бази даних для віртуального планетарію. Розглянуті основні вимоги до системи, серед яких зберігання даних про небесні об'єкти, можливість їхнього швидкого пошуку та фільтрації, а також інтеграція з іншими компонентами програми. Окремо було проаналізовано існуючі рішення для подібних задач, зокрема програмне забезпечення, яке використовує бази даних для зберігання астрономічних даних, такі як Stellarium та Celestia. Також розглянуто використання сучасних технологій, таких як Entity Framework і SQL Server, для реалізації архітектури бази даних.

Розділ 2. Проєктування бази даних

У другому розділі розглядається процес проєктування бази даних для віртуального планетарію. Спочатку була створена структура бази даних, яка включає таблиці для зберігання інформації про зірки, планети, галактики, комети та туманності. Для кожної категорії небесних об'єктів були розроблені сутності та їхні зв'язки. Окремо було описано процес моделювання даних із використанням ER-діаграм. Наведено приклади SQL-запитів для виконання операцій з даними, таких як вставка нових об'єктів, оновлення інформації та пошук небесних тіл за різними критеріями. Структура бази даних забезпечує гнучкість і масштабованість для обробки великих обсягів астрономічних даних.

Розділ 3. Опис роботи програмного продукту

У третьому розділі детально описано роботу програмного продукту, який реалізує віртуальний планетарій з інтегрованою базою даних. Описано основні функції програми, зокрема пошук і фільтрація небесних об'єктів, відображення даних на екрані, а також взаємодія користувача з інтерфейсом. Окремо розглянуто роботу API, яке забезпечує обробку запитів до бази даних, а також

інтеграцію з фреймворком .NET. Було проведено тестування основних функцій, зокрема перевірку швидкості виконання запитів до бази даних і взаємодії між компонентами програми.

Розділ 4. Адміністрування бази даних

У цьому розділі розглядаються питання адміністрування бази даних для віртуального планетарію. Описано заходи з резервного копіювання та відновлення даних, а також методи оптимізації продуктивності, такі як індексація таблиць і використання кешування. Розглянуто також питання безпеки даних: захист від несанкціонованого доступу до бази даних, управління правами користувачів, а також методи запобігання некоректним діям користувачів через обмеження прав на виконання операцій з даними.

Висновок

У висновку підсумовано виконану роботу, а також проведено аналіз отриманих навичок і знань у процесі створення бази даних для віртуального планетарію. Розроблена база даних забезпечує ефективне зберігання та обробку даних про небесні об'єкти, а також зручний інтерфейс для взаємодії користувача з програмою. Програмний продукт може бути корисним для астрономів, науковців та ентузіастів, що бажають вивчати космічні об'єкти.

Список використаних джерел

1. "Entity Framework Core Documentation", Microsoft, доступно за посиланням: https://learn.microsoft.com/en-us/ef/core/.
2. "Stellarium", доступно за посиланням: https://stellarium.org/.
3. "Celestia", доступно за посиланням: https://celestia.space/.
4. "SQL Server Documentation", Microsoft, доступно за посиланням: https://learn.microsoft.com/en-us/sql/sql-server/.
5. "Астрономія. Курс лекцій", Автор: Іванов І.І., видавництво "Наука", 2018.

Додатки

1. ER-діаграми бази даних.
2. Схеми таблиць бази даних.
3. Приклади SQL-запитів для виконання операцій з базою даних.
4. Опис API для взаємодії з базою даних.
5. Коди програмних компонентів для роботи з базою даних.

ЗМІСТ

ВСТУП

Актуальність теми
У сучасному світі стрімкий розвиток інформаційних технологій охоплює все більше сфер, включаючи астрономію та освіту. Віртуальні планетарії стають популярним інструментом для вивчення небесних об'єктів, як серед професійних астрономів, так і серед аматорів. Актуальність даної теми полягає в розробці ефективної бази даних для віртуального планетарію, яка дозволяє систематизувати та зберігати великі обсяги інформації про космічні об'єкти, забезпечуючи зручний доступ до цих даних для користувачів. Створення такої бази сприяє популяризації науки, підвищенню рівня астрономічної освіти та розвитку цифрових інструментів у галузі астрономії.

Мета і завдання дослідження
Основною метою цієї курсової роботи є створення бази даних для віртуального планетарію, яка забезпечить ефективне зберігання, обробку та відображення даних про небесні об'єкти.

Для досягнення мети визначено такі завдання:

- аналіз вимог до функціональності бази даних і потреб користувачів;
- проєктування структури бази даних, включаючи моделювання таблиць і зв’язків;
- розробка алгоритмів для виконання запитів і маніпуляцій із даними;
- створення API для взаємодії з базою даних і інтеграції з інтерфейсом користувача;
- тестування та оптимізація бази даних для забезпечення її продуктивності та надійності.

Об’єкт дослідження — методи і технології розробки баз даних для зберігання й обробки інформації про небесні об'єкти.
Предмет дослідження — процеси проєктування, реалізації та адміністрування бази даних для забезпечення її функціональності та інтеграції з віртуальним планетарієм.

РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ

1.1 Аналіз задачі, засобів та методів її вирішення

Метою курсової роботи є розробка бази даних для віртуального планетарію, яка дозволить зберігати й обробляти інформацію про небесні об'єкти, такі як зірки,

планети, галактики, комети та туманності. Для реалізації проєкту були обрані такі технології:

- **C# і платформа .NET** для розробки програмного забезпечення;
- **Entity Framework Core** для об'єктно-реляційного відображення (ORM);
- **SQL Server** як основна СКБД для зберігання даних;
- **WPF** для створення графічного інтерфейсу користувача.

Використання ORM Entity Framework Core дозволяє зручно працювати з базою даних через об’єктно-орієнтовану модель, зменшуючи кількість SQL-запитів і підвищуючи продуктивність розробки. Для реалізації API було використано .NET, що забезпечує обробку запитів і взаємодію з інтерфейсом користувача.

Особливу увагу приділено питанням безпеки даних, оптимізації продуктивності та резервному копіюванню. Реалізація передбачає масштабованість для розширення структури бази даних у майбутньому.

1.2 Аналіз існуючого програмного забезпечення за тематикою курсової роботи

Для аналізу аналогів було обрано програму *Stellarium* (https://stellarium.org/), популярний інструмент для віртуальної візуалізації космічних об'єктів.

Основні функції Stellarium:

- Реалістичне відображення зоряного неба в режимі реального часу.
- Пошук і фільтрація об'єктів за різними критеріями, включаючи назву, тип і координати.
- База даних із великою кількістю небесних об'єктів, зокрема зірок, планет, комет і супутників.
- Інтуїтивно зрозумілий інтерфейс із можливістю налаштування вигляду.

Stellarium використовує заздалегідь створені каталоги небесних об'єктів, що забезпечує швидкий доступ до інформації. Проте його функціонал не завжди дозволяє інтегрувати нові дані чи модифікувати існуючі. У рамках цієї курсової роботи розроблена база даних для віртуального планетарію буде забезпечувати гнучкість у додаванні нових об'єктів і моделюванні взаємозв'язків між ними.

Таким чином, аналіз існуючого програмного забезпечення допоміг визначити ключові вимоги до бази даних і виявити напрями для вдосконалення функціональності.

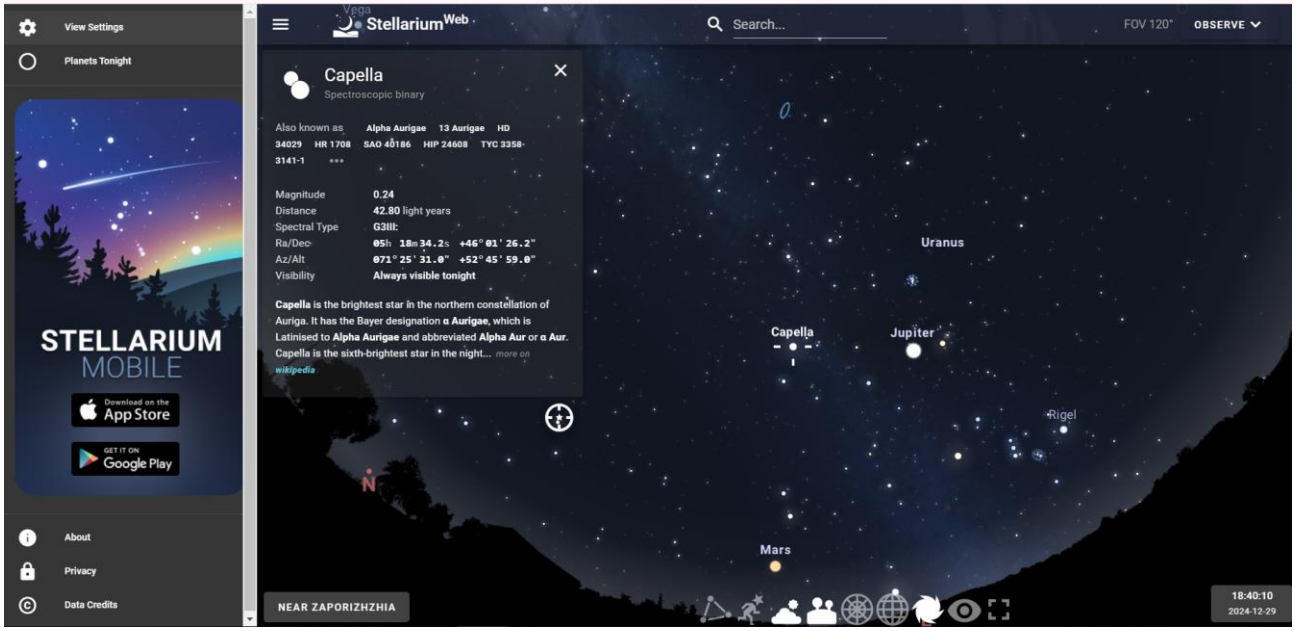

Рис. 1.1. Вигляд сторінки веб-додатку

Другим для порівняння було обрано програму *Celestia* (https://celestiaproject.space/), безкоштовний тривимірний космічний симулятор, що дозволяє досліджувати Всесвіт у реальному часі.

На відміну від Stellarium, який зосереджений на віртуальному відображенні неба із Землі, *Celestia* дозволяє подорожувати космосом, досліджуючи об'єкти на будь-якій відстані від нашої планети. Інструмент забезпечує багату базу даних зірок, планет, супутників та інших космічних тіл.

Основні функції Celestia:

- Тривимірна візуалізація космосу з можливістю подорожувати між об'єктами.
- Підтримка каталогів із сотнями тисяч зірок і планет.
- Можливість розширення бази даних через додавання користувацьких каталогів.
- Відображення орбіт планет і траєкторій космічних апаратів.

Celestia забезпечує інтерактивність і масштабованість, що дозволяє вивчати космос у деталях. Однак його база даних в основному обмежується читанням готових каталогів, і немає зручного інтерфейсу для внесення нових даних або їх модифікації.

Розроблена в рамках курсової роботи база даних вирішує цю проблему, забезпечуючи простоту в додаванні нових об'єктів, моделюванні їхніх характеристик і забезпеченні зручного доступу для користувачів через API та графічний інтерфейс.

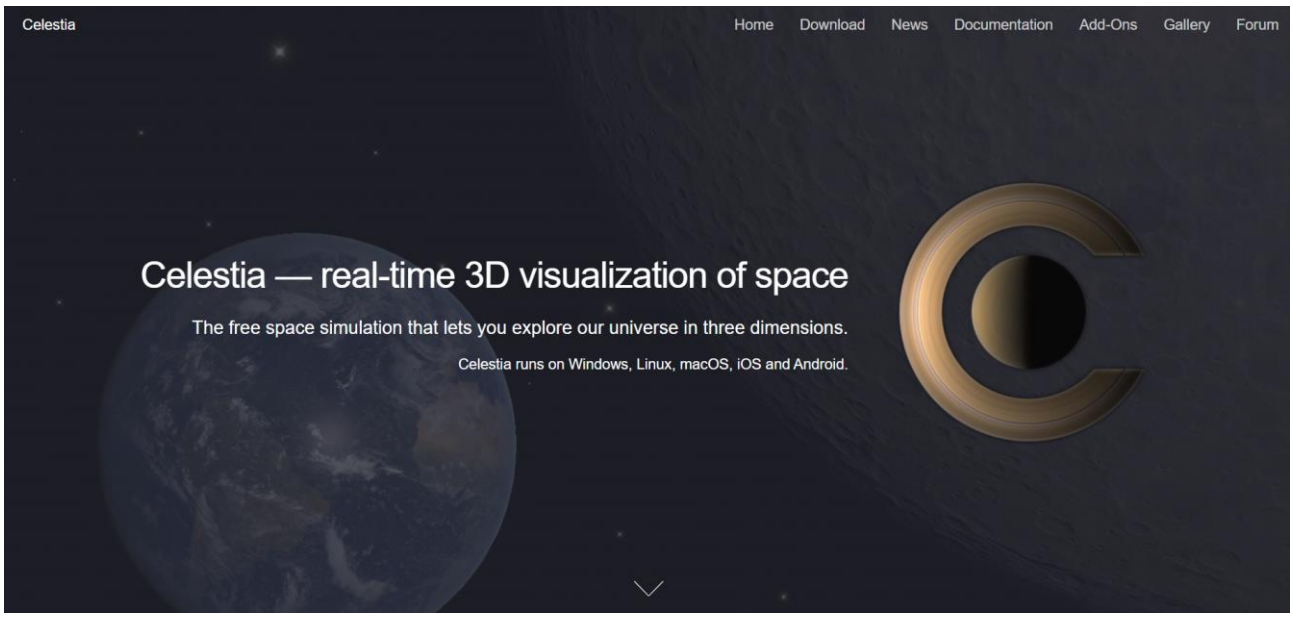

Рис. 1.2. Вигляд сторінки Home

Отже, після порівняння різних реалізацій програм для віртуального планетарію на трьох обраних сервісах, я прийшов до висновку, що найбільш перспективним і ефективним напрямком розвитку мого програмного забезпечення є впровадження функцій, схожих до реалізації, яка існує у третьому аналізованому сервісі *Celestia*. Це обумовлено тим, що саме в цьому сервісі я виявив найбільш гнучкі можливості для візуалізації об'єктів, інтерактивності та масштабування бази даних.

Дизайн інтерфейсу цього сервісу, з акцентом на простоті управління та доступності функцій, також став важливим фактором для забезпечення зручності користувачів. Реалізовані можливості динамічного моделювання орбіт, взаємодії між об'єктами та додавання користувацьких каталогів є цінними для створення сучасного інструменту, який буде цікавий як для дослідників, так і для освітніх цілей.

1.3 Обґрунтування вибору засобів реалізації

У наш час існує велика кількість різних систем управління базами даних (СУБД), і для вибору найкращого варіанту необхідно провести порівняння характеристик кожної з обраних СУБД. Важливо також враховувати їх переваги, недоліки та особливості для конкретних завдань.

У рамках даної курсової роботи я зупинив свій вибір на використанні платформи **Microsoft SQL Server** через її стабільність, високу продуктивність та інтеграцію з екосистемою .NET. Серед основних причин вибору:

- Підтримка складних запитів і функцій, які необхідні для роботи з великим обсягом даних про об'єкти космосу.

- Інтеграція з ORM Entity Framework, що спрощує доступ до даних у додатку WPF.
- Широкі можливості для забезпечення безпеки, резервного копіювання та відновлення даних.
- Підтримка розширень для роботи з геометричними та просторовими даними, що актуально для візуалізації орбіт та розташування космічних об'єктів.

Для розробки інтерфейсу користувача було обрано **WPF (Windows Presentation Foundation)**, який дозволяє створювати зручний графічний інтерфейс із використанням сучасних технологій анімації та відображення даних.

Для роботи з базою даних використано ORM **Entity Framework Core**, що забезпечує зручну взаємодію між програмним кодом і базою даних, знижуючи кількість ручного написання SQL-запитів.

Поєднання цих інструментів дозволяє створити сучасний, продуктивний і зручний у використанні додаток для віртуального планетарію.

Таблиця 1.1

Порівняння характеристик СУБД

Характеристика	MS SQL Server	PostgreSQL	SQLite
Модель даних	Реляційна	Реляційна	Реляційна
Адміністративне керування	Відмінно	Добре	Задовільно
Графічні інструменти	Відмінно	Добре	Відсутні
Простота обслуговування	Відмінно	Добре	Відмінно
Робота з декількома ЦП	Відмінно	Відмінно	Задовільно
Одночасний доступ декількох користувачів	Відмінно	Відмінно	Задовільно
Підключення до Web	Відмінно	Добре	Добре
Побудова БД	Відмінно	Відмінно	Задовільно
Мова SQL	Відмінно	Відмінно	Відмінно
Вбудована мова програмування	Відмінно	Відмінно	Обмежено
Підтримка об'єктно-орієнтованих парадигм	Добре	Відмінно	Обмежено
Інтеграція з іншими СУБД	Відмінно	Добре	Задовільно
Масштаб застосування	Великі підприємства	Підприємства будь-якого розміру	Невеликі проєкти

Таким чином, після проведеного порівняння було визначено, що для поточної курсової роботи найкраще підійде MS SQL Server. Це обумовлено її масштабованістю, високою продуктивністю при роботі з великими обсягами даних, потужними інструментами для адміністрування, розвиненим графічним інтерфейсом, а також можливістю ефективно працювати з API та підтримкою складних запитів на базі SQL. MS SQL Server забезпечує стабільність, безпеку та високу продуктивність, що робить її ідеальним вибором для проєкту, спрямованого на побудову бази даних віртуального планетарію.

Висновки до першого розділу

У пункті 1.1 було розглянуто можливий сценарій реалізації програмного забезпечення для віртуального планетарію, охоплюючи вибір мови програмування C# та використання платформи .NET для створення додатку WPF. Для збереження даних було обрано СУБД MS SQL Server, а для інтегрованого середовища розробки — Microsoft Visual Studio. Детально розглянуто методи та підходи, які можуть бути використані для вирішення поставлених завдань.

У пункті 1.2 проведено аналіз існуючих систем для візуалізації та моделювання космічних об’єктів. Виділено ключові переваги та недоліки кожної реалізації, що дало змогу зробити висновки щодо найбільш актуального та ефективного напрямку розвитку власного програмного забезпечення для створення віртуального планетарію.

У пункті 1.3 було проведено аналіз трьох СУБД, визначено їх основні переваги та недоліки. Таким чином, мною було обрано СУБД MS SQL Server через її масштабованість, високу продуктивність, потужні адміністративні інструменти, а також стабільну інтеграцію з іншими сервісами для побудови надійного та ефективного програмного забезпечення.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Проєктування загального алгоритму роботи програми

На даному етапі необхідно створити загальний алгоритм роботи програмного додатку для віртуального планетарію. Для повноцінного та ефективного використання розробленої бази даних.

Користувач може:

- Переглядати список об’єктів у базі даних (планети, зорі, галактики, комети тощо);

- Фільтрувати об’єкти за назвою;
- Отримувати детальну інформацію про вибраний об’єкт;
- Додавати, редагувати або видаляти свої записи (за наявності відповідних прав доступу).

Адміністратор може:

- Керувати базою даних: додавати, редагувати та видаляти записи будь-якого типу;
- Створювати та змінювати категорії та параметри космічних об’єктів;
- Генерувати звіти про наповненість бази даних та активність користувачів.

Такий розподіл ролей дозволяє забезпечити гнучке використання додатку відповідно до завдань і рівня доступу кожного користувача.

Отже, перш ніж детально розібрати проєктування бази даних, варто зазначити, що створення блок-схеми, яка описує загальний принцип роботи програмного додатку, наразі не є необхідним. Це пов’язано з тим, що мій проєкт спрямований виключно на створення та тестування бази даних для віртуального планетарію, а не на розробку повноцінного програмного забезпечення. Основна мета полягає у перевірці функціональності, оптимізації структури та забезпеченні коректної взаємодії між елементами бази даних.

2.2 Розробка функціональних алгоритмів роботи програми

Фундаментальною складовою цієї курсової роботи є база даних, тому необхідно навести таблиці та схеми, які демонструють її структуру та будову. Логічно розпочати з діаграми бази даних (рис. 2.1).

Структура бази даних:

- Таблиці для збереження даних про типи об’єктів (планети, зорі, галактики тощо);
- Таблиці для характеристик об’єктів, таких як маса, розмір, розташування тощо;
- Таблиці для збереження користувацьких записів і прав доступу.

Наступним кроком є розробка функцій додатку для роботи з базою даних, таких як пошук, фільтрація, додавання та видалення записів, що будуть реалізовані через ORM Entity Framework Core.

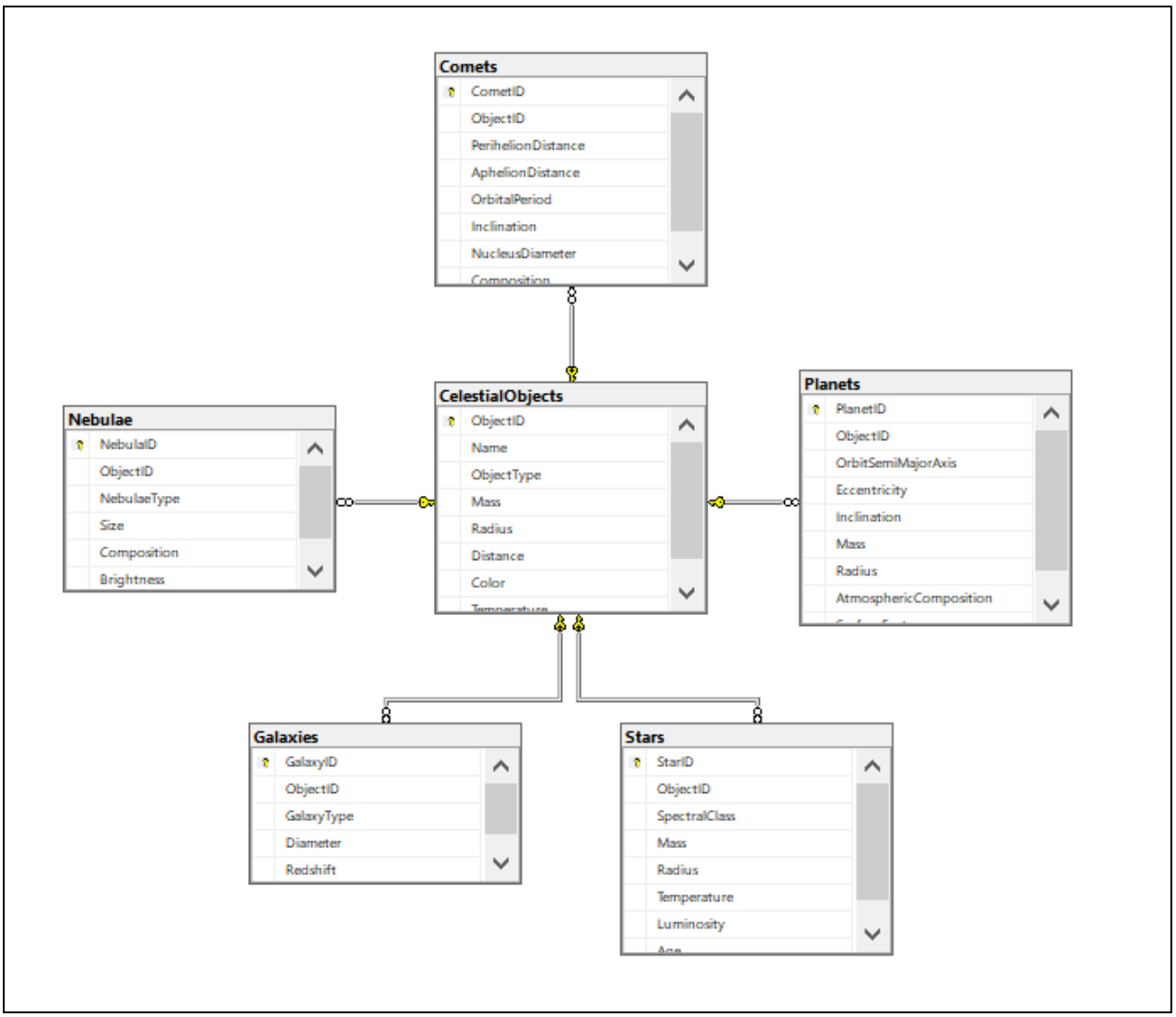

Рис. 2.1. Діаграма бази даних

На діаграмі вище зображені зв’язки всіх створених таблиць. Ключовою таблицею у моєму проєкті є таблиця «CelestialObjects», яка містить інформацію про усі об’єкти. До неї через зв’язки приєднані інші таблиці, такі як «Planets», «Comets», «Stars», «Galaxies», «Nebulaes», які зберігають дані про планети, комети, зірки, галактики та туманності відповідно. Ця структура забезпечує ефективну організацію та обробку даних.

Структура таблиць БД:

Таблиця 2.1 «CelestialObjects»

Поле	Ключ	Тип	Призначення
ObjectId	PK	Guid	Унікальний ідентифікатор об'єкта
Name		varchar(255)	Назва об'єкта
ObjectType		varchar(255)	Тип об'єкта
Mass		float	Маса об'єкта
Radius		float	Радіус об'єкта
Distance		float	Відстань об'єкта
Color		varchar(255)	Колір об'єкта
Temperature		float	Температура об'єкта

Таблиця 2.2 «Comets»

Поле	Ключ	Тип	Призначення
CometId	PK	Guid	Унікальний ідентифікатор комети
CometName		varchar(255)	Назва комети
PerihelionDistance		float	Перигелійна відстань
AphelionDistance		float	Афелійна відстань
OrbitalPeriod		float	Орбітальний період
Inclination		float	Нахил орбіти
NucleusDiameter		float	Діаметр ядра
Composition		varchar(255)	Склад ядра
ObjectId	FK	Guid	Зв'язок із таблицею CelestialObjects

Таблиця 2.3 «Galaxies»

Поле	Ключ	Тип	Призначення
GalaxyId	PK	Guid	Унікальний ідентифікатор галактики
GalaxyName		varchar(255)	Назва галактики
GalaxyType		varchar(255)	Тип галактики
Diameter		float	Діаметр галактики
Redshift		float	Червоний зсув
ObjectId	FK	Guid	Зв'язок із таблицею CelestialObjects

Таблиця 2.4 «Nebulaes»

Поле	Ключ	Тип	Призначення
NebulaeId	PK	Guid	Унікальний ідентифікатор туманності
NebulaeName		varchar(255)	Назва туманності
NebulaeType		varchar(255)	Тип туманності
Size		float	Розмір туманності
Composition		varchar(255)	Хімічний склад
Brightness		float	Яскравість
ObjectId	FK	Guid	Зв'язок із таблицею CelestialObjects

Таблиця 2.5 «Planets»

Поле	Ключ	Тип	Призначення
PlanetId	PK	Guid	Унікальний ідентифікатор планети
PlanetName		varchar(255)	Назва планети
OrbitSemiMajorAxis		float	Вісь орбіти
Eccentricity		float	Ексцентриситет
Inclination		float	Нахил орбіти
Mass		float	Маса планети
Radius		float	Радіус планети
AtmosphericComposition		varchar(255)	Атмосферний склад
SurfaceFeatures		varchar(255)	Особливості поверхні
ObjectId	FK	Guid	Зв'язок із таблицею CelestialObjects

Таблиця 2.6 «Teams»

Поле	Ключ	Тип	Призначення
StarId	PK	Guid	Унікальний ідентифікатор зірки
StarName		varchar(255)	Назва зірки
SpectralClass		varchar(255)	Спектральний клас
Mass		float	Маса зірки
Radius		float	Радіус зірки
Temperature		int	Температура зірки
Luminosity		float	Світність
Age		float	Вік
ObjectId	FK	Guid	Зв'язок із таблицею CelestialObjects

Висновки до другого розділу:

У пункті 2.1 було описано проєктування загального алгоритму роботи віртуального планетарію. Створено блок-схеми, які демонструють, як система повинна функціонувати, зокрема, обробку даних про небесні об’єкти, їх властивості та взаємозв’язки. Описано взаємодію користувача з інтерфейсом системи для доступу до інформації про об'єкти, додавання нових записів і редагування існуючих.

У пункті 2.2 було спроєктовано та створено базу даних для віртуального планетарію, яка включає таблиці для збереження даних про небесні об’єкти різних типів, таких як зірки, планети, галактики, комети та туманності. Також реалізовано зв’язки між цими таблицями, що забезпечує цілісність і зручність роботи з даними. Було створено необхідні механізми для забезпечення коректності обробки даних та підтримки їх актуальності.

Розділ 3. Опис роботи з програмним додатком та його тестування

3.1 Опис роботи з програмним додатком

Після входу у програму користувач отримує доступ до інтуїтивно зрозумілого інтерфейсу, який дозволяє ефективно управляти даними. Інтерфейс поділений на декілька вкладок, кожна з яких відповідає за роботу з певною таблицею бази даних. Користувач може перемикатися між вкладками для взаємодії з різними аспектами системи, такими як інформація про зірки, планети, галактики, комети або туманності.

На кожній сторінці представлені такі функціональні можливості:

CRUD-операції:

- **Створення**: Користувач може додати нові записи, наприклад, створити новий запис про небесний об’єкт із вказанням його параметрів, таких як маса, радіус, температура тощо.
- **Читання**: Дані відображаються у вигляді таблиць із можливістю перегляду деталей кожного запису, таких як склад або відстань від Землі.
- **Оновлення**: Можна редагувати існуючі записи через спеціальні форми, наприклад, змінити діаметр галактики або оновити інформацію про спектральний клас зірки.
- **Видалення**: Є можливість видаляти записи, такі як видалення неточних або застарілих даних про об'єкти.

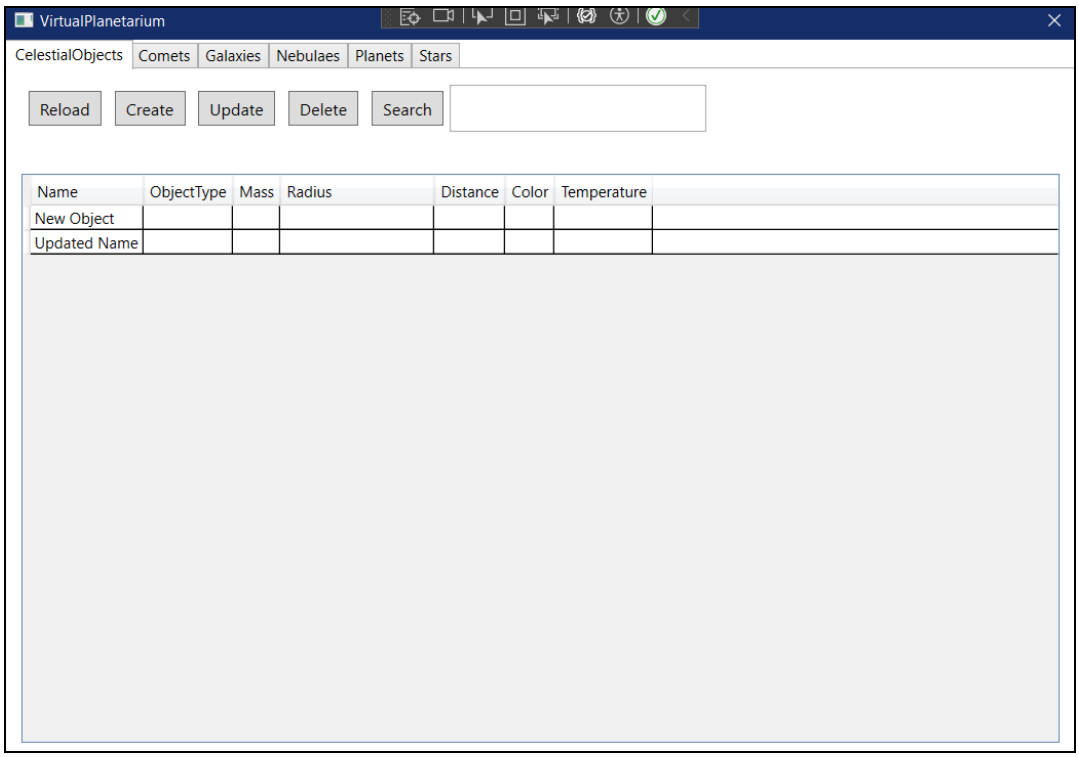

Рис 3.1. Головне меню додатку

3.2 Реалізація операцій обробки даних

В курсовій роботі був використаний патерн проєктування «WPF», що дозволило зручно розділяти різні частини програми (рис. 3.2.).

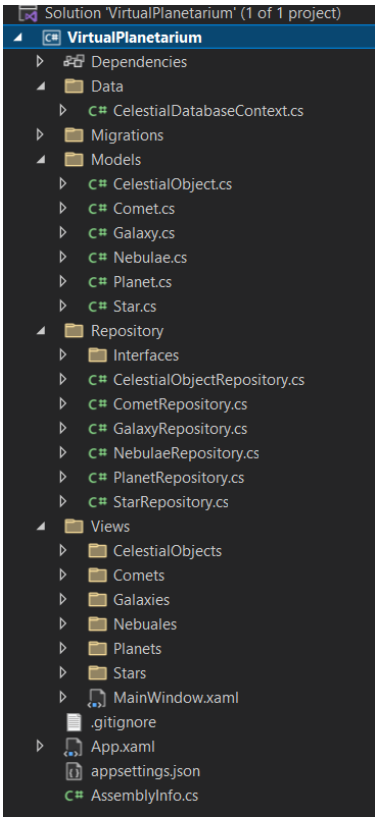

Рис 3.2. Структура проекту

Для роботи з базою даних, потрібно до неї підключитися. Для цього в каталозі «Data» у мене прописані загальні відомості про підключення.

Основна логіка роботи з БД представлена в класі «CelestialDatabaseContext», що знаходиться в каталозі «Data». Там описане підключення до БД та загальний код сгенерований Entity Framework для взяємодії з базою данних.

Лістинг класу «CelestialDatabaseContext»:

```
using System;
using System.Collections.Generic;
using Microsoft.EntityFrameworkCore;
using VirtualPlanetarium.Models;

namespace VirtualPlanetarium.Data;

public partial class CelestialDatabaseContext : DbContext
{
    public CelestialDatabaseContext()
    {
    }

    public CelestialDatabaseContext(DbContextOptions<CelestialDatabaseContext>
options)
        : base(options)
    {
    }

    public virtual DbSet<CelestialObject> CelestialObjects { get; set; }

    public virtual DbSet<Comet> Comets { get; set; }

    public virtual DbSet<Galaxy> Galaxies { get; set; }

    public virtual DbSet<Nebulae> Nebulaes { get; set; }

    public virtual DbSet<Planet> Planets { get; set; }

    public virtual DbSet<Star> Stars { get; set; }

    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
#warning To protect potentially sensitive information in your connection string, you
should move it out of source code. You can avoid scaffolding the connection string by
using the Name= syntax to read it from configuration - see
https://go.microsoft.com/fwlink/?linkid=2131148. For more guidance on storing
connection strings, see https://go.microsoft.com/fwlink/?LinkId=723263.
        => optionsBuilder.UseSqlServer("Data Source=(localdb)\\MSSQLLocalDB;Initial
Catalog=CelestialDatabase;Integrated Security=True;");

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.Entity<CelestialObject>(entity =>
        {
            entity.HasKey(e => e.ObjectId).HasName("PK__Celestia__9A6192B18111D5D6");

            entity.Property(e => e.ObjectId).HasColumnName("ObjectID");
```

```
            entity.Property(e => e.Name)
                .HasMaxLength(100)
                .IsUnicode(false);
            entity.Property(e => e.ObjectType)
                .HasMaxLength(50)
                .IsUnicode(false);
        });

        modelBuilder.Entity<Comet>(entity =>
        {
            entity.HasKey(e => e.CometName).HasName("PK__Comets__6D2DFA460122FA05");

            entity.Property(e => e.CometName).HasColumnName("CometID");
            entity.Property(e => e.Composition).HasColumnType("text");
            entity.Property(e => e.CometId).HasColumnName("ObjectID");

            entity.HasOne(d => d.Object).WithMany(p => p.Comets)
                .HasForeignKey(d => d.CometId)
                .OnDelete(DeleteBehavior.ClientSetNull)
                .HasConstraintName("FK__Comets__ObjectID__412EB0B6");
        });

        modelBuilder.Entity<Galaxy>(entity =>
        {
            entity.HasKey(e =>
e.GalaxyName).HasName("PK__Galaxies__D77289FF4364299B");

            entity.Property(e => e.GalaxyName).HasColumnName("GalaxyID");
            entity.Property(e => e.GalaxyType)
                .HasMaxLength(50)
                .IsUnicode(false);
            entity.Property(e => e.GalaxyId).HasColumnName("ObjectID");

            entity.HasOne(d => d.Object).WithMany(p => p.Galaxies)
                .HasForeignKey(d => d.GalaxyId)
                .OnDelete(DeleteBehavior.ClientSetNull)
                .HasConstraintName("FK__Galaxies__Object__3E52440B");
        });

        modelBuilder.Entity<Nebulae>(entity =>
        {
            entity.HasKey(e =>
e.NebulaeName).HasName("PK__Nebulae__5FEAEDEB32215280");

            entity.ToTable("Nebulae");

            entity.Property(e => e.NebulaeName).HasColumnName("NebulaID");
            entity.Property(e => e.Composition).HasColumnType("text");
            entity.Property(e => e.NebulaeType)
                .HasMaxLength(50)
                .IsUnicode(false);
            entity.Property(e => e.NebulaeId).HasColumnName("ObjectID");

            entity.HasOne(d => d.Object).WithMany(p => p.Nebulaes)
                .HasForeignKey(d => d.NebulaeId)
                .OnDelete(DeleteBehavior.ClientSetNull)
                .HasConstraintName("FK__Nebulae__ObjectI__440B1D61");
        });

        modelBuilder.Entity<Planet>(entity =>
        {
            entity.HasKey(e => e.PlanetName).HasName("PK__Planets__1B0638C51DA90671");

            entity.Property(e => e.PlanetName).HasColumnName("PlanetID");
```

```
            entity.Property(e => e.AtmosphericComposition).HasColumnType("text");
            entity.Property(e => e.PlanetId).HasColumnName("ObjectID");
            entity.Property(e => e.SurfaceFeatures).HasColumnType("text");

            entity.HasOne(d => d.Object).WithMany(p => p.Planets)
                .HasForeignKey(d => d.PlanetId)
                .OnDelete(DeleteBehavior.ClientSetNull)
                .HasConstraintName("FK__Planets__ObjectI__3B75D760");
        });

        modelBuilder.Entity<Star>(entity =>
        {
            entity.HasKey(e => e.StarName).HasName("PK__Stars__06ABC6475484F84A");

            entity.Property(e => e.StarName).HasColumnName("StarID");
            entity.Property(e => e.StarId).HasColumnName("ObjectID");
            entity.Property(e => e.SpectralClass)
                .HasMaxLength(1)
                .IsUnicode(false)
                .IsFixedLength();

            entity.HasOne(d => d.Object).WithMany(p => p.Stars)
                .HasForeignKey(d => d.StarId)
                .OnDelete(DeleteBehavior.ClientSetNull)
                .HasConstraintName("FK__Stars__ObjectID__38996AB5");
        });

        OnModelCreatingPartial(modelBuilder);
    }

    partial void OnModelCreatingPartial(ModelBuilder modelBuilder);
}
```

Таким чином, мені не доведеться вручну прописувати SQL-код, а достатньо буде правильно викликати та передати параметри в метод.

Висновки до третього розділу

У третьому розділі було розглянуто функціонал програмного додатка, реалізованого в рамках курсової роботи, а також підходи до роботи з даними.

Інтерфейс користувача:

Програмний додаток забезпечує інтуїтивно зрозумілий інтерфейс із поділом на вкладки, кожна з яких відповідає за роботу з певним аспектом системи. Завдяки цьому користувач може легко виконувати CRUD-операції: створювати, переглядати, оновлювати та видаляти дані.

Реалізація функціоналу:

Для виконання завдань із керування даними було використано сучасні підходи, зокрема патерн WPF із використанням технології Entity Framework. Це

дозволило автоматизувати взаємодію з базою даних і спростити реалізацію складних операцій.

Архітектура проєкту:

Проєкт побудований із використанням чіткої структури. Каталог «Data» містить усю необхідну інформацію для роботи з базою даних, включно з підключенням через файл налаштувань і класом контексту CelestialDatabaseContext, який забезпечує взаємодію з таблицями бази даних.

Використання Entity Framework:

Ця технологія дозволила відмовитися від написання SQL-запитів вручну, автоматизувавши створення, зчитування, оновлення та видалення записів. Клас контексту CelestialDatabaseContext забезпечує зв’язок між об’єктами додатка та таблицями бази даних.

Зручність розширення:

Завдяки використанню Entity Framework і чіткій структурі проєкту, додаток легко масштабувати й розширювати, додаючи нові таблиці, функціонал або змінюючи існуючу логіку роботи з даними.

Таким чином, у третьому розділі описано, як програмний додаток забезпечує зручність використання для кінцевого користувача та ефективність обробки даних завдяки інтеграції сучасних технологій і підходів до розробки.

РОЗДІЛ 4. АДМІНІСТРУВАННЯ БАЗ ДАНИХ ТА БЕЗПЕКА

4.1 Розробка заходів захисту інформації в БД. У цьому розділі розглянуто адміністрування бази даних та заходи забезпечення безпеки даних. Основною метою є виявлення можливих слабких місць у системі та запобігання їх використанню сторонніми особами чи некоректними запитами.

Виділено три основні ролі користувачів: адміністратор, авторизований користувач та неавторизований користувач.

- **Адміністратор**: користувач, який має повний доступ до управління системою. Головний адміністратор може призначати інших адміністраторів.

- **Авторизований користувач**: має доступ до читання даних та створення власних записів.
- **Неавторизований користувач**: має лише обмежений доступ до перегляду.

Для чіткого розмежування можливостей створено матрицю доступу (таблиця 4.1):

Таблиця 4.1. Матриця доступу ролей

Таблиці	Неавторизований користувач	Авторизований користувач	Адміністратор
Requests	0	1	5
Roles	0	0	5
Teams	0	1	5
Users	0	1	5
UsersRequests	0	2	5
WorkDays	0	3	5
WorkSessions	0	3	5

Розшифрування значень:

- 0 — немає доступу.
- 1 — читання.
- 2 — вставка даних.
- 3 — редагування даних.
- 5 — повний доступ.

Висновки до розділу 4: Для захисту даних створено матрицю доступу, яка регулює права доступу для кожної ролі. Це підвищує безпеку системи, зменшуючи ризики несанкціонованого доступу та помилок. Заходи безпеки враховують принцип мінімізації привілеїв і гарантують ефективний поділ прав доступу.

ВИСНОВКИ

В результаті виконання курсової роботи було створено базу даних віртуального планетарію

1. У першому розділі проаналізовано існуючі рішення для віртуального планетарію, визначено переваги та недоліки, обрано ефективні підходи для розробки системи.
2. У другому розділі спроектовано структуру бази даних, створено діаграму зв’язків та описано взаємодію компонентів.
3. У третьому розділі розглянуто інтерфейс програми, описано основні функції системи та реалізацію операцій з даними.
4. У четвертому розділі розроблено заходи безпеки: створено матрицю доступу, впроваджено механізми захисту даних.

Підсумки:

- Створено базу даних із використанням мов C# та SQL.
- Використано сучасні технології (ASP.NET Core, WPF).
- Реалізовано систему, яка може бути основою для подальшого розвитку (додання React, Chakra UI, GraphQL).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Гілевич В. Бази даних: теорія та практика. – Харків: Фоліо, 2020.
2. MySQL Documentation. – URL: https://dev.mysql.com/doc/.
3. MSDN Library: C# Programming Guide. – URL: https://learn.microsoft.com/en-us/dotnet/csharp/.
4. ASP.NET Core Documentation. – URL: https://learn.microsoft.com/en-us/aspnet/core/.
5. React Official Documentation. – URL: https://reactjs.org/docs/getting-started.html.
6. Chakra UI Documentation. – URL: https://chakra-ui.com/docs.
7. Dapper Documentation. – URL: https://github.com/DapperLib/Dapper.
8. GraphQL Documentation. – URL: https://graphql.org/.
9. Різник М. Основи проектування інформаційних систем. – Київ: Кондор, 2018.
10. Learn SQL – Interactive Tutorial. – URL: https://sqlzoo.net/.
11. Бази даних / Освітній портал ДУ «Житомирська політехніка». – URL: https://learn.ztu.edu.ua/.

ДОДАТКИ

Додаток А. Технічне завдання

1. **Загальні положення**

- **Найменування програмного засобу:** «VirtualPlanetarium».
- **Призначення:** обробка та аналіз даних про робочі сесії співробітників.

2. **Підстава для розробки**
   - Виконується згідно з навчальним планом спеціальності 121 «Інженерія програмного забезпечення».
3. **Вимоги до програми**
   - **Функціональні:** створення, редагування, видалення записів, search
   - **Надійність:** резервне копіювання, захист від збоїв.
   - **Технічні:** оптимальність використання пам’яті, швидкодія.
4. **Документація:** інструкція з встановлення, керівництво користувача, адміністрування бази даних.