НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

«На правах рукопису»
УДК 004.056.5

До захисту допущено:
Завідувач кафедри
_____ Сергій СТІРЕНКО
«___»_____20__ р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою «Інженерія програмного забезпечення комп'ютерних систем»

зі спеціальності 121 «Інженерія програмного забезпечення»

на тему: «Веб-портал про космічні апарати і технології»

Виконав (-ла):
студент (-ка) VI курсу, групи ІМ-12мп
Кіпріянов Гліб Олегович _____

Науковий керівник:
доцент кафедри ОТ, к.т.н., доцент
Селіванов Віктор Левович _____

Консультант з нормоконтролю:
професор кафедри ОТ, д.т.н., професор
Жабін Валерій Іванович _____

Рецензент:
доцент кафедри ІСТ, к.т.н., доцент
Шимкович Володимир Миколайович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2022 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Рівень вищої освіти – другий (магістерський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення комп'ютерних систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Сергій СТІРЕНКО
«___»_____20__ р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Кіпріянов Гліб Олегович

1. Тема дисертації «Веб-портал про космічні апарати і технології», науковий керівник дисертації Селіванов Віктор Левович, доцент кафедри ОТ, к.т.н., доцент, затверджені наказом по університету від «_09_»___11___ 2022 р. №_4115-с_

2. Термін подання студентом дисертації _____

3. Об'єкт дослідження _*процес роботи веб-порталу про космічні технології і апарати, в тому числі пошуку, перекладу на українську мову, зберігання та відтворення інформації*_____

4. Вихідні дані

5. Перелік завдань, які потрібно розробити ___створити веб-портал, що надаватиме інформацію про космічні технології українською мовою, використовуючи розроблені засоби пошуку та перекладу_____

6. Орієнтовний перелік графічного (ілюстративного) матеріалу *48 малюнків, 14 таблиць*

7. Орієнтовний перелік публікацій

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Професор, д.т.н. Жабін В. І.		

9. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Вивчення літератури	18.09.2022	
2	Складання і узгодження технічного завдання	25.09.2022	
3	Написання вступної частини та огляд рішень	25.09.2022	
4	Створення веб-порталу та бази даних	30.10.2022	
5	Оформлення документації ДП	11.12.2022	
6	Попередній захист та проходження нормативного контролю	19.12.2022	
7	Подання ДП рецензенту		

Студент Гліб КІПРІЯНОВ

Науковий керівник Віктор СЕЛІВАНОВ

РЕФЕРАТ

на магістерську дисертацію

виконану на тему: Веб-портад про космічні апарати і технології

студентом: Кіпріяновим Глібом Олеговичем

Загальний об’єм роботи становить 115 сторінок. Проект містить реферат, вступ, чотири розділи, висновки. У проекті наявні 48 рисунків, 14 таблиць, два додатки, 40 джерел у переліку посилань.

У XXI столітті космонавтика і аерокосмічні технології вступають у нову епоху. Нові матеріали, методи виробництва, комп’ютери з великими обчислювальними потужностями значно здешевили вартість доступу у космос. Колись передові у освоєнні космічного простору держави намагаються втримати свої домінуючі позиції. Країни, що раніше не мали власних космічних програм, розглядають можливості використання космосу. Найшвидші темпи розвитку показують приватні компанії, деякі з яких навіть запускають власні пілотовані космічні кораблі і мають плани колонізації Місяця.

В таких умовах пошук актуальної інформації про досягнення космонавтики може бути дуже непростим. Таким чином, створення мережевого ресурсу, що надавав би інформацію про космічні технології і апарати, буде актуальною темою.

Мета проекту – розробка веб-порталу, присвяченого космічним технологіям і апаратам. Веб-портал надаватиме можливість користувачам Інтернету, які цікавляться дослідженням і освоєнням космосу, інформацію українською мовою. Для спрощення роботи адміністраторів веб-портал матиме можливість парсингу даних і перекладу на українську мову.

Об’єкт дослідження – процес роботи веб-порталу про космічні технології і апарати, в тому числі пошуку, перекладу на українську мову, зберігання та відтворення інформації.

Предмет дослідження – методи та засоби забезпечення роботи веб-порталу, пошуку, перекладу, зберігання та інформації.

В ході роботи над проектом був розроблений інформаційний ресурс, що може надавати інформацію про космічні технології і апарати українською мовою. При цьому для наповнення інформацією бази даних було застосовано парсинг і автоматичний переклад. Були досліджені і застосовані сучасні методи розробки програмного забезпечення. Окрім цього, під час роботи був проведений огляд та аналіз існуючих рішень, виділено вимоги до програмного продукту та цілі розробки.

Практичне значення роботи полягає в тому, що готовий проект дасть можливість знайти інформацію про космічні технології, запуски, орбітальні і міжпланетні апарати українською мовою. На сьогоднішній день така інформація часто неактуальна, неповна, інколи суперечлива. Україномовних Інтернет-ресурсів, присвячених космонавтиці, майже не існує.

Ключові слова: веб-портал, ASP.NET Core, Entity Framework Core, C#, бази даних, космічні апарати і технології.

ABSTRACT

The total volume of work is 115 pages. The project contains an abstract, an introduction, four chapters, and conclusions. The project includes 48 figures, 14 tables, two appendices, and 40 sources in the list of references.

In the 21st century, cosmonautics and aerospace technologies are entering a new era. New materials, manufacturing methods, computers with high computing power have significantly reduced the cost of access to space. When advanced states in military space try to maintain their dominant positions. Countries that previously did not have their own space programs are considering the possibilities of using space. The fastest pace of development is demonstrated by private companies, some of which even launch their own manned spacecraft and have plans to colonize the moon.

In such conditions, searching for up-to-date information on the achievements of cosmonautics can be very difficult. Thus, the creation of a network resource that provides information about space technologies and devices will be a relevant topic.

The metaproject is the development of a web portal dedicated to space technologies and devices. The web portal will provide information in Ukrainian to Internet users interested in research and space exploration. To simplify the work of web portal administrators, it will be possible to parse data and translate it into Ukrainian.

The object of the research is the process of the web portal about space technologies and devices, including search, translation into Ukrainian, storage and reproduction of information.

The subject of research is methods and means of ensuring the operation of the web portal, search, translation, storage and information.

In the course of work on the project, an information resource was developed that can provide information about space technologies and devices in Ukrainian. At the same time, syntactic analysis and automatic translation were

used to supplement the database information. Modern methods of software development were researched and applied. on one, during this work, a review and analysis of available solutions was carried out, requirements for the software product and target developments were made.

The practical significance of the work arises from the fact that the finished project has the opportunity to find information about space technologies, launches, orbital and interplanetary vehicles in Ukrainian. To date, such information is often irrelevant, incomplete when it is mentioned. There are almost no Ukrainian-language Internet resources devoted to cosmonautics.

Keywords: web portal, ASP.NET Core, Entity Framework Core, C#, databases, space vehicles and technologies.

Пояснювальна записка

до магістерської дисертації

на тему: «Веб-портад про космічні апарати і технології»

Київ – 2022

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

Веб-сайт — сукупність веб-сторінок, пов'язаних за змістом, опублікованих у мережі Інтернет під єдиним доменним ім'ям, розташованих принаймні на одному веб-сервері.

Мережа Інтернет — глобальна система сполучених комп'ютерних мереж, пов'язана через стек протоколів TCP/IP.

Парсинг, або синтаксичний аналіз — це процес аналізу послідовності символів у природній мові, мовах програмування або структурах даних.

Космічний апарат — технічний пристрій, призначений для роботи у космічному просторі або на поверхні інших небесних тіл. Більшість космічних апаратів не здатні самостійно покинути атмосферу землі.

Ракета-носій — це апарат, призначений для доставки корисного навантаження (супутників) у космічний простір, працюючий на принципах реактивного руху.

Архітектура програмного забезпечення — це фундаментальні структури програми та методи створення таких програм.

Парсинг — це процес аналізу послідовності символів у природній мові, мовах програмування або структурах даних.

ВСТУП

Дослідження космосу — відкриття та освоєння людиною космічного простору і небесних тіл за допомогою космічних технологій і апаратів. Для дослідження космосу використовуються як пілотовані космічні кораблі, так і автоматичні космічні апарати. Завдяки космонавтиці людство може знаходити відповіді на фундаментальні питання про наше місце у Всесвіті і історію сонячної системи. Через вирішення задач, що постають перед космонавтикою, з'являються нові технології і нові способи їх застосування. Людство розширює свій багаж знань про Всесвіт навколо.

Сьогодні інтерес держав до досліджень космосу повертається. НАСА продовжують вивчення Марсу новим марсоходом «Персеверанс» («Наполегливість») зі спеціально розробленим дроном «Інженьюіті» («Винахідливість») на борту. Китай висадив на Марс власний марсохід, і став другою державою (після США), якій це вдалося. На орбіту Марсу вийшла автоматична міжпланетна станція Об'єднаних Арабських Еміратів.

Крім цього, НАСА створило міжнародну програму «Артеміда» з метою повторної висадки людини на Місяць і створення. Запуск першої місії з некерованим обльотом Місяця відбувся 16 листопада 2022 року.

Приватні компанії також почали запускати пілотовані космічні місії, зокрема SpaceX працює над надважким багаторазовим космічним кораблем Starship, а кораблі Crew Dragon регулярно літають до Міжнародної космічної станції.

Україна також намагається розвивати космічну галузь. Подаються пропозиції збудувати космодроми у південних областях, або на узбережжі Канади. Український супутник Січ-2-30, запущений у 2022 році, дозволяє знімати поверхню Землі з роздільною здатністю 7,8 метрів на піксель, і буде інтегрований в європейське сузір'я супутників Copernicus.

В таких умовах пошук актуальної інформації про досягнення космонавтики може бути дуже непростим. Таким чином, створення

мережевого ресурсу, що надавав би інформацію про космічні технології і апарати, буде актуальною темою.

Мета проекту – розробка веб-порталу, присвяченого космічним технологіям і апаратам. Веб-портал надаватиме можливість користувачам Інтернету, які цікавляться дослідженням і освоєнням космосу, інформацію українською мовою. Для спрощення роботи адміністраторів веб-портал матиме можливість парсингу даних і перекладу на українську мову.

Об’єкт дослідження – процес роботи веб-порталу про космічні технології і апарати, в тому числі пошуку, перекладу на українську мову, зберігання та відтворення інформації.

Предмет дослідження – методи та засоби забезпечення роботи веб-порталу, пошуку, перекладу, зберігання та інформації.

Практичне значення роботи полягає в тому, що готовий проект дасть можливість знайти інформацію про космічні технології, запуски, орбітальні і міжпланетні апарати українською мовою. На сьогоднішній день така інформація часто неактуальна, неповна, інколи суперечлива. Україномовних Інтернет-ресурсів, присвячених космонавтиці, майже не існує.

В ході роботи над проектом був розроблений інформаційний ресурс, що може надавати інформацію про космічні технології і апарати українською мовою. При цьому для наповнення інформацією бази даних було застосовано парсинг і автоматичний переклад. Були досліджені і застосовані сучасні методи розробки програмного забезпечення. Окрім цього, під час роботи був проведений огляд та аналіз існуючих рішень, виділено вимоги до програмного продукту та цілі розробки.

Для розробки інформаційного ресурсу використовувалася мова програмування C#. Проект розроблено на платформі ASP.NET Core, з використанням бази даних SQLite.

Пояснювальна записка складається з таких розділів: вступ, основні розділи, висновки та список використаних джерел. Загальний обсяг пояснювальної записки проекту 115 сторінок.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Загальний опис предметної області

Дослідження і освоєння космічного простору завжди було і досі залишається областю використання останніх наукових досягнень і найбільш передових технологій. Без використання космічних технологій життя людей по всьому світу значно відрізнялося б від сьогоднішнього. Важливість космічних супутників очевидна навіть країнам, які не розвивають космічні технології самостійно, і не мають власних космодромів, ракет-носіїв і високотехнологічних виробничих потужностей. За необхідності, країна може замовити виведення свого супутника на орбіту у іншого космічного агенства або приватної компанії, або навіть купити доступ до даних супутника, що вже знаходиться на орбіті.

Рис. 1.1. Американські вчені тримають макет першого американського супутника – «Explorer 1», у натуральну величину

Людство почало запуски в космос 4 жовтня 1957 року, коли Радянський Союз запустив ПС-1, перший штучний супутник, який вийшов на орбіту Землі. Станом на кінець січня 2022 року в космос було запущено 12 293 об'єкти корисного навантаження. Окрім цього, в космосі відстежується близько двадцяти п’яти тисяч рукотворних об’єктів, що не є активними космічними апаратами. неіснуючі створені людиною об’єкти в космосі, головним чином на орбіті Землі, які більше не виконують корисної функції. Це космічне сміття — космічні апарати, що перестали працювати, залишені на орбіті ступені ракет-носіїв, різноманітні деталі, що були відділені в ході польотів. Особливо численні на навколоземній орбіті уламки від зіткнень ступенів ракет і космічних кораблів. Це лише об’єкти, що можна відстежити. Загальна чисельність рукотворних об’єктів в космосі будь-якого розміру оцінюється в сто тридцять мільйонів, більшість з яких зосереджена на низькій навколоземній орбіті.

Не менш важливою, а для звичайної людини, скоріше, навіть більш важливою є поява комп’ютерних мереж і об’єднання комп’ютерних мереж у одну мережу, відому як Інтернет.

Перші комп’ютерні мережі з’явилися у відповідь на запуск Радянським Союзом першого штучного супутника землі і відповідно, ракетно-ядерну загрозу. Їх розробка була профінансована Міністерством оборони США. Таким чином Міністерство оборони хотіло отримати надійну систему передавання інформації, що буде працювати навіть у випадку виходу з ладу частини вузлів внаслідок, наприклад, ядерного удару. На практиці, звичайно ж, лінії зв’язку виходять з ладу з набагато розповсюдженіших причин, але це приводить до втрати доступу до мережі невеликої кількості користувачів, і не впливає на роботу мережі в цілому.

Перші комп’ютерні мережі використовувалися співробітниками науково-дослідних інститутів для обміну файлами і електронною поштою. В ході їх розширення вдосконалювалося мережеве програмне забезпечення, з’явилися протоколи TCP, IP, система DNS. У 1989 році Тім Бернерс-Лі

розробив протокол HTTP — Hypertext Transfer Protocol, мову розмітки HTML, перший веб-сервер, що надавав HTML-документи, та перший веб-браузер, що перетворював HTML-документ у текст на екрані. Інформаційне середовище, яке він створив, стало називатися World Wide Web. З цього моменту використання комп'ютерних мереж стало доступне технічно необізнаним користувачам, і почався бурхливий розвиток Інтернету.

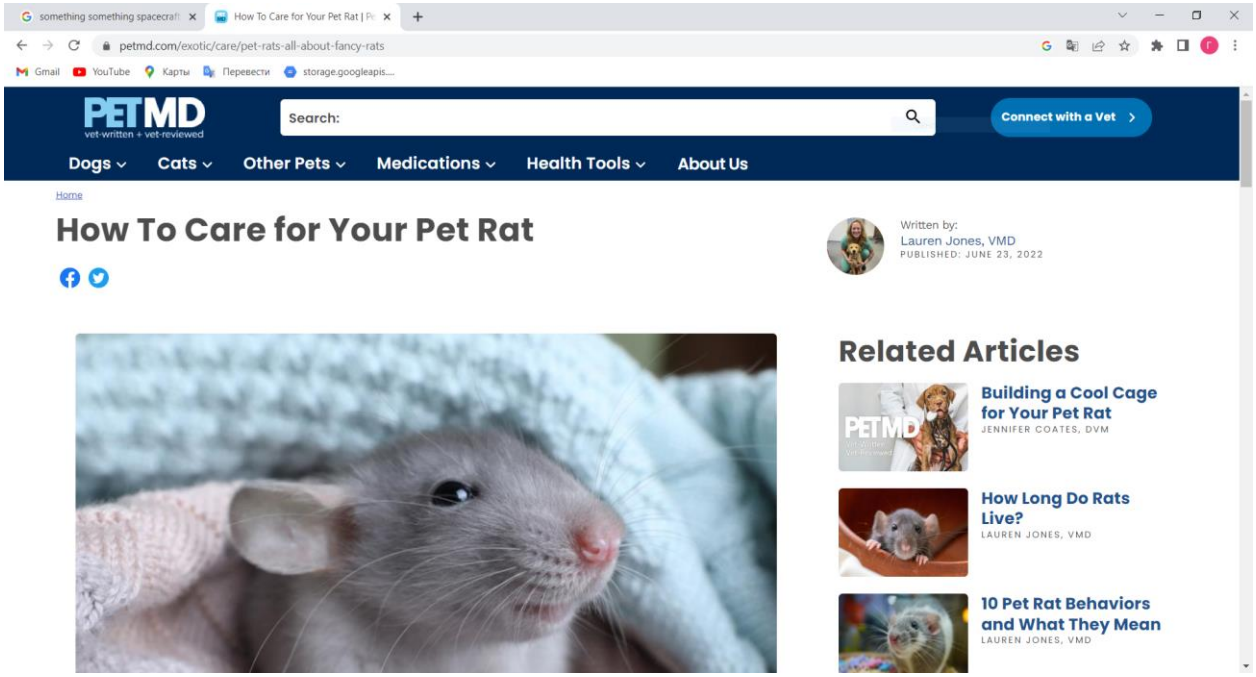


Рис. 1.2. Браузер Google Chrome відображає веб-сторінку

Веб-сайти можуть використовуватися для будь-яких цілей, що на певному етапі потребують отримання інформації з Інтернету, від бізнесу до розваг, від освіти до ведення щоденника. Веб-сайт розгортається на сервері — віддаленому комп'ютері, що приймає і обробляє HTTP-запити від клієнта, а у відповідь надсилає повідомлення, що обробляються браузером на боці клієнта і виводяться на екран у вигляді веб-сторінок.

1.2 Дослідження космосу — поняття, мета, завдання, засоби

Дослідження космосу — це використання астрономії та космічних технологій з метою дослідження космічного простору. У той час як дослідження космосу здійснюються в основному астрономами за допомогою телескопів, його фізичне дослідження, однак, проводиться як за допомогою роботизованих космічних зондів, так і за допомогою пілотованих польотів людини в космос. Фізичні дослідження космосу, як і його класична форма — астрономія, є одним із основних джерел космічної науки. Межею між атмосферним і космічним простором вважається лінія Кармана — уявна межа, що проходить по висоті сто кілометрів над рівнем моря. На цій висоті атмосфера настільки розріджена, що для генерації підйомної сили об’єкту доведеться розвинути першу космічну швидкість.

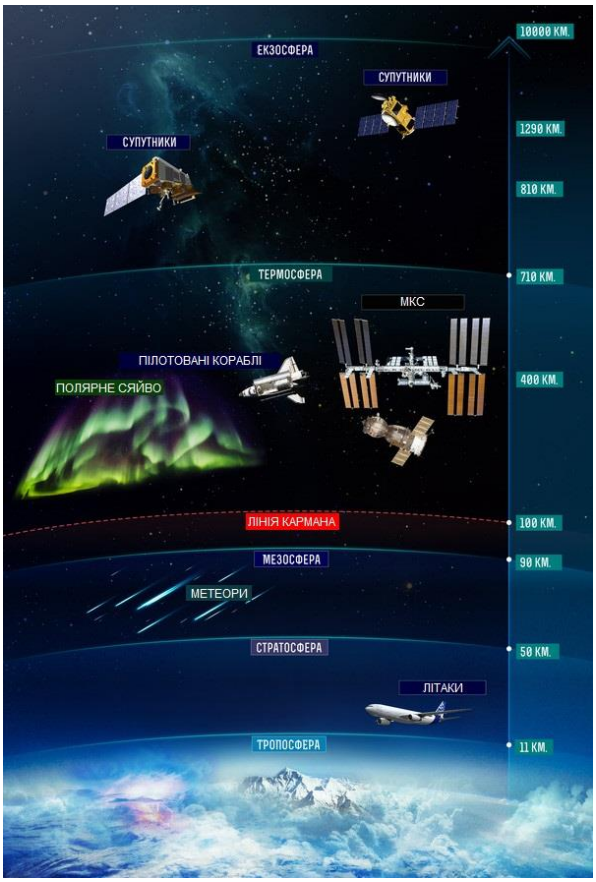


Рис. 1.3. Межі шарів атмосфери, лінія Кармана і висоти деяких орбіт

Діяльність людини в космосі, пряма, або опосередкована, через автоматичні космічні зонди, зокрема її тривалість і постійність, може бути самоціллю. Проте, використання космосу має чимало способів практичного застосування, від досліджень і комерційного використання космосу до побудови космічного поселення, або навіть мілітаризації космічного простору. Діяльність людини в космосі реалізується та підтримується завдяки розвитку та застосуванню космічних технологій, зокрема космонавтики та космічної інфраструктури, тим самим стимулюючи науковий прогрес.

Аргументами за проведення дослідження космосу є:

- розширення наукових знань;
- забезпечення військової та стратегічної переваги держави;
- національний престиж;
- об’єднання націй;
- забезпечення майбутнього виживання людства.

Сучасні та заплановані на близьке майбутнє космічні програми використовуються для:

- спостереження за Землею, супутники-шпигуни, метеорологічні супутники, картографічні супутники;
- збору інформації про космічний простір;
- забезпечення супутникового зв'язку, телебачення, Інтернету;
- забезпечення супутникової навігації;
- проведення експериментів в невагомості і унікальних умовах космічного простору;
- космічного туризму;
- захисту Землі від потенційно небезпечних об'єктів;
- колонізації космосу.

У той час як спостереження за об’єктами в космосі, відомі як астрономія, виникли ще до письменної історії, саме розробка великих і відносно ефективних ракет у середині двадцятого століття дозволила фізичному дослідженню космосу стати реальністю. Саме космічні апарати дають найбільшу частку даних. У таблиці нижче показані основні історичні етапи дослідження космосу.

Таблиця 1.1

Основні історичні етапи дослідження космосу

Часовий проміжок	Досягнення	Практичне значення
V тисячоліття до нашої ери — XVII століття нашої ери — спостереження неозброєним оком.	Передбачення положень Сонця, місяця, п’яти видимих неозброєним оком планет.	Календарі, навігація по зорям.
XVII століття — перша половина XX століття — поява телескопів та обсерваторій.	Відкриття нових об’єктів сонячної системи, зоряних об’єктів, покращення розуміння будови Всесвіту.	Практичне підтвердження законів фізики, використання їх для пошуку нових космічних об’єктів.
1945 — 1957 роки — розвиток ракетної техніки.	Вихід за межі земної атмосфери.	Дослідження властивостей земної атмосфери, впливу перевантажень та космічного середовища на живі організми, вдосконалення ракетної техніки, розвиток ракет як зброї.

Таблиця 1.1 (продовження)

Основні історичні етапи дослідження космосу

1957 — 1975 роки — космічна гонка.	Поява ракет-носіїв на основі міжконтинентальних балістичних ракет. Запуск безпілотних космічних апаратів на навколоземну орбіту, до Місяця. Дослідження внутрішніх планет Сонячної системи. Пілотовані космічні польоти. Висадка людини на Місяць. Перші космічні станції.	Поява метеорологічних супутників, супутників зв'язку, розвідувальних супутників. Експерименти із протисупутниковою зброєю. Поява космічного законодавства.
1975 — 1999 роки — кінець космічної гонки.	Поява космічних кораблів багаторазового використання. Місії до зовнішніх планет Сонячної системи, комет, астероїдів. Довгочасні космічні станції. Розвиток засобів доставки в інших країнах. Міжнародні місії.	Створення глобальних систем навігації (GPS, ГЛОНАСС). Поява космічних телескопів. Проведення довготривалих експериментів на орбіті. Розвиток міжнародної співпраці.

Таблиця 1.1 (продовження)

Основні історичні етапи дослідження космосу

XXI століття — сучасний етап дослідження космосу.	Побудова Міжнародної космічної станції. Перший китайський космонавт і космічна станція. Поява і швидкий розвиток приватних космічних компаній. Поява ракет багаторазового використання. Фокус на дослідженні Марсу з використанням Марсоходів.	Всебічне вивчення об’єктів Сонячної системи, особливо Марсу. Подальше зниження ціни запуску вантажу в космос. Контракти на запуски і постачання МКС з приватними компаніями. Перші приватні космонавти. Поява космічного туризму.

Отримання доступу до космосу та космічної діяльності людини було досягнуто через створення космічної промисловості, космічної інфраструктури, та міжнародного космічного права. Сьогодні міжнародним пріоритетом є забезпечення сталого розвитку космічної діяльності. Організація Об’єднаних Націй бачить необхідність сприяти довгостроковому сталому розвитку космічної діяльності, а Сполучені Штати вважають це ключовим пунктом своєї сучасної політики у відношенні космосу.

1.3 Веб-сайти

Веб-сайт — це набір веб-сторінок і пов’язаного вмісту, який ідентифікується загальним доменним іменем і публікується принаймні на одному веб-сервері. Гіперпосилання між веб-сторінками керують навігацією сайту, яка часто починається з домашньої сторінки. Будь-який веб-сайт може

містити гіперпосилання на будь-який інший веб-сайт, тому відмінність між окремими сайтами, як сприймає користувач, може бути розмитою.

Усі загальнодоступні веб-сайти разом складають Всесвітню павутину. Існують також приватні веб-сайти, до яких можна отримати доступ лише в приватній мережі. Таким може бути, наприклад, внутрішній веб-сайт компанії для її співробітників.

Веб-сайти, як правило, присвячені певній темі, наприклад новинам, освіті, торгівлі, розвагам або спілкуванню. Це може бути особистий веб-сайт, корпоративний веб-сайт для компанії, урядовий веб-сайт, веб-сайт організації тощо. Веб-сайти можуть створюватися окремою особою, компанією чи іншою організацією.

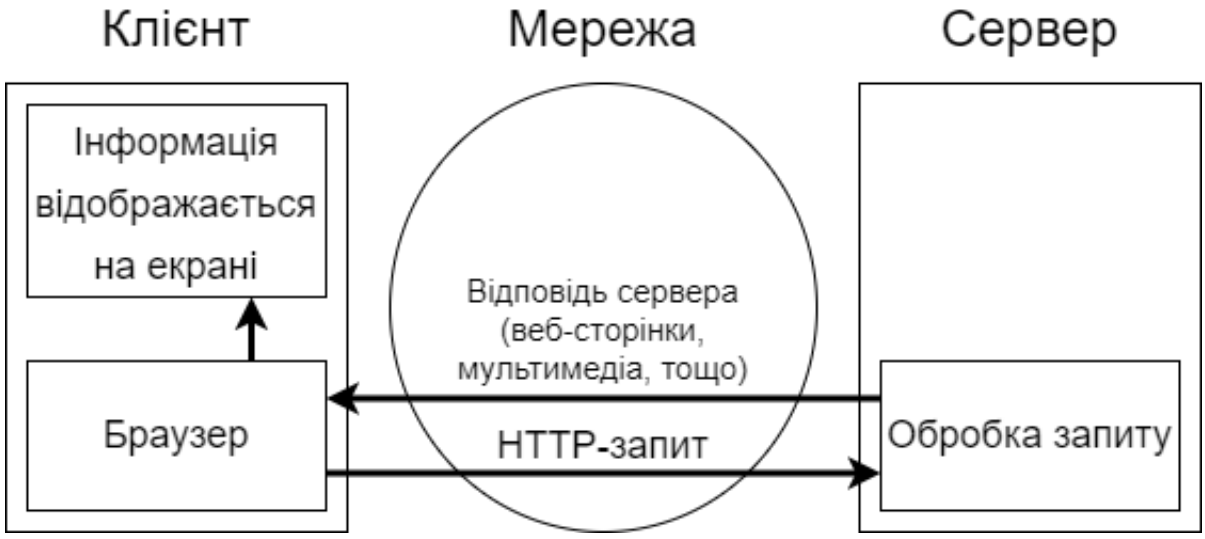


Рис. 1.4. Загальна схема роботи веб-сайту

Користувачі можуть отримувати доступ до веб-сайтів із різних пристроїв, включаючи настільні ПК, ноутбуки, планшети та смартфони. Розвиток смартфонів і кишенькових пристроїв у 2010-х роках призвів до появи адаптивного дизайну веб-сайтів. Адаптивний дизайн дозволяє веб-сайтам змінювати зовнішній вигляд сторінок відповідно до пристрою, з якого відбувається запит.

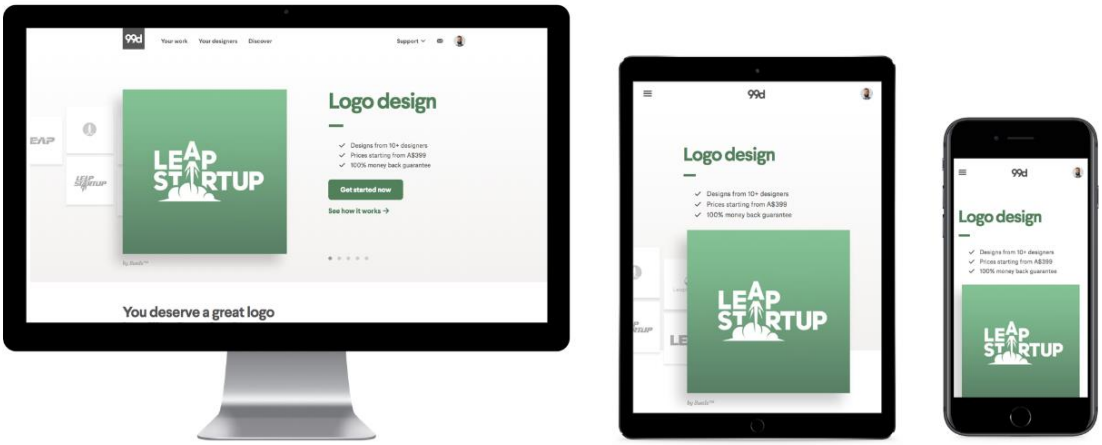


Рис. 1.5. Веб-сайт з адаптивним дизайном на різних пристроях: моніторі комп’ютера, планшеті, смартфоні

Програма, яка використовується на цих пристроях для перегляду веб-сайтів, називається веб-браузером. Деякі веб-сайти потребують реєстрації або підписки користувача для доступу до вмісту. Реєстрація використовується на більшості сучасних веб-сайтів, як правило, для розсилання електронної пошти. Серед сайтів, на яких доступна реєстрація чимало бізнес-сайтів, веб-сайтів новин, журналів, ігор, файлообмінників. На сайтах електронної пошти, у соціальних мережах, веб-сайтах дошок оголошень реєстрація обов’язкова.

Перші веб-сайти містили лише текст, скоро з’явилися зображення. Для додавання аудіо, відео та інтерактивності (наприклад, для веб-сторінки, що надає функцію калькулятора) стали використовуватися плагіни веб-браузера — програмні компоненти, що розширюють функції існуючої програми. Прикладами таких плагінів є Microsoft Silverlight і Adobe Flash Player. Сучасні браузери мають вбудовані можливості найпопулярніших плагінів. JavaScript також вбудований у сучасні браузери, що дозволяє серверу надсилати JavaScript-код у веб-браузер, на основі якого браузер інтерактивно змінює вміст сторінки та може надсилати подальші запити до сервера.

1.2.1 Критерії для аналізу аналогів

Під час підготовки до написання проекту було вирішено знайти та проаналізувати існуючі присвячені дослідженню космосу веб-сайти. Зявдяки цьому можна буде краще сформулювати бачення проекту. На початку роботи були виділені декілька основних критеріїв, за якими можна оцінювати такі веб-сайти:

– можливість переглядання списку запусків у космос в хронологічному порядку;

– можливість переглядання інформації про супутники і космічні апарати певної країни;

– можливість переглядання інформації про кожен космічний апарат окремо;

– можливість переглядання інформації про запуски, що планувалися як орбітальні, але з якихось причин не досягли орбіти;

– можливість перегляду статистики;

– зручний і зрозумілий інтерфейс, привабливий дизайн.

В Інтернеті існує декілька сайтів, що надають інформацію про космічні технології і апарати. Вони були оглянуті, проаналізовані і оцінені відповідно до виділених критеріїв.

1.2.2 Онлайн-енциклопедія «Вікіпедія»

Вікіпедія — загальнодоступна, багатомовна і безкоштовна онлайн-енциклопедія, що була заснована Ларрі Сенгером та Джиммі Вейлзом і запущена 15 січня 2001 року. Вікіпедія є найбільшою і найпопулярнішою онлайн-енциклопедією. Сьогодні цей сайт є п’ятим за популярністю сайтом у світі. Сайт вважається найповнішим джерелом інформації в усіх областях знань людства.

Матеріали сайту Вікіпедія можуть бути вільно використані, розповсюджені та вдосконалені на умовах ліцензій Creative Commons Attribution Share-Alike та GFDL. Вікіпедію підтримує некомерційна організація «Wikimedia Foundation». Організація фінансується завдяки добровільним пожертвам. Саме тому Вікіпедія не показує рекламу, не має платних послуг, а матеріали використовуються безкоштовно.

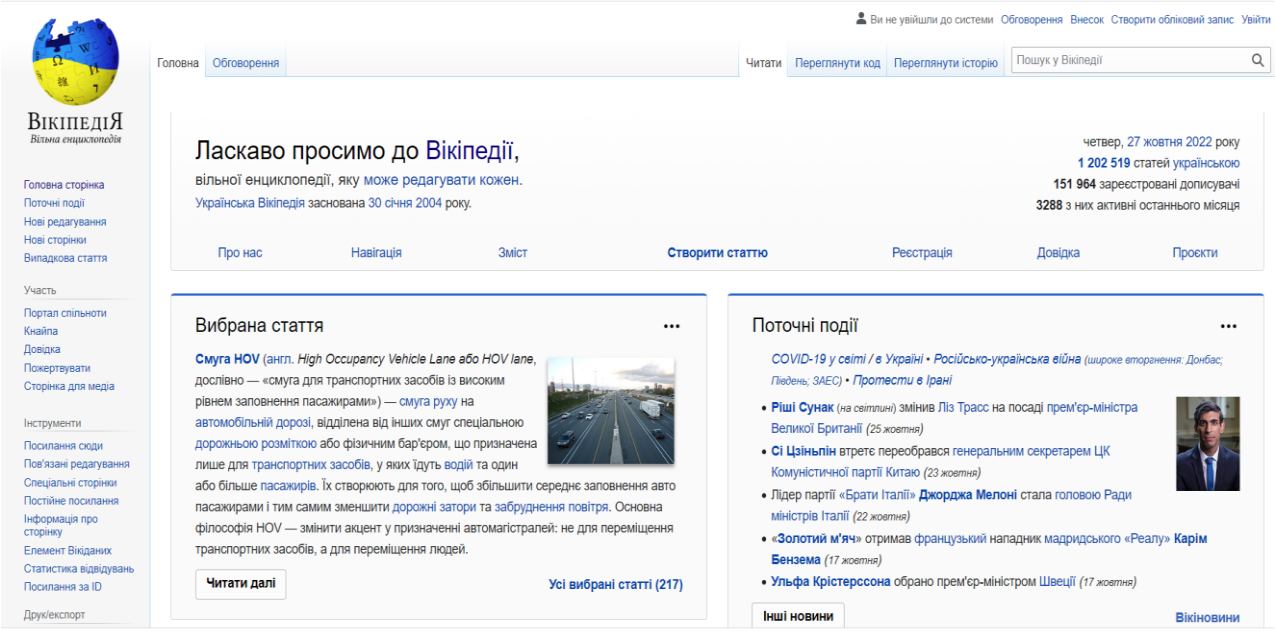


Рис. 1.6. Головна сторінка української Вікіпедії

На 15 січня 2022 року на сайті міститься понад п’ятдесят вісім мільйонів статей. Статті написані 329 мовами. З усіх мовних розділів найповніший, звичайно, англійський. Він містить понад шість мільйонів статей.

Кожен, хто має доступ до Вікіпедії, також має можливість редагувати її статті. Статті на Вікіпедії пишуться добровольцями з усього світу. Як правило, ці люди об’єднані у спільноту, і їх називають вікіпедистами.

Вікіпедія функціонує на принципах вікі. Вікі — це веб-сайт, що дозволяє користувачам самостійно змінювати вміст сторінок через браузер, з використанням спрощеної і зручнішої, порівняно з HTML, вікірозмітки

тексту. Вперше така система була використана у 1995 році. Її оснивними особливостями є:

– багаторазове редагування тексту;

– вікірозмітка, як спосіб спростити розмітку HTML;

– збереження історії змін сторінки і можливість її перегляду у всіх користувачів;

– доступ до редагування сторінок всім користувачам;

– сторінки, пов’язані через гіперпосилання в тексті;

– дуже простий в опануванні редактор сторінок.

Вікірозмітка (або вікітекст) — це мова розмітки, що використовуєтся для створення сторінок на сайтах, що функціонують на принципах вікі. Вікірозмітка не є однією мовою. Синтаксис різних вікірозміток залежить від програмного забезпечення кожного окремого вікі-сайту. Основною метою вікірозмітки є спрощення розмітки HTML. Після завершення редагування вікірозмітка перетворюється в HTML, що відображається у браузерах.

Вікіпедія використовує програмний рушій, що називається MediaWiki. MediaWiki — це один з найпотужніших рушіїв для вікі-сайтів, що був написаний спеціально для Вікіпедії. Більше того, він використовується на багатьох інших вікі-сайтах.

MediaWiki вільно розповсюджується на правах GNU General Public License. MediaWiki написаний мовою загального призначення PHP і для збереження даних використовує реляційну базу даних (MySQL, PostgreSQL, SQLite або Oracle DB).

Редагувати вміст статей Вікіпедії може кожен, хто зайшов на сайт, тобто, майже кожен користувач Інтернету. Правила Вікіпедії, окрім головних «П’яти основ», також визначаються і змінюються спільнотою. Таким чином, однією з проблем Вікіпедії є її надмірна бюрократизація. Нові користувачі можуть не орієнтуватися в правилах, або редагувати вміст статей згідно із застарілими правилами.

Ще одним об’єктом критики часто стає достовірність Вікіпедії. Правилами заборонена публікація на Вікіпедії оригінальних досліджень, а користувачі повинні надавати джерела інформації своїх змін. Дослідження показують, що достовірність Вікіпедії не поступається професійним друкованим і цифровим енциклопедіям.

Оскільки Вікіпедія має дуже велику кількість статей, під час огляду існуючих рішень були оглянуті лише статті, що містять інформацію про космічні технології.

Рис. 1.7. Сторінка Вікіпедії, присвячена першому штучному супутнику Землі

Переваги:

– присутній хронологічний список космічних запусків;

– є можливість переглядання статистики кожного року дослідження космосу, з окремими рядками для кожної країни;

– присутня основна інформація про космічні кораблі, ракети-носії, інфраструктуру, інші технології.

Недоліки:

– неповна інформація: багато космічних кораблів ще не не мають статей, про них є лише згадка з країною і датою запуску;

– списки космічних апаратів по країнам відсутні.

На Вікіпедії присутні 329 мовних розділів. За кількістю статей український мовний розділ посідає шістнадцяте місце. Хоча самі статті про космічні апарати українською мовою представлені досить повно, хронологічний список космічних запусків не має україномовної версії.

1.2.3 Сайт «Gunter's Space Page»

Сайт «Gunter's Space Page» присвячений збору інформації про космічні технології. Автор слідкує за подіями в космічній галузі, і своєчасно оновлює список запусків.

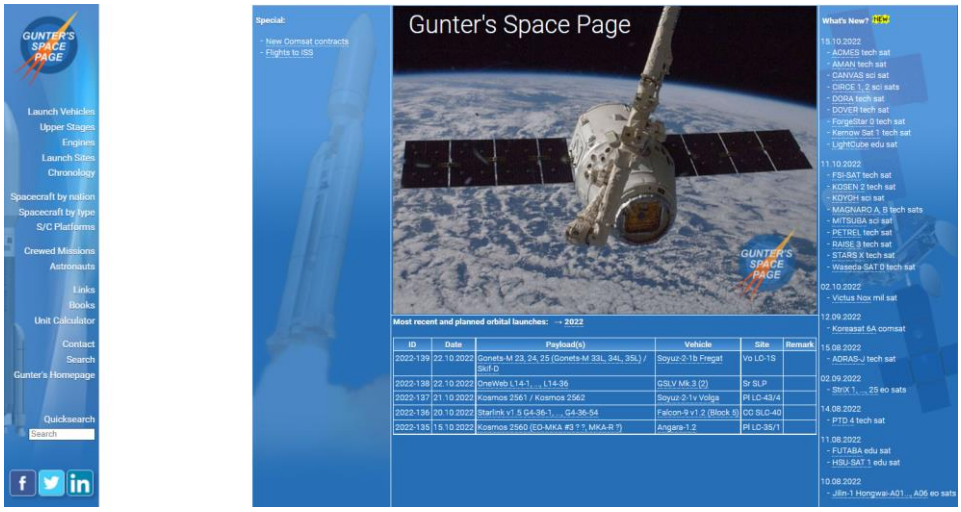


Рис. 1.8. Домашня сторінка сайту «Gunter's Space Page»

На сайті можна знайти інформацію про інші космічні технології: ракети, двигуни, верхні ступені, космодроми. Крім цього, автор надає список посилань на літературу, що може бути цікавою відвідувачам ресурсу.

На жаль, інформація на сайті часто неповна. Багато сторінок не існують, ще більше містять дуже мало інформації, деякі відображають

застарілі дані. Наприклад, сторінка, присвячена українському супутнику «Січ-2-30» не говорить про те, що супутник був запущений у 2022 році.

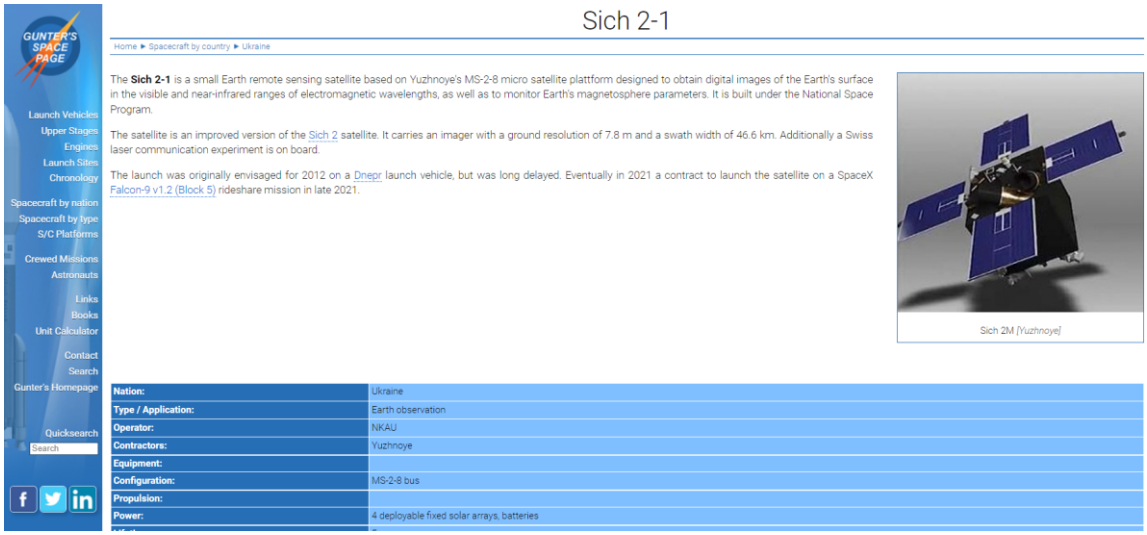


Рис. 1.9. Сторінка сайту «Gunter's Space Page» про супутник «Січ-2-30»

Сортування на сайті обмежується роком запуску, країною і призначенням космічного апарату. На деяких сторінках відсутній навіть інтерфейс.

Переваги:

– присутній хронологічний список космічних запусків;

– присутні списки космічних апаратів і фільтри за великою кількістю ознак;

– присутня основна інформація про космічні кораблі, ракети-носії;

– доступний перегляд статистики, статистика подається у вигляді кругових діаграм.

Недоліки:

– інформація про космічні апарати не містить ані NSSDC ID, ні SCN, які є універсальними ідентифікаторами космічного апарату, що сильно ускладнює пошук на інших ресурсах;

– інформація не оновлюється вчасно;

– деякі розділи сайту потребують оновлення.

1.2.4 Сайт «The Spacecraft Encyclopedia»

Сайт «The Spacecraft Encyclopedia» містить інформацію про космічні апарати, розділяє їх за країнами і за роком запуску. Окрім цього, на сайті зібрано чимало додаткової інформації про освоєння космосу в цілому.

Рис. 1.10. Домашня сторінка сайту «The Spacecraft Encyclopedia»

Окрім списку космічних запусків, на сайті також представлена статистика, що показує відсоток успішних запусків, кількість космічних апаратів різних країн, співвідношення цивільних і військових супутників, та багато іншого.

Сайт має велику кількість недоліків, особливо в плані архітектури і дизайну. Головна сторінка містить майже всю наявну на сайті інформацію, і посилання часто ведуть просто на іншу секцію головної сторінки. Окремих сторінок, присвячених космічним апаратам, а також ракетам-носіям, немає. Замість цього вся наявна інформація про космічний апарат подається прямо на сторінці зі списком запусків. Навігація по цій сторінці здійснюється шляхом переходу до присвяченої космічному апарату секції сторінки за посиланням, що розташоване у верхній частині сторінки.

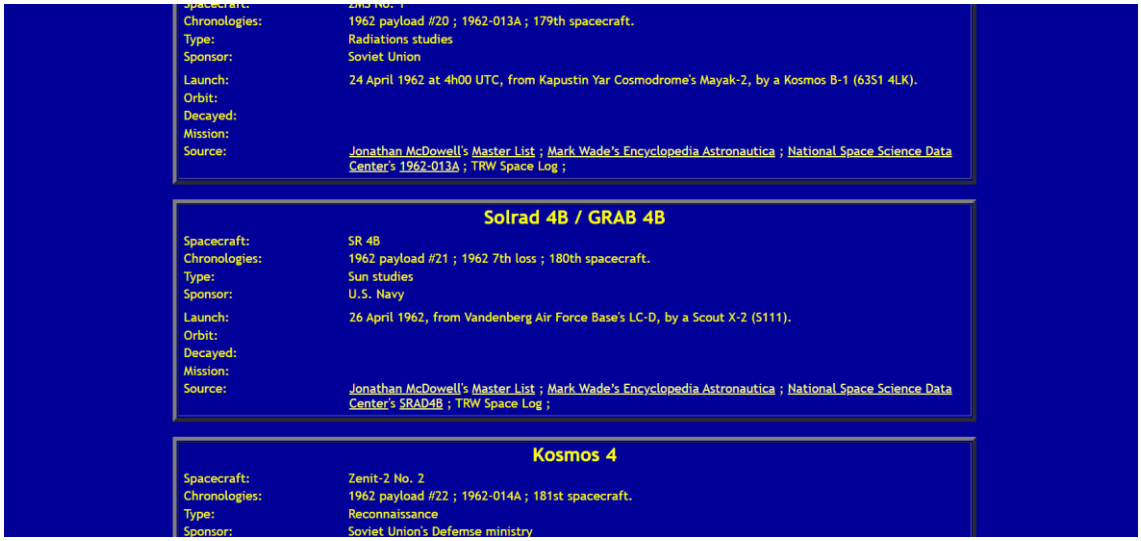


Рис. 1.11. Частина сторінка сайту «The Spacecraft Encyclopedia», присвяченої запускам 1962 року, секція «Solrad 4B / GRAB 4B»

Окрім того, хоча інформації про ранній період розвитку космонавтики достатньо, останні відомості про космічні запуски на цьому сайті внесені у 2017 році.

Варто зазначити, що інколи на сайті представлена дійсно цікава і унікальна інформація. Але у випадках, коли треба отримати загальні відомості про космічну техніку, його рекомендувати не можна.

Переваги:

– присутній хронологічний список космічних запусків;

– присутні списки космічних апаратів за призначенням, за країною, за роком запуску і за ракетою-носієм.

Недоліки:

– дуже застаріла інформація: немає жодних даних про запуски після 2017 року;

– немає окремих сторінок для кожного космічного апарата, космічному апарату присвчена секція в хронологічній таблиці, де вона відображається разом з іншими такими ж секціями;

– інтерфейс і переходи між сторінками дуже незручні, дизайн і архітектура сайту в цілому застарілі.

1.2.5 Сайт «NSSDCA Master Catalog»

Сайт надає доступ до бази даних НАСА, де міститься обширна інформація про космічні апарати, їх призначення, і забрані в польоті наукові дані. Архів «NASA Space Science Data Coordinated Archive (NSSDCA)» є архівом даних багатьох місій NASA. В архіві зібрані дані з астрономії та астрофізики, фізики сонячної та космічної плазми, планетології та вивчення Місяця. NSSDCA співпрацює з іншими архівами NASA, що спеціалізуються на космічній галузі і надають доступ до даних дослідникам, а в деяких випадках і широкій громадськості.

NSSDCA підтримує активних дослідників космічної фізики та астрофізики. Веб-сервіси дозволяють NSSDCA підтримувати взаємодію з громадськістю. Ця підтримка надається у формі інформації про космічні кораблі та доступу до цифрових версій вибраних зображень. NSSDCA також надає доступ до частин своєї бази даних, що містить інформацію, що архівується в NSSDCA, космічні кораблі та експерименти.

NSSDCA Master Catalog надає інформацію у відповідь на запити. Запити можуть включати:

- запити про космічні апарати;
- запити про дані експериментів;
- запити про дані, що збираються;
- запити про персонал, пов’язаний із процесом збору даних або проведенням наукових експериментів;
- запити про публікації, написані на основі даних з архівів НАСА, або відносяться до експериментів і місій НАСА;
- планетарні карти;
- нові і нещодавно оновлені колекції даних;
- події, пов’язані з дослідженням сонячної системи.

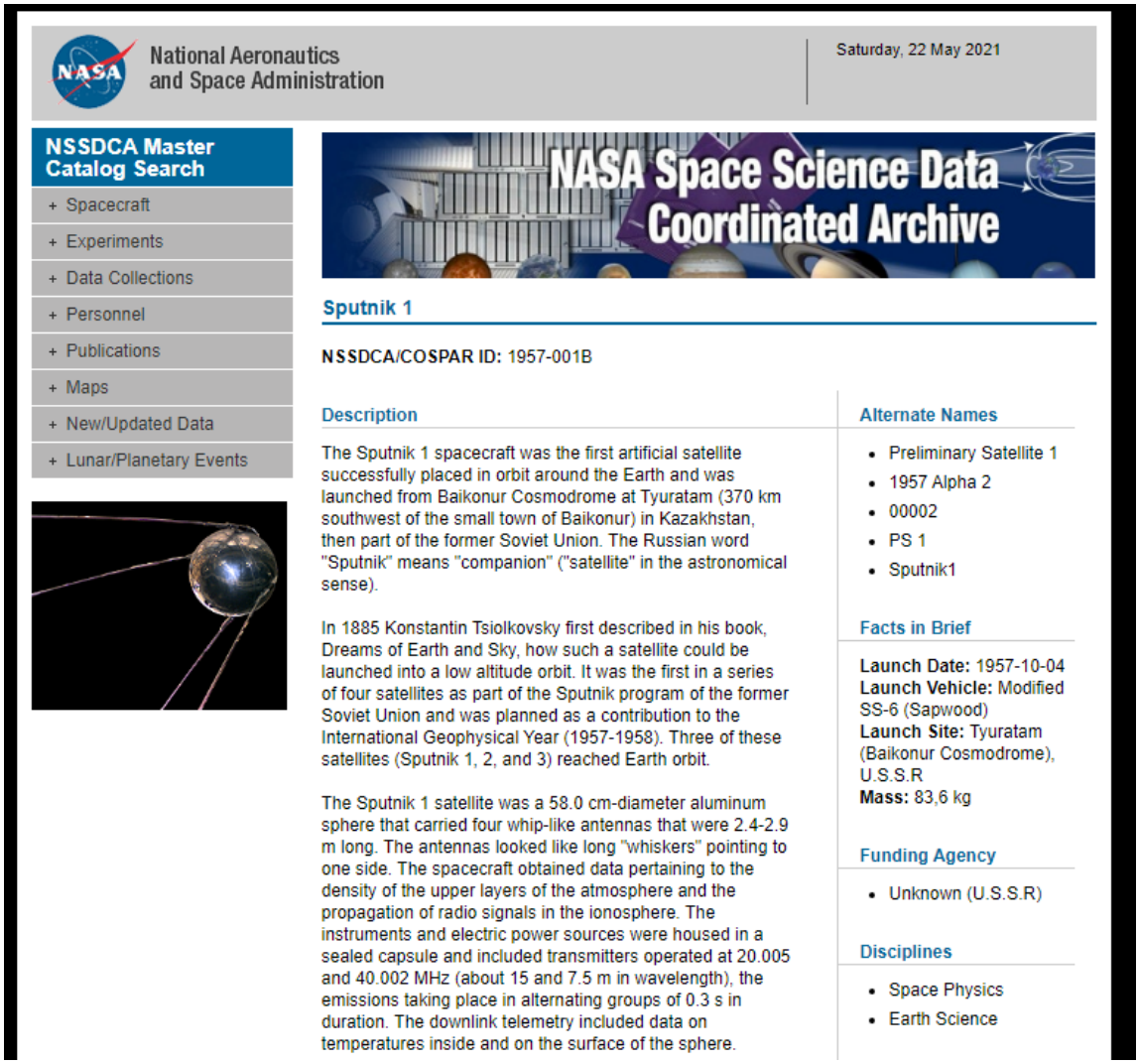


Рис. 1.12. Сторінка сайту «NSSDCA Master Catalog»

Сайт надає доступ до бази даних НАСА. Дані, які можна тут отримати, є унікальними. Але сам сайт є, по великому рахунку, лише механізмом відправки запитів в базу даних. І хоча ці дані будуть дуже цінними для фахівців у науках про космос, користувач, що просто цікавиться дослідженням космосу, навряд чи знайде сайт зручним.

1.2.6 Підсумковий аналіз розглянутих рішень

Перед пошуком аналогів було визначено найважливіші функції, які повинен виконувати сайт, подібний тому, що створюється в цьому проекті. В ході пошуку існуючих рішень було ретельно розглянуто та проаналізовано існуючі аналоги. Результати підсумкового аналізу наведені у таблиці 1.2.

Таблиця 1.2

Підсумковий аналіз

Критерій	Вікіпедія	Gunter's Space Page	The Spacecraft Encyclopedia	NSSDCA Master Catalog
Хронологічний порядок запусків	+	+	+	+
Перегляд космічних апаратів за країною	–	+	+	–
Інформація про кожен апарат окремо	+–	+	–	+
Невдалі запуски	+	+	+	+
Перегляд статистики	+	+	+	–
Зручність інтерфейсу	+	+	–	+–

1.4 Основні вимоги

Було розглянуто та проаналізовано існуючі сайти з інформацією про дослідження космосу і космічні апарати. В результаті пошуку та аналізу існуючих рішень було визначено критерії оцінки аналогів, знайдено три веб-сайти про космічні технології і апарати, виявлено їх переваги та недоліки.

На основі результатів аналізу існуючих рішень та проведеного дослідження предметної області були встановлені вимоги до проекту. Необхідно створити веб-портал про космічні апарати і технології. Щоб користувач мережі міг зручно використовувати цей веб-портал, необхідно надавати йому наступні можливості:

– переглядати інформацію про успішні і невдалі запуски у вигляді хронологічної таблиці;

– отримувати детальну інформацію про космічні апарати і технології, призначення, деталі місії, перебіг польоту;

– перегдядати статистику надійності запусків ракет різних країн, та статистику за певний рік.

1.5 Функціональні можливості

Основним джерелом вимог до веб-порталу, що розробляється в даному проекті, є предметна область і її особливості. Виходячи з огляду предметної області, можна визначити необхідний користувачу функціонал. Він включатиме в себе:

– отримання хронологічного списку запусків космічних апаратів;

– отримання інформації про кількість успішних і провальних космічних запусків однієї країни, або за один рік;

– отримання статистики космічних запусків однієї країни, або статистики певного року дослідження космосу, або статистики запусків певної країни за певний рік;

– отримання списку ракет-носіїв або космічних апаратів, розроблених в одній країні;

– отримання детальної інформації про космічні апарати, їх призначення, характеристики, перебіг польоту;

– отримання інформації про ракети-носії;

– можливість адміністрування системи: внесення нової інформації, редагування неточної;

– за умови наявності прав адміністратора можливість створити ще один обліковий запис для того, щоб додати нового адміністратора.

Для того, щоб наочно зобразити поставлені в проекті вимоги і функціонал веб-сайту, який буде створено під час роботи над проектом, була підготована діаграма використання. Її зображено на рисунку 1.13.

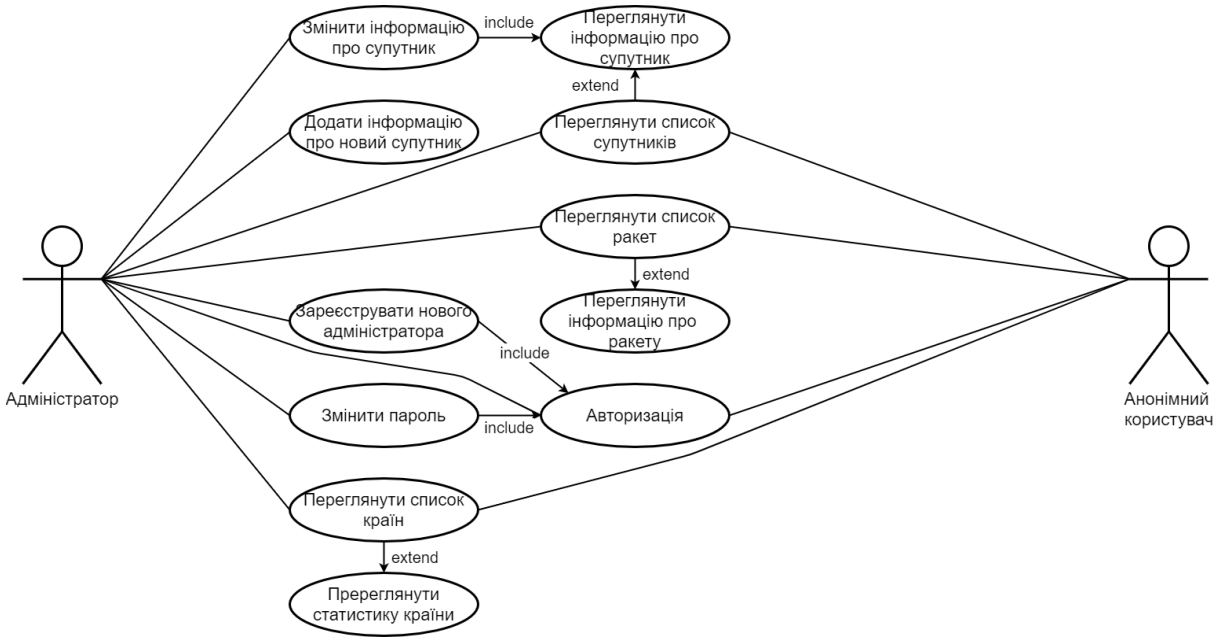


Рис. 1.13. Діаграма використання, створена на основі сформованих вимог до проекту

Якщо вдасться створити веб-сайт з переліченим функціоналом, то користувач зможе отримувати інформацію про ракети-носії, космічні технології та апарати, дізнатися про мету їх створення і запуску, деталі будови. На основі статистики, що буде відображатися, користувач зможе скласти думку про надійність космічної техніки в різні періоди дослідження космосу. Можна буде переглянути списки космічних апаратів однієї з країн, що здійснює запуски в космос, та отримати уявлення про її внесок в освоєння космічного простору.

Висновок до розділу 1

В цьому розділі пояснювальної записки був представлений досить детальний опис предметної області. Окрім цього, було розглянуто та проаналізовано існуючі сайти з інформацією про дослідження космосу і космічні апарати. Опис предметної області надає головні причини розвитку космонавтики, завдання, що стоять перед нею, її практичне значення. Також опис предметної області містить загальну інформацію про веб-сайти. Описано коротку історію комп'ютерних мереж в цілому, появу веб-сайтів і основні принципи їх роботи.

Були визначені вимоги до веб-сайту, що надає інформацію про космічні апарати і технології. Під час підсумкового аналізу були виявлені особливості, переваги та недоліки існуючих аналогів. З усіх розглянутих веб-сайтів лише один відповідає всім вимогам, маючи при цьому і свої недоліки. Лише один з веб-сайтів містить інформацію українською мовою.

Таким чином, актуальною є задача створення мережевого ресурсу, що надавав би інформацію про космічні технології і апарати.

У цьому розділі роботи були встановлені вимоги до веб-сайту, що буде розроблений в ході роботи над проектом. Основними вимогами до проекту є вимоги, що стосуються функцій, які доступні користувачу. Ці функції були детально описані, і на основі цих функцій була створена діаграма використання.

На основі цих вимог можна буде краще побудувати архітектуру програми і написати інструкцію для роботи з програмою для користувача і адміністратора.

РОЗДІЛ 2
ВИБІР ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ПРОГРАМИ

2.1 Проектування архітектури додатку

Перед розробкою будь-якого програмного забезпечення варто спроектувати його архітектуру. Архітектура програмного забезпечення — це фундаментальні структури програми та методи створення таких програм.

Архітектура функціонує як план програми та проекту, на основі якого можна приблизно визначити завдання, які необхідно виконати розробникам. Архітектура програмного забезпечення полягає в тому, щоб на основі вимог до програмного забезпечення обрати структуру програми так, щоб вона якнайкраще виконувала покладені на неї функції.

Наприклад, системи керування космічним кораблем «Спейс Шаттл» (або будь-яким іншим об’єктом, для якого затримки у керуванні будуть критичними) повинні бути дуже швидкими та дуже надійними. Для цього використовуються системи обчислень в реальному часі.

З точки зору процесів розробки архітектура програмного забезпечення — це спосіб структурування складної системи, який дає уявлення про її елементи, функції, властивості, і на основі якого можна розпочинати розробку. Бажані якості системи вказані у таблиці 2.1.

Таблиця 2.1

Бажані якості програмної системи

Категорія	Властивість	Опис
Проектування	Цілісність	Послідовність і злагодженість дизайну компонентів або модулів.
	Гнучкість	Легкість внесення змін до системи.
	Придатність до повторного використання	Здатність компонентів або рішень бути придатними для використання в інших програмах.

Таблиця 2.1 (продовження)

Бажані якості програмної системи

Використання	Сумісність	Здатність системи успішно працювати шляхом обміну інформацією з іншими системами.
	Керованість	Легкість керування програмою системними адміністраторами.
	Надійність	Здатність системи прожовжувати роботу протягом тривалого часу.
	Масштабованість	Здатність системи витримувати збільшене навантаження без зниження продуктивності системи, або легкість розширення.
	Безпека	Захищеність системи від злочинних дій: викрадення інформації, пошкодження системи.
	Продуктивність	Здатність системи виконувати запити протягом прийнятного проміжку часу.
	Доступність	Час, протягом якого система активна і може виконувати запити.
Підтримка	Підтримуваність	Доступність інформації, необхідної для виявлення та вирішення проблеми, коли система не працює належним чином.
	Портативність	Здатність системи працювати в різних апаратних і програмних середовищах.
Використання	Зручність	Відповідність системи потребам користувачів.

Під час розробки заздалегідь спроектована структура програми забезпечує ряд переваг.

Це дає основу для аналізу поведінки системи до того, як система буде створена. Такий аналіз дає можливість перевірити, чи майбутня програмна система відповідає вимогам до її фактичного створення. Оскільки зміна архітектури програми після її створення це дуже довгий і важкий процес, проектування суттєво зменшує ризики.

Спроектована архітектура системи полегшує комунікацію із замовниками, сприяючи створенню системи, яка краще задовольняє їхні потреби. Огляд архітектури програмної системи з точки зору замовників допомагає краще зрозуміти наслідки заявлених вимог і проектних рішень, що ґрунтуються на них. Архітектура дає можливість повідомляти про рішення розробників до впровадження системи, що дозволяє скоректувати їх, поки це ще відносно легко.

Архітектура програмного забезпечення допомагає зменшити ризики та ймовірність відмови.

Архітектура програмного забезпечення — це один із засобів управління витратами в складних IT-проектах.

Архітектура програмного забезпечення дозволяє повторно використовувати елементи і рішення. Повну архітектуру програмного забезпечення або її частини, окремі архітектурні стратегії та рішення, можна повторно використовувати в системах, до яких сформовані схожі вимоги, що також зменшить витрати часу на розробку і зменшить кількість помилок за рахунок вже перевірених і відпрацьованих рішень.

За час існування World Wide Web веб-сайти стали надзвичайно популярними і пройшли свою історію розвитку. Тому на сьогодні існує багато архітектурних стилів, що застосовуються саме для веб-сайтів і дозволяють отримати ті чи інші властивості.

Статична веб-сторінка — це веб-сторінка, що відображається у веб-браузері користувача в тому вигляді, в якому вона зберігається на сервері.

Статична веб-сторінка, як правило, відображає однакову інформацію для всіх користувачів. Статичні веб-сторінки є HTML-документами, які зберігаються у файловій системі сервера та надаються через HTTP. Статичні веб-сторінки підходять для вмісту, який ніколи не потребує оновлення. Веб-сайти зі статичними сторінками мають свої переваги:

– менше вразливостей з точки зору безпеки;

– краща продуктивність на боці користувача;

– немає залежностей від інших програм: баз даних, сторонніх серверів.

Проте, підтримка великої кількості статичних сторінок у вигляді файлів може бути непрактичною без автоматизованих інструментів, таких як генератори статичних сайтів. Будь-яка персоналізація чи інтерактивність має працювати на стороні клієнта. Ці недоліки призвели до того, що сьогодні майже повсюдно використовуються динамічні веб-сторінки.

Рис. 2.1. Схема роботи веб-сайту зі статичними сторінками

Динамічна веб-сторінка — це веб-сторінка, що створюється за допомогою програмного коду. Створення кожної веб-сторінки, включаючи налаштування додаткової обробки на стороні клієнта, може контролюватися сервером. На стороні клієнта браузер формує динамічну веб-сторінку за допомогою вбудованого JavaScript, якщо код був надісланий сервером разом із сторінкою. JavaScript може взаємодіяти зі сторінкою через об'єктну модель документа. Якщо веб-сторінка є динамічною на стороні клієнта, вона може бути розміщена на сервері як звичайна статична сторінка.

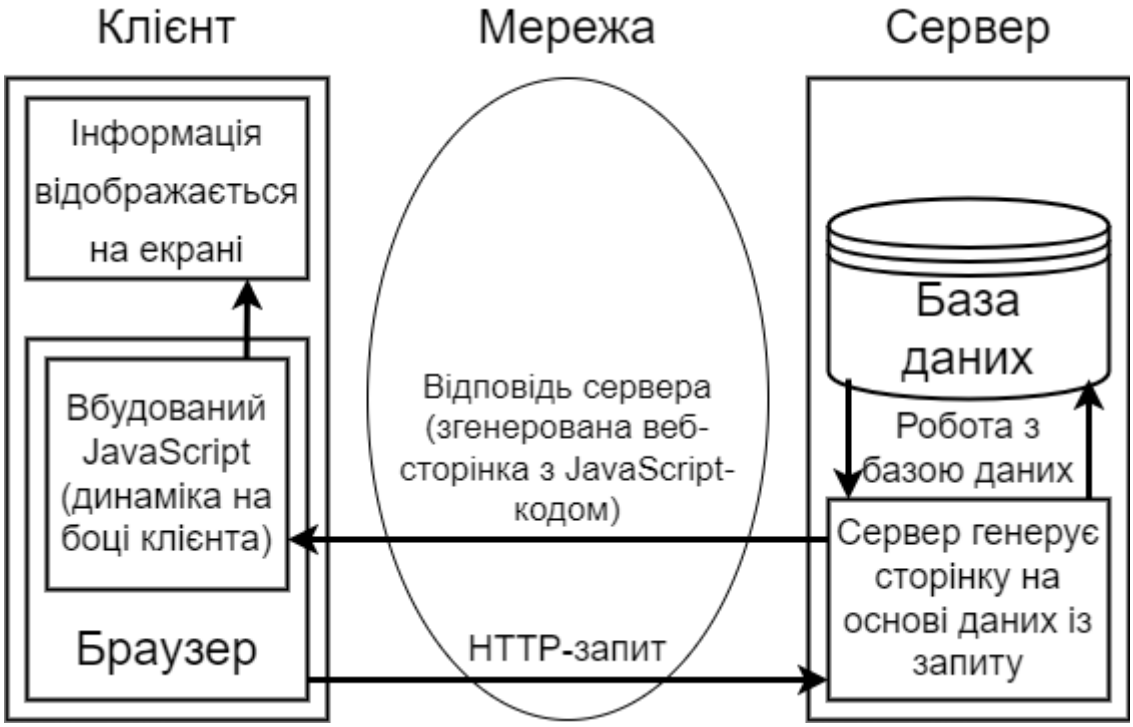


Рис. 2.2. Схема роботи сайту з динамічними веб-сторінками

Сервіс-орієнтована архітектура – це архітектурний стиль, який зосереджується на окремих сервісах. Кожен сервіс це окрема функціональна одиниця, що використовується і оновлюється незалежно від інших. Ця архітектура орієнтується на незалежність від конкретних реалізацій і систем.

Сервіс має чотири основні властивості:

– логічно представляє певну функцію з конкретним результатом;
– самодостатній;

– не розкриває деталей реалізації;

– може включати в себе інші сервіси.

Сервіс-орієнтована архітектура розглядає систему як об’єднання сервісів, кожен з яких виконує певний етап роботи. принцип SOA спільний з модульним програмуванням. Сервіси пов’язані між собою через інтерфейси. Таким чином, сервіс повинен реалізувати інтерфейс, після чого він може замінити собою інший сервіс, що реалізовує такий самий інтерфейс.

Мікросервісна архітектура — варіант сервіс-орієнтованої архітектури — це архітектурний стиль, який розглядає програму як набір слабко пов’язаних, мініатюрних сервісів, які спілкуються через легкі протоколи. Ціль такої архітектури полягає в тому, що команди можуть розробляти та розгортати свої сервіси незалежно одна від одної. Це дозволяє компаніям розробляти великі проекти і забезпечувати підтримку, відмовостійкість та легкість оновлень, а один мікросервіс може застосовуватися і окремо від проекту.

Мікросервісна архітектура має свої слабкі місця. Так, особливу увагу треба приділяти розробці інтерфейсів. Також дуже важливо правильно підібрати розмір мікросервісу. Надто великі сервіси не дадуть відчутних переваг, а при дуже подрібненій структурі програми збільшаться витрати часу на розробку і розгортання проекту в цілому.

Монолітна архітектура – архітектура, згідно з якою весь код і всі функції проекту об’єднані в одній програмі на одній платформі. На відміну від мікросервісної архітектури, цей підхід не жертвує легкістю розробки і розгортання.

По-перше, монолітні програми можуть мати кращу продуктивність, ніж модульні програми. Також їх легше тестувати та налагоджувати, оскільки з меншою кількістю елементів менше сценаріїв, які треба тестувати. Легше вносити зміни, коли весь код знаходиться в одному місці, і легше розгортати програму, якщо це всього один виконуваний файл або папка. Окрім цього, не виникає необхідності налаштовувати зв’язок і сумісність між мікросервісами.

На початку життєвого циклу розробки програмного забезпечення зазвичай легше використовувати монолітну архітектуру. Єдина кодова база також спрощує логування, моніторинг продуктивності та інші завдання на етапі розробки.

Для складних програм із частими змінами коду, або ключовими вимогами до масштабованості цей стиль не підходить. Тим не менш, монолітна архітектура зазвичай краща для невеликих програм і досі широко використовується.

2.2 ASP.NET Core

ASP.NET — це платформа із відкритим вихідним кодом, призначена для веб-розробки та створення динамічних веб-сторінок. Назва розшифровується як Active Server Pages Network Enabled Technologies. Перша версія з’явилася у січні 2002 року і була наступником технології Microsoft Active Server Pages. ASP.NET побудовано на Common Language Runtime, що дозволяє програмістам писати код ASP.NET за допомогою будь-якої підтримуваної мови .NET.

Наступником ASP.NET є ASP.NET Core. Нова платформа використовує новий компілятор .NET з відкритим вихідним кодом ("Roslyn"), і працює в різних операційних системах. Незважаючи на те, що платформа була повністю переписана, вона має високий ступінь концептуальної сумісності з ASP.NET. ASP.NET Core підтримує керування версіями так, що різні програми, що розробляються на одній машині, можуть орієнтуватися на різні версії ASP.NET Core, що було неможливо з попередніми версіями ASP.NET. Перша версія платформи з’явилася в червні 2016 року.

Ключові особливості ASP.NET Core:

– вбудована віртуальна машина CLR, що дозволяє писати проекти всіма мовами, що підтримуються в CLR;

– єдиний стек для створення web UI та web API;

– вбудований сервер IIS для розгортання проектів;

– новий легкий і модульний конвеєр для обробки HTTP-запитів;

– модульна структура, що надає можливість отримання необхідних модулів (пакетів) через менеджер пакетів NuGet;

– оптимізація під використання мережевих і хмарних технологій;

– вбудований інструментарій для впровадження залежностей;

– кроссплатформенна розробка і розгортання на Windows, Linux і MacOS;

– створення програм для різних версій ASP.NET Core на одному й тому самому комп’ютері.

– розвиток платформи як відкритий код.

2.3 ASP .NET Core Razor Pages

ASP .NET Core Razor Pages — це технологія ASP.NET, яка використовується для створення динамічних веб-сторінок за допомогою мов програмування C# або Visual Basic. Razor — це механізм перегляду з простим синтаксисом, який був випущений як частина MVC 3. Razor став компонентом AspNetWebStack, а потім перейшов у ASP.NET Core. Razor Pages надає альтернативу поширеному у веб-розробці паттерну Model-View-Controller.

Синтаксис Razor — це синтаксис розмітки, заснований на мові програмування C#. Razor забезпечує оптимізований синтаксис для генерації HTML, орієнтований на код, але з мінімальним переходом між HTML і кодом. Кожна сторінка Razor складається з файла .cshtml, що є сумішшю конструкцій HTML, CSS і C#, та файла .cshtml.cs, що написаний мовою C# і відповідає за поведінку сторінки. Завдяки цьому в одній сторінці об’єднуються дані і методи роботи з ними. Інші переваги включають підтримку IntelliSense та можливість unit-тестування.

2.4 ADO.NET Entity Framework

Entity Framework Core — це легка, відкрита і кроссплатформенна технологія для доступу до даних від компанії Microsoft.

Object-relational mapping — це спосіб перетворення сутностей бази даних в об’єкти в системі типів об’єктно-орієнтованої мови програмування. Таким чином програміст отримує доступ до інформації в базі даних в зручній для себе формі, і незалежно від конкретної реалізації бази даних. Entity Framework Core дозволяє розробнику працювати з базою даних, використовуючи об’єкти .NET і прибирає необхідність писати більшу частину коду доступу до даних.

Entity Framework Core працює на основі платформи .NET Core. Тобто його можна використовувати у консольних додатках, програмах з графічним інтерфейсом, веб-сервісах, та інших типах програм, що працюють на платформі .NET Core. Як і .NET Core, Entity Framework Core може використовуватися із трьома операційними системами: Windows, Linux і MacOS.

Entity Framework Core фокусується на сутностях і зв’язках. Типи сутностей є сукупністю кількох типізованих полів – кожне поле відображається на певному стовпці в базі даних – і можуть містити інформацію з кількох таблиць. Типи сутностей можуть бути пов’язані один з одним, незалежно від зв’язків у фізичній схемі. Типи сутностей утворюють клас об’єктів, яким відповідають сутності.

Entity Framework Core використовує бібліотеку LINQ для створення запитів і отримання результатів з бази даних. LINQ (Language Integrated Query) — це компонент Microsoft .NET Framework, який додає можливості створення запитів у базу даних мовам програмування .NET. LINQ розширює мову, додаючи вирази запитів, які схожі на оператори SQL, і їх можна використовувати для зручного вилучення й обробки даних із масивів,

перелічуваних класів, документів XML, реляційних баз даних і сторонніх джерел даних.

Метод «code first» («спочатку код») дозволяє написати готовий спосіб доступу до бази даних, не створюючи саму базу даних вручну. Для цього необхідно спочатку створити клас контексту бази даних і класи сутностей, що відобразять сутності і всі зв’язки між ними в майбутній базі. Процес, що відобразить об’єктно-орієнтовану модель в схему бази даних, називається міграцією. Таким чином, після проведення міграції за допомогою Entity Framework Core програміст отримає готову базу даних, залежно від вибраного провайдера бази даних.

2.5 ASP.NET Core Identity

ASP.NET Core Identity є вбудованою в ASP.NET Core системою аутентифікації та авторизації. Система дозволяє створювати облікові записи користувачів, керувати їми, або використовувати для аутентифікації облікові записи Google, Facebook, Twitter або Microsoft.

ASP.NET Core Identity можна налаштувати на використання бази даних для зберігання імен користувачів, паролів і даних профілю. Microsoft Visual Studio дозволяє додавати основу для системи аутентифікації ще на етапі створення проекту. У такому випадку в проект заздалегідь будуть внесені необхідні зміни. Окрім цього, будуть згенеровані сторінки Razor Pages, призначені для базових операцій облікових записів: входу і виходу, створення облікового запису і зміни пароля.

ASP.NET Core Identity використовує Entity Framework Core для операцій з даними користувачів. При включенні Identity в програмі також автоматично згенерується клас контексту бази даних.

2.6 База даних SQLite

SQLite — це легка реляційна система керування базами даних, написана мовою програмування C. SQLite належить до вбудованих баз даних. Тобто, це не окрема програма, на відміну від, наприклад, Microsoft SQL Server. Це бібліотека, яку розорбники вбудовують у свої програми. Такі системи баз даних є дуже поширеними, вони використовуються у браузерах, операційних системах, мобільних телефонах і багатьох інших програмах. Сьогодні більшість мов програмування мають спосіб взаємодії з SQLite.

SQLite була розроблена, щоб дозволити програмам працювати без встановленої системи керування базою даних і контролю адміністратора бази даних. На відміну від клієнт-серверних систем керування базами даних, SQLite не має автономних процесів, до яких звертається програма, що використовує базу. Замість цього SQLite інтегрує бібліотеку SQLite в код програми. Таким чином програма може виконувати запити до бази даних через виклики функцій SQLite, що зменшує затримку в операціях з базою даних, і не вимагає додаткових ресурсів на зв’язок між процесами. SQLite зберігає базу даних як єдиний кроссплатформенний файл на комп’ютері, дозволяючи кільком процесам або потокам отримувати доступ до бази даних одночасно.

SQLite має наступні переваги та особливості:

– SQLite не потребує установки, попереднього налаштування або адміністрування;

– кроссплатформенність: SQLite працює в найрізноманітниших опреаційних системах, окрім найпопулярніших Windows, Linux та Mac OS може працювати на BSD, Solaris, VxWorks, мобільних операційних системах (iOS, Android);

– відкритий код, що можна використовувати з будь-якою метою;

– бібліотека самодостатня і не має залежностей від інших бібліотек;

– стовідсоткове покриття коду тестами;

– транзакції зберігають атомарність, послідовність, ізольованість, і стійкість (ACID) навіть після збоїв операційної системи або втрати електроживлення;

– SQLite швидша навіть за деякі системні операції введення виведення, хоча на практиці SQLite витрачає час, щоб забезпечити надійну фіксацію кожної транзакції;

– надзвичайна компактність: бібліотека займає менш ніж 350 кілобайт, на відміну від клієнт-серверних баз даних, де програма часто займає гігабайти;

– підтримка баз даних розміром більше терабайта, і розмірів рядка більше гігабайта (довжина 2^{31}-1 або 2147483647);

– база даних зберігається як єдиний файл на диску, і за рахунок кроссплатформенності може бути переміщений і використаний на інших пристроях;

– при всій мініатюрності SQLite підтримує повноцінний SQL.

2.7 Проектування серверної частини

У побудові розподілених мережевих програм в більшості випадків використовується клієнт-серверна архітектура.

Клієнт-серверна архітектура — це структура розподілених систем, що розподіляє завдання між постачальниками послуг, що називаються серверами, і відправниками запитів, що називаються клієнтами. Зазвичай клієнти і сервери знаходяться на різних машинах і передають дані через комп’ютерну мережу, проте інколи клієнт і сервер знаходяться на одній машині.

Сервер надає послуги клієнтам, що надсилають запити на отримання послуг. Сервер може надавати будь-які ресурси, від програм до обчислювальних потужностей і пристроїв зберігання даних. Як правило, сервер призначений для роботи в якості централізованої системи, що обробляє запити від багатьох клієнтів. Навантаження і вимоги до комп'ютера, на якому розгорнутий сервер, значно зростають. Навіть якщо цей комп'ютер не відрізняється від інших комп'ютерів в мережі, його називають сервером, оскільки він виконує таку роль. Відповідно, клієнтами, окрім програм, називають комп'ютери у мережі, що надсилають запити на доступ до ресурсу.

Клієнтські програми поділяються на два види. Товстий клієнт буде проводити необхідні логічні операції на боці клієнта, використовуючи сервер лише для зберігання і отримання даних. Тонкий клієнт, навпаки, відповідає лише за представлення, а вся логіка обробки даних зосереджена на сервері. Для веб-сайтів прикладом товстого клієнта може бути веб-сторінка з динамікою на боці клієнта, кодом JavaScript, що виконується у браузері. Тонким клієнтом, навпаки буде сторінка, що повністю генерується сервером, або взагалі статична веб-сторінка.

Трирівнева архітектура — це архітектурний стиль клієнт-серверних програм, у якому структура системи розділяється на три рівні: представницький рівень (інтерфейс користувача), рівень обробки інформації (або бізнес-логіки), та рівень доступу до даних.

Рівень представлення – це рівень, що відображає інтерфейс користувача, і з яким користувач безпосередньо взаємодіє. Часто окрім інтерфейсу цей рівень забезпечує найпростіші операції, як правило, пов'язані з введенням і виведенням: авторизацію, шифрування, перевірку значень, що вводяться. У ASP .NET Core цей рівень представлений веб-сторінками Razor Pages. Окрім них, до цього рівня також відносяться контроллери, моделі веб-сторінок, стилі CSS, скрипти JavaScript. У випадку веб-сайтів цей рівень

представлений браузером на комп’ютері клієнта і завантаженою веб-сторінкою з даними.

Рівень бізнес-логіки – це рівень, що відповідає за виконання операцій з даними, що визначені логікою програми. Компоненти цього рівня обробляють запити, та відправляють інформацію представницькому рівню.

Рис. 2.3. Схема трирівневої архітектури

Рівень доступу до даних повинен забезпечувати інтерфейс для рівня бізнес-логіки, який надає методи керування збереженими даними не створюючи залежностей від механізмів зберігання даних. Відділення рівня доступу до даних дозволяє заміняти спосіб зберігання даних, не вносячи зміни в шар бізнес-логіки. У випадку Entity Framework Core, цей рівень буде складатися з контексту бази даних, і класів репозиторіїв. Саме вони і дозволяють розробнику працювати з базою даних, використовуючи об’єкти .NET. Ці класи вже реалізовані у Entity Framework Core, тому програмісту не потрібно писати їх вручну.

Висновок до розділу 2

В цьому розділі пояснювальної записки було детально розглянуто поняття архітектури програмного забезпечення, розібрані найбільш розповсюджені архітектурні стилі, визначені їх переваги та недоліки. Кожен архітектурний стиль орієнтується на свою сферу застосування.

Також було розглянуто сучасні засоби розробки веб-сайтів і проведено огляд технологій, які використовуються для створення клієнт-серверних систем. При виборі технологій для розробки перевага надавалася в першу чергу надійним, швидким, легким в освоєнні технологіям, добре пристосованим до розробки веб-сайтів.

Каркасом системи буде платформа ASP .NET Core, для створення веб-сторінок використовуватиметься ASP .NET Core Razor Pages, системою авторизації та аутентифікації користувачів буде ASP .NET Core Identity. Всі ці технології є частиною однієї платформи і добре взаємодіють одна з одною. Для управління базою даних була вибрана система SQLite. Вона буде інтегрована в систему за допомогою ADO .NET Entity Framework Core. В якості середовища розробки обрано Microsoft Visual Studio 2022 Community Edition. Це універсальне і одне із найкраще адаптованих під вище зазначені платформи середовищ. Всі обрані технології розповсюджуються безкоштовно.

Головними вимогами до архітектури додатку були були швидкість та зручність розробки, легкість внесення змін. Тому для створення клієнт-серверної була обрана добре відома трирівнева архітектура. Для розробки серверної частини була обрана монолітна архітектура, оскільки це пришвидшить розробку і підвищить продуктивність програми.

РОЗДІЛ 3
ОПИС ПРОГРАМИ

3.1 Структура діалогу

Веб-сайт було розроблено з орієнтуванням на вимоги, встановлені в розділі 2. Для того, щоб наочно зобразити функціонал створеного веб-сайту, була підготована діаграма використання.

Інтерфейс веб-сайту зручний і зрозумілий. Завдяки ньому відвідувачі веб-сайту можуть переглядати інформацію про космічні апарати, що їх цікавлять. Адміністратори мають доступ до функціоналу, що дозволяє редагувати інформацію. Вони здійснюють вхід в систему завдяки системі аутентифікації та авторизації. Також їм доступна зміна пароля.

В якості клієнтської частини використовуються динамічні веб-сторінки, що відображаються в браузері користувача. Сторінки написані мовою програмування C# з використанням технології Razor Pages. За основу ця технологія бере створення розмітки мовою HTML, і дизайн на основі стилів CSS.

Навігація між основними сторінками веб-сайту здійснюється через кнопками на верхній панелі.

Коли відвідувач заходить на сайт, першою він бачить головну сторінку. На верхній панелі розміщений заголовок «Довідник космічних апаратів». При наисканні на нього користувач перейде на головну сторінку з будь-якої іншої сторінки сайту.

Сторінка «Супутники» показує список всіх запущених космічних апаратів. Невдалі запуски також показані в цьому списку. Перехід на сторінку здійснюється за допомогою кнопки «Супутники» на верхній панелі. При натисканні на назву космічного апарату в цій таблиці відкриється веб-сторінка, що містить всю наявну інформацію про космічний апарат. Ця сторінка зображена на рисунку 3.1.

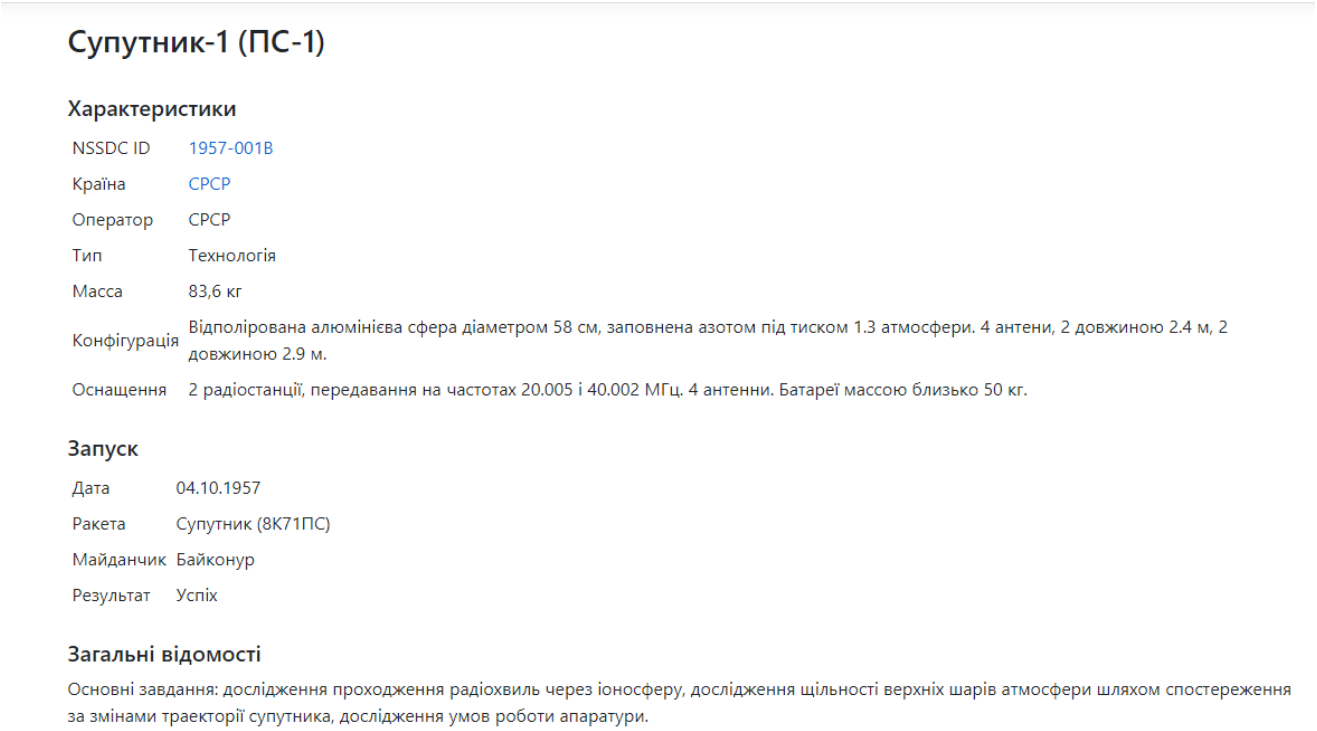


Рис. 3.1. Сторінка космічного апарату

Кнопка «Хронологія», що також знаходиться на верхній панелі, відправить відвідувача на сторінку, що показує список запусків космічних апаратів у хронологічному порядку. Згідно з вимогами, наявні також сторінки зі списками запусків у кожному році окремо. Перейти на них можна за посиланнями у верхній частині сторінки «Хронологія». На цих сторінках, окрім списку запусків, показана статистика надійності запусків, що дає можливість скласти уявлення про надійність космічної техніки певного періоду.

Сторінка «Країни» показує список країн, дані про космічні апарати яких є в системі. Також на сторінці показана статистика запусків країн. Сторінка покаже список країн, та їх статистику запусків. Сторінки кожної з країн окремо показують списки ракет-носіїв і космічних апаратів цієї країни. Для переходу на цю сторінку користувачу треба натиснути на посилання в назві країни в таблиці.

Сторінка «Ракети» відображає список ракет-носіїв. Для переходу на цю сторінку потрібно натиснути кнопку «Ракети» на верхній панелі. Сторінка зображена на рисунку 3.2.

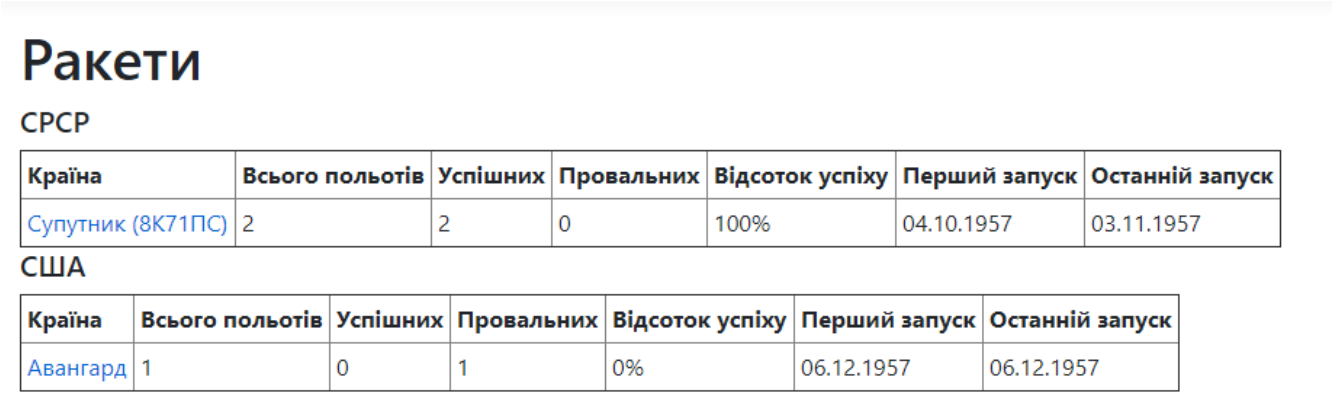


Рис. 3.2. Сторінка зі списком ракет

Відомості про кожну ракету-носій відображаються на окремій сторінці. Як і у випадку з космічними апаратами, відвідувач може перейти туди за посиланням в таблиці.

Для того, щоб отримати доступ до всього фунціоналу, адміністратор повинен виконати вхід в систему. На сторінку входу в систему можна перейти, натиснувши кнопку «Увійти» справа на верхній панелі. Щоб увійти в свій обліковий запис, адміністратор повинен ввести свій логін і пароль.

З виконанням входу в систему зовнішній вигляд веб-сторінок дещо змінюється. Зрозуміти, що вхід в систему виконано, можна по напису «Вітаємо,» разом з логіном адміністратора на верхній панелі. Кнопка «Увійти» змінюється на кнопку «Вийти». При натисканні цієї кнопки адміністратор вийде з системи.

Увійшовши в систему адміністратор отримує доступ до можливостей редагування інформації на сервері та управління обліковим записом. Для того, щоб змінити пароль від свого облікового запису, адміністратору потрібно натиснути кнопку «Змінити пароль». Сторінка «Змінити пароль»

попросить у адміністратора ввести існуючий пароль, а потім ввести і підтвердити новий.

Адміністратор може створити новий обліковий запис. Це потрібно для того, щоб мати змогу дати повноваження іншому адміністратору. Для цього йому треба натиснути на кнопку «Додати користувача», а на веб-сторінці, що відкриється, ввести логін і пароль для нового облікового запису.

Для редагування інформації про один з об’єктів адміністратору треба буде перейти на веб-сторінку для редагування. Для цього необхідно спершу перейти на сторінку, що відображає інформацію про об’єкт. Якщо відвідувач виконав вхід в систему, то посилання на сторінку для редагування даних буде доступне під заголовком сторінки. На цій сторінці присутні різні поля для заповнення, в залежності від об’єкта, інформацію про який редагують. Натиснувши кнопку «Змінити», адміністратор відправить оновлені дані в серверну базу даних.

Для додавання в базу відомостей про новий об’єкт адміністратор повинен перейти на сторінку зі списком відповідних об’єктів. Посилання на сторінку, що дозволяє додавати нові об’єкти, також буде доступне лише відвідувачу, що виконав вхід. На сторінці для додавання нових космічних апаратів є можливість пошуку за NSSDC ID. Знаючи NSSDC ID космічного апарату, можна застосувати систему пошуку та перекладу. При натисканні кнопки «Шукати», сервер спробує знайти інформацію про космічний апарат і надасть результати пошуку.

3.2 Структура рівня доступу до даних

Зв’язок між серверною частиною веб-сайту і базою даних відбувається за допомогою рівня доступу до даних. Цей рівень можна розділити на дві частини: класи моделей даних (класи-сутності) і класи, що виконують операції в базі. На рисунку 3.3 зображена діаграма класів для рівня доступу до даних.

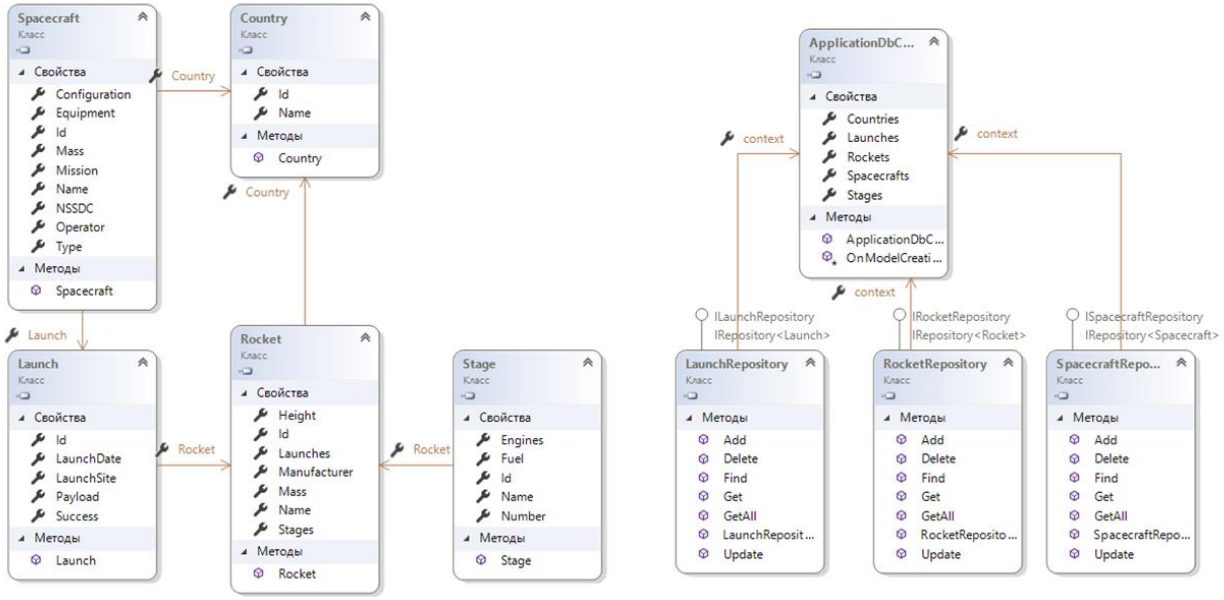


Рис. 3.3. Діаграма класів для рівня доступу до даних

Класи-сутності для об’єктно-реляційного відображення в ADO.NET Entity Framework строго типізовані та повністю сумісні як з системою типів, що використовується в системі управління базою даних, так і з загальною системою типів .NET Framework. Кожен клас сутності має унікальне поле-ключ, яка однозначно ідентифікує кожен екземпляр сутності.

Клас «Country» відображає країну. В класі наявні лише два поля: «Id» та «Name», що представляють ключ та назву країни відповідно. Для створеного веб-сайту інші характеристики країни не важливі. Клас представлений на рисунку 3.4.

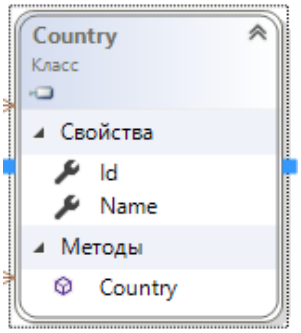


Рис. 3.4. Клас «Country»

Клас «Spacecraft» відображає космічний апарат. Клас представлений на рисунку 3.5.

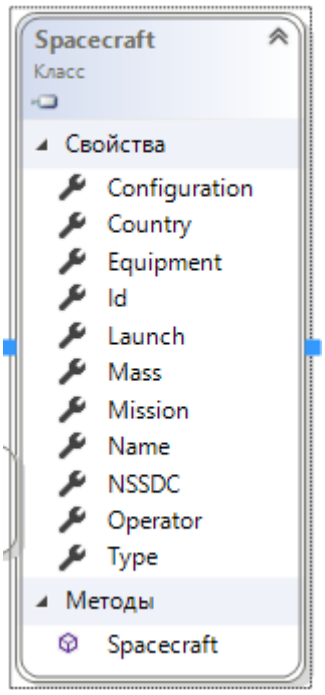


Рис. 3.5. Клас «Spacecraft»

Клас «Launch» відображає запуск. Ця сутність потрібна, щоб зв’язати класи «Spacecraft» та «Rocket». Окрім цього, сутність потрібна для зберігання даних про дату, місце і результат запуску. Клас представлений на рисунку 3.6.

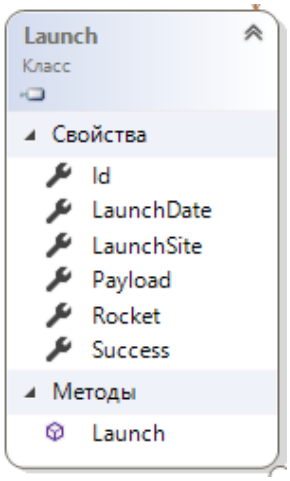


Рис. 3.6. Клас «Launch»

Клас «Rocket» відображає ракету-носій. Сутність зберігає інформацію про основні характеристики ракети-носія. Для полегшення доступу до даних в клас включений список запусків Launches. Завдяки платформі Entity Framework це ніяк не вплине на структуру бази даних. Клас представлений на рисунку 3.7.

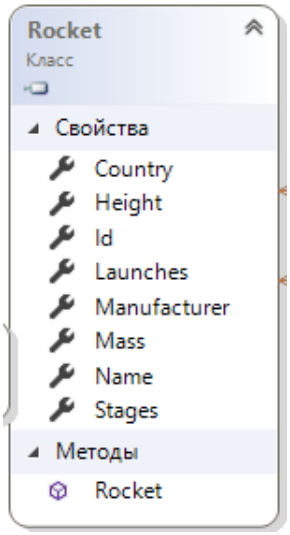


Рис. 3.7. Клас «Rocket»

Клас «Stage» відображає один із ступенів ракети-носія. Сутність використовується для зберігання додаткової інформації про ракету, особливо що стосується її ступенів. Клас представлений на рисунку 3.8.

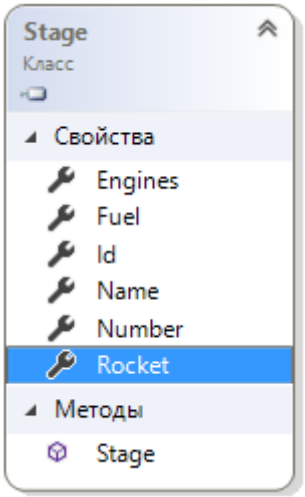


Рис. 3.8. Клас «Stage»

Контекст бази даних представлений класом «ApplicationDbContext» є класом контексту даних. Клас використовується Entity Framework Core для всієї роботи з базою даних. Клас комбінує в собі реалізацію паттернів «репозиторій» і «одиниця роботи». Тобто, контекст бази даних інкапсулює в собі контроль за підключеннями і ефективним виконанням запитів. Система аутентифікації ASP .NET Core Identity також потребує класу контексту бази даних для роботи.

Клас контексту повинен включати в себе поля типу «DbSet», які є об’єктно-орієнтованим відображенням таблиць в базі даних. Ці об’єкти надають зручний спосіб доступу до даних в таблицях, і дозволяють працювати з ними, як з колекціями в C#. Система аутентифікації Identity також потребує таблиці для збереження даних користувачів. Проте, після підключення системи в проект таблиці будуть створені автоматично. Використовувати їх можна через інтерфейс системи аутентифікації, тому не потрібно створювати для них поля в класі «ApplicationDbContext».

Клас представлений на рисунку 3.9.

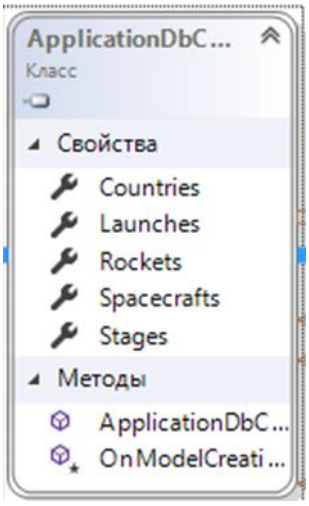


Рис. 3.9. Клас «ApplicationDbContext»

Класи-репозиторії «SpacecraftRepository», «RocketRepository і «LaunchRepository» також створені на основі EntityFramework, що надає базову реалізацію паттерну «Репозиторій». Ці класи, з одного боку, надають

можливість працювати з записами в базі даних методами об’єктно-орієнтованого програмування, а з іншого боку, виконують операції з базою даних, навіть не вимагаючи від програміста знання SQL. Класи представлені на рисунку 3.10.

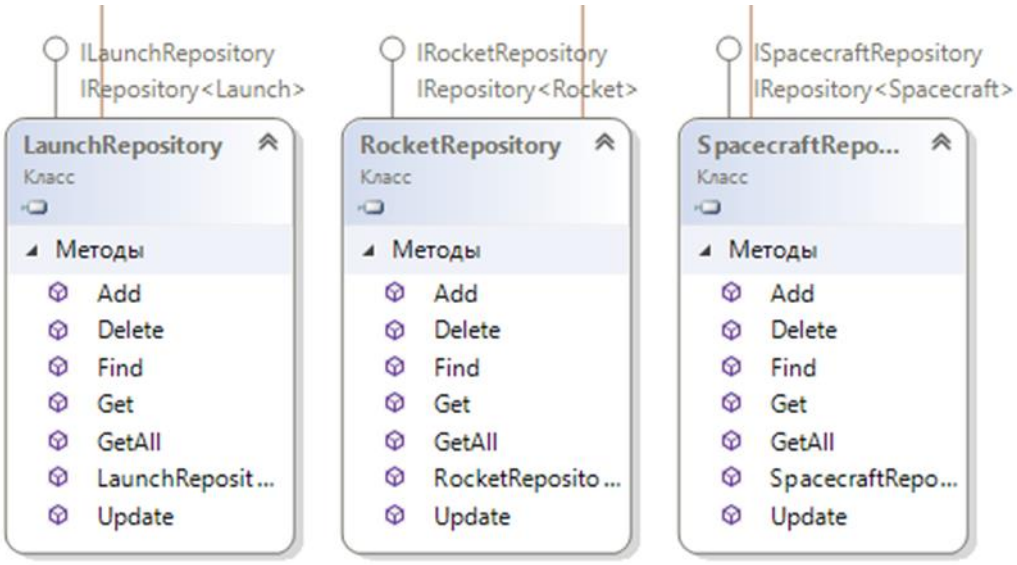


Рис. 3.10. Класи репозиторіїв

В основі кожного з цих класів лежить паттерн «Репозиторій». У випадку, якщо виникне необхідність внесення змін в код, ці класи доведеться змінювати. Щоб запобігти цьому, для кожного з класів був спроектований інтерфейс. Таким чином, можна буде легко замінити старий клас-репозиторій новим.

3.3 Структура шару бізнес-логіки

У трирівневій архітектурній моделі рівень бізнес логіки є проміжним рівнем між представницьким рівнем і рівнем доступу до даних. Основним завданням цього рівня є виконання логічних операцій над даними, прийняття і обробка запитів від представницького рівня, отримання інформації від рівня

доступу до даних, формування і відправка відповідей назад на представницький рівень.

Саме цей рівень містить сервіси та їх інтерфейси. Для деяких операцій необхідні моделі даних, що не мають власних таблиць в базі даних, і тому їх не можна включити в рівень доступу до даних. Тому ці моделі розміщені на рівні бізнес-логіки. На рисунку 3.11 зображена діаграма класів для рівня бізнес-логіки.

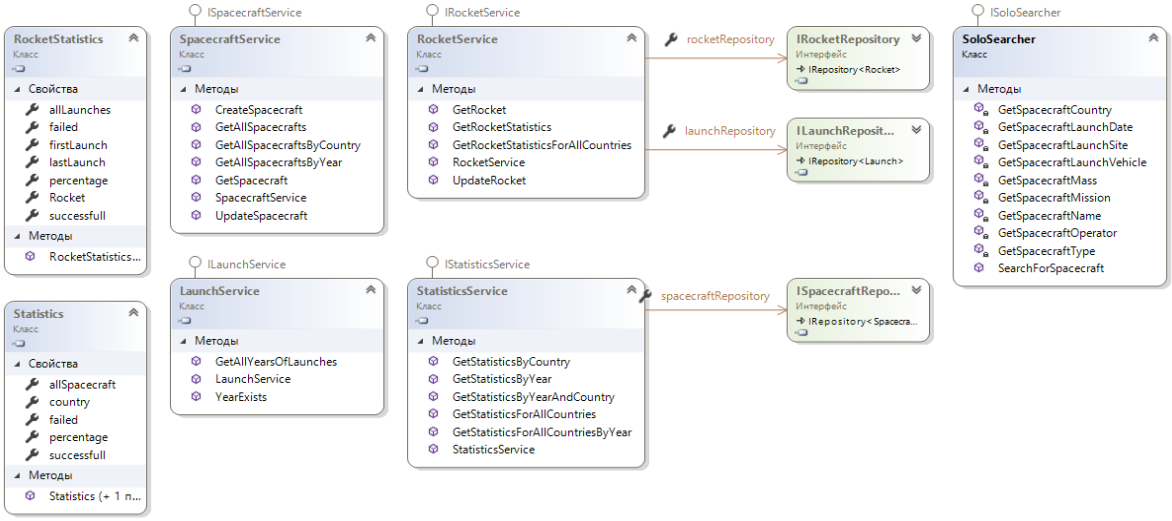


Рис. 3.11. Діаграма класів для рівня бізнес-логіки

Клас «SpacecraftService» – це клас-сервіс, тобто він включає в себе певну логіку і може використовуватися незалежно від інших класів. Цей клас додає, змінює, сортує, виконує операції з інформацією про космічні кораблі.

Клас представлений на рисунку 3.12.

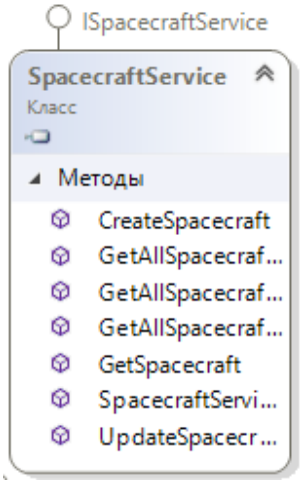


Рис. 3.12. Клас «SpacecraftService»

У класі «LaunchService» містяться методи, що відповідають за отримання даних про космічні запуски. Ці дані використовуються для побудови хронологічної таблиці космічних запусків.

Клас представлений на рисунку 3.13.

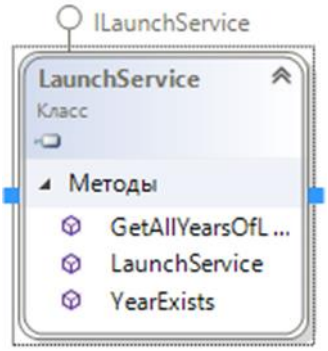


Рис. 3.13. Клас «LaunchService»

У класі «RocketService» містяться методи, що відповідають за операції з інформацією про ракети-носії. Клас необхідний для отримання або внесення змін в дані про ракету-носій. Також, в класі знаходяться методи, що обчислюють успішність запусків ракет-носіїв та надають статистику. Клас представлений на рисунку 3.14.

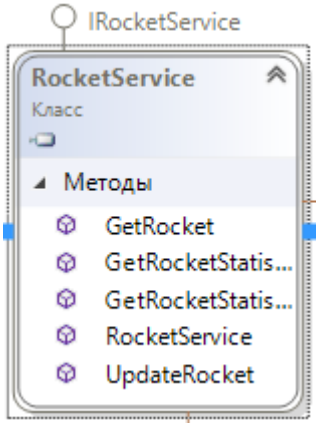


Рис. 3.14. Клас «RocketService»

У класі «StatisticsService» містяться методи, що відповідають за отримання всієї іншої статистики, що використовується в програмі. Клас дозволяє обчислювати статистику космічних запусків за один рік, для однієї країни, та статистику для однієї країни в певний рік.

Клас представлений на рисунку 3.15.

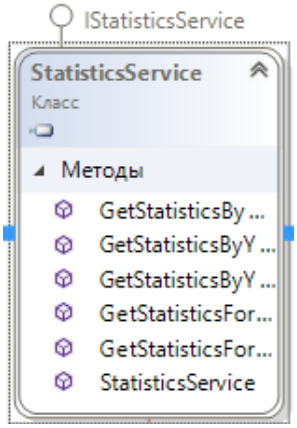


Рис. 3.15. Клас «StatisticsService»

В класі «StatisticsService» використовуються ще дві моделі для роботи зі статистикою. Це «Statistics», та «RocketStatistics». Перша модель представляє статистику космічних запусків однієї країни за певний період часу, а друга – статистику успішності запусків ракет-носія. Обчислення статистики може проводитися в конструкторах цих класів при наданні

вибірки запусків. Класи «Statistics», та «RocketStatistics» представлені на рисунку 3.16.

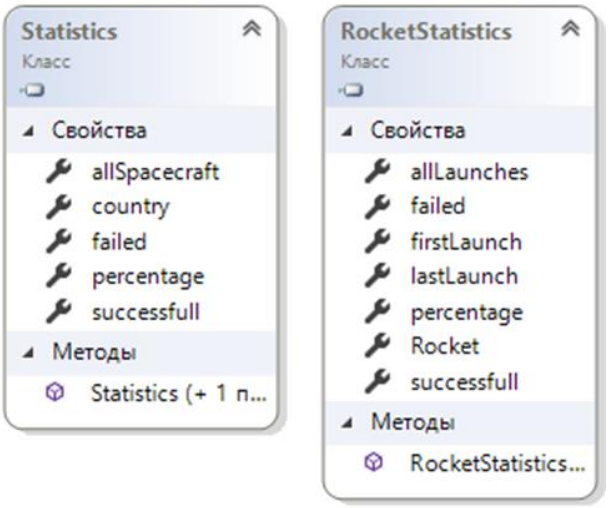


Рис. 3.16. Класи статистики

Інформацію про статистику не можна просто записати в базу даних, оскільки додавання нової інформації змусить оновлювати статистичні показники в багатьох таблицях. Можна отримувати статистику використовуючи запити SQL для її підрахунку, проте такий спосіб не буде зручним для програмістів, що не працюють з базами даних. Також обидва цих способи будуть збільшувати час роботи з базою даних.

Для пошуку та перекладу інформації про космічні апарати в програмі реалізований клас «SoloSearcher». Це клас-сервіс, в якому містяться методи для отримання з мережі різноманітних даних про космічні апарати.

Пошук заснований на NSSDC ID космічного апарату. Це міжнародний ідентифікатор, що складається з року запуску, трицифрового номеру запуску в році, і буквенного позначення об’єкта. Після отримання ідентифікатора на сайт архіву НАСА «NSSDCA Master Catalog», буде надісланий запит з цим ідентифікатором. У відповідь повинна надійти сторінка, що містить відомості про космічний апарат. Цей клас представлений на рисунку 3.17.

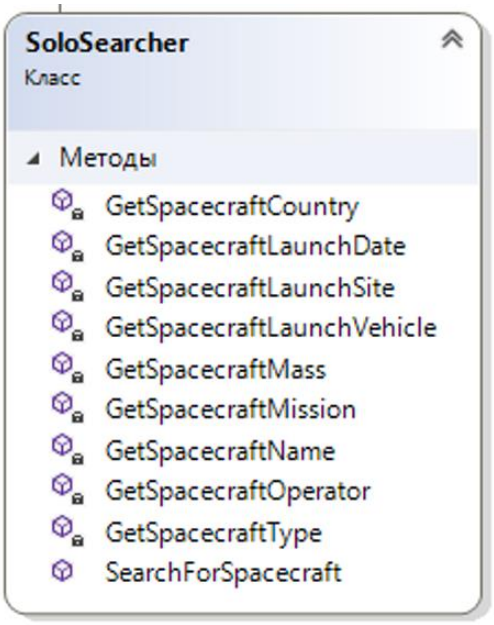


Рис. 3.17. Клас «SoloSearcher»

Якщо на будь-якому етапі в ході роботи сервісу виникне помилка, або якщо за вказаним ідентифікатором не вдасться знайти даних, в якості відповіді буде передано рядок «NoData». Таким чином веб-сайт не припинить роботу.

Пошук заснований на NSSDC ID космічного апарату. Це міжнародний ідентифікатор, що складається з року запуску, трицифрового номеру запуску в році, і буквенного позначення об’єкта. Після отримання ідентифікатора на сайт архіву НАСА «NSSDCA Master Catalog», буде надісланий запит з цим ідентифікатором. У відповідь повинна надійти сторінка, що містить відомості про космічний апарат.

Якщо на будь-якому етапі в ході роботи сервісу виникне помилка, або якщо за вказаним ідентифікатором не вдасться знайти даних, в якості відповіді буде передано рядок «NoData». Таким чином веб-сайт не припинить роботу.

Якщо пошук буде успішним, то буде отримано текст HTML-документу, в якому містяться відомості про космічний апарат. Для того, щоб виокремити корисну інформацію з документу, треба скористатися парсингом. Парсинг, або синтаксичний аналіз — це процес аналізу послідовності символів у

природній мові, мовах програмування або структурах даних. Для парсингу HTML-документів було вирішено скористатися бібліотекою «HTML Agility Pack». Для її підключення можна просто скористатися менеджером пакетів NuGet.

Клас «SoloSearcher» має різні методи для отримання інформації про різні характеристики космічного апарата. Характеристики можуть включати країну, дату та місце запуску, масу, місію, оператора і ракету-носій, на якій був проведений запуск. В результаті парсингу повинні бути отримані короткі відомості про знайдений космічний апарат англійською мовою.

Наступним етапом обробки даних повинен бути переклад тексту на українську мову. Переклад тексту адміністратором самостійно буде занадто довгим і незручним. Для уникнення цього в програму необхідно включити систему машинного перекладу.

Машинний переклад — це галузь математичної лінгвістики, яка використовує програмне забезпечення для перекладу тексту або мовлення з однієї природної мови на іншу. На сьогоднішній день найпопулярнішою системою машинного перекладу, без сумніву, є Google-перекладач. У Google-перекладача є API, завдяки якому можна відправляти запити і отримувати переклад. Для того, щоб мати змогу скористатися цим API в програмах на C#, потрібно підключити бібліотеку «Google.Cloud.Translation.V2». Для інших мов програмування існують свої механізми, але Google-перекладач може бути підключений майже в будь-який проект.

Після того, як текст буде перекладено, результат пошуку, парсингу та перекладу буде відправлений на клієнтський комп’ютер і адміністратор зможе його відредагувати.

Машинний переклад продовжує вдосконалюватися, проте сьогодні точність і граматика машинного перекладу залишають бажати кращого. Наприклад, одного разу Google-перекладач переклав з англійської на українську слово «USSR» як «США». Окрім цього, різні країни мають різні назви для одних і тих самих об’єктів. У багатьох англомовних джерелах

ракета-носій, що вивела на орбіту «Супутник-1», називається «Modified SS-6 Sapwood», тобто модифікована ракета Р-7. Адміністратор веб-сайту повинен провести остаточний огляд знайденої інформації та виправлення помилок перекладу і підтвердити внесення даних в базу.

3.4 Класи конфігурування програми

Для стартової конфігурації програми використовуються класи «Startup» і «Program».

Додаток на основі ASP.NET Core повинен містити клас «Startup». При запуску цей клас буде виконуватися першим. Назва «Startup» відповідає конвенції іменування ASP.NET Core. Проте, використовуючи метод «webBuilder.UseStartup<Startup>()» в класі «Program» можна використовувати свій клас в якості стартового. Під час запуску програми цей клас буде завантажений середовищем ASP.NET Core. Клас «Startup» обов’язково повинен включати метод «Configure» і опціонально метод «ConfigureServices».

Метод «Configure» — це місце, де можна налаштувати конвеєр обробки запитів за допомогою екземпляра IApplicationBuilder, який надається вбудованим контейнером IoC. ASP.NET Core надає різні компоненти для конвеєра обробки запитів. Цей метод дозволяє включати лише необхідні компоненти, що підвищує продуктивність. Також в методі є два необов’язкових параметри: IHostingEnvironment і ILoggerFactory. Це вбудовані в ASP.NET Core сервіси, що відповідають за отримання і використання даних про середовище та логування.

Метод «ConfigureServices» — це метод, в якому можна зареєструвати сервіси. Цей метод налаштує програму для того, щоб можна було використовувати вибрані сервіси. У цьому додатку такими сервісами є веб-сторінки ASP .NET Core Razor Pages, система аутентифікації та авторизації ASP .NET Core Identity. Також доданий сервіс, що перенаправить

користувача на сторінку входу в систему, якщо той не буде авторизований та спробує потрапити на сторінку, призначену лише для адміністратора. Також тут додається вбудований сервіс для ін’єкції залежностей. Після реєстрації інтерфейсу і реалізації в контейнері потрібно просто додати інтерфейс в конструктор класу, що буде його використовувати. При створенні контейнер ін’єкції залежностей автоматично надасть зареєстровану реалізацію.

3.5 Структура представницького рівня

Верхній рівень трирівневої архітектурної моделі — це рівень користувацький, або представницький рівень. Цей рівень представляє результати роботи інших рівнів користувачу, і саме з цим рівнем користувач може взаємодіяти через браузер.

Файли веб-сторінок програми розміщені на цьому рівні. Кожна сторінка Razor складається з файла .cshtml, що є сумішшю конструкцій HTML, CSS і C#, та файла .cshtml.cs, що написаний мовою C# і відповідає за поведінку сторінки. Таким чином сторінкою Razor можна керувати як будь-яким іншим об’єктом в об’єктно-орієнтованому програмуванні. На відміну від популярного паттерну Model–View–Controller, що розбиває преадставницький рівень на три елементи: модель для збереження даних, вигляд для взаємодії з користувачем та контроллер для обробки запитів, сторінка Razor містить весь контролюючий код в одному файлі. Тому від класів контроллерів можна відмовитися.

При створенні каркасу програми в Microsoft Visual Studio класи конфігурування створюються в тому ж самому проекті, що й макети веб-сторінок. Тому для зручності вони разом з макетами веб-сторінок виведені на діаграму класів представницького рівня. На рисунку 3.18 зображена діаграма класів для представницького рівня.

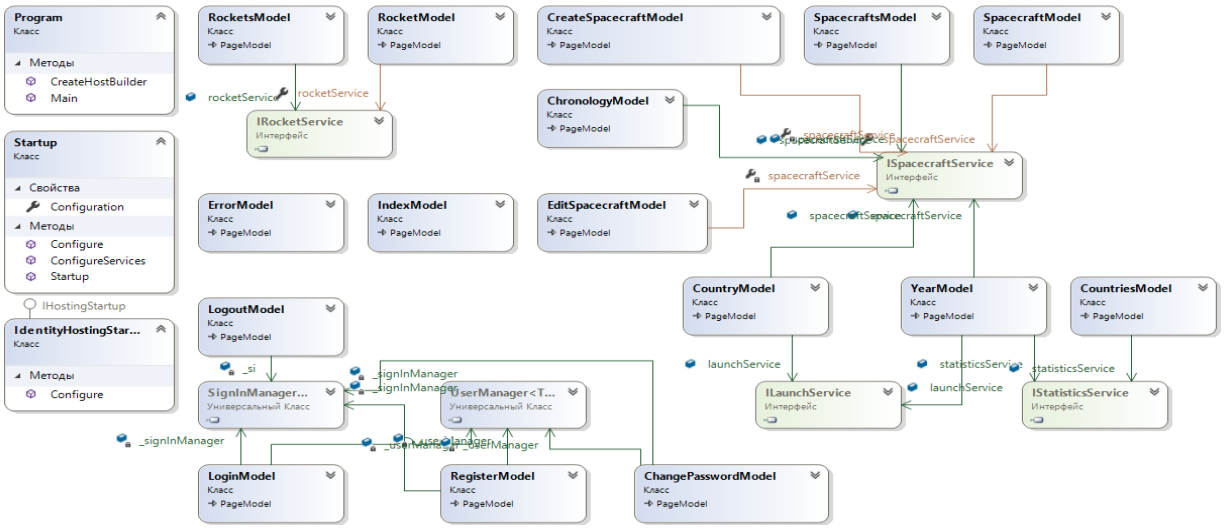


Рис. 3.18. Діаграма класів для представницького рівня

Сторінка «Index» це стартова сторінка веб-сайту. На ній немає ніякого особливого функціоналу. Це привід покращити її в майбутньому. Ця сторінка показана на рисунку 3.19.

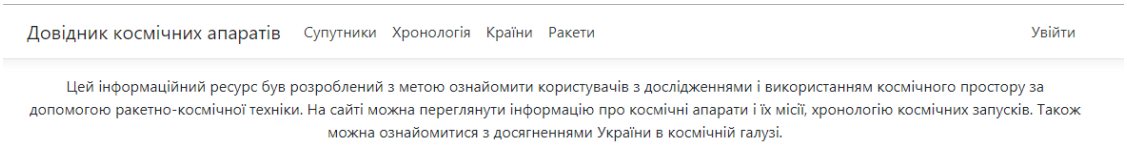


Рис. 3.19. Веб-сторінка Index

Призначення сторінки «Spacecrafts» — надання таблиці з космічними апаратами. Для цього використовується сервіс «SpacecraftService». Ця сторінка показана на рисунку 3.20.

Рис. 3.20. Веб-сторінка «Spacecrafts»

Призначення сторінки «Spacecraft» — надання наявної інформації про один вибраний космічний апарат. Для цього теж використовується сервіс «SpacecraftService». Ця сторінка показана на рисунку 3.21.

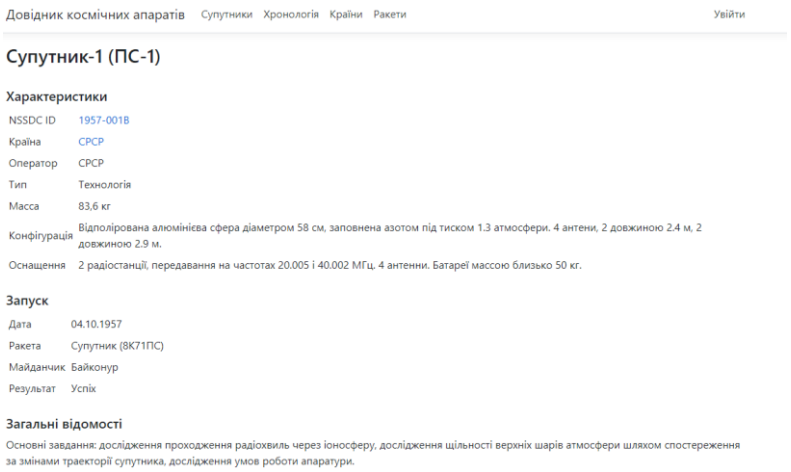


Рис. 3.21. Веб-сторінка «Spacecraft»

Ця сторінка надає наявну інформацію про космічний апарат. Відвідувач зможе дізнатися, з якою метою космічний апарат був запущений, яка маса апарату, якою країною він використовувався, яке обладнання було на борту. Також може бути включена додаткова інформація. На сторінці також вказано NSSDC ID космічного апарату. За прикріпленим до NSSDC ID посиланням зацікавлений користувач зможе перейти до, що відправляє користувача на сторінку сайту NSSDCA, де зможе отримати більше інформаціїї англійською мовою.

Призначення сторінки «Chronology» — представлення таблиці з списком хронологічних запусків космічних апаратів. Таблиця показує базову інформацію про космічний апарат. На сторінці є посилання на інші сторінки, що демонструють списки запусків космічних апаратів для кожного року. Ця сторінка показана на рисунку 3.22.

Рис. 3.22. Веб-сторінка Chronology

Призначення сторінки «Year» — представлення таблиці з списком космічних запусків певного року. Також на сторінці подається таблиця зі статистикою запусків цього року.

Для отримання всіх необхідних даних використовуються три сервіси. Сервіс «LaunchService» визначає, чи проводилися запуски в попередній і наступний рік. Сервіс «StatisticsService» відповідає за отримання статистики, що відображається на сторінці. Сервіс «SpacecraftService» отримує список запущених в цьому році космічних апаратів.

Клітинки в таблицях містять посилання на сторінки країн та космічних апаратів, що запускалися в цьому році. Ця сторінка показана на рисунку 3.23.

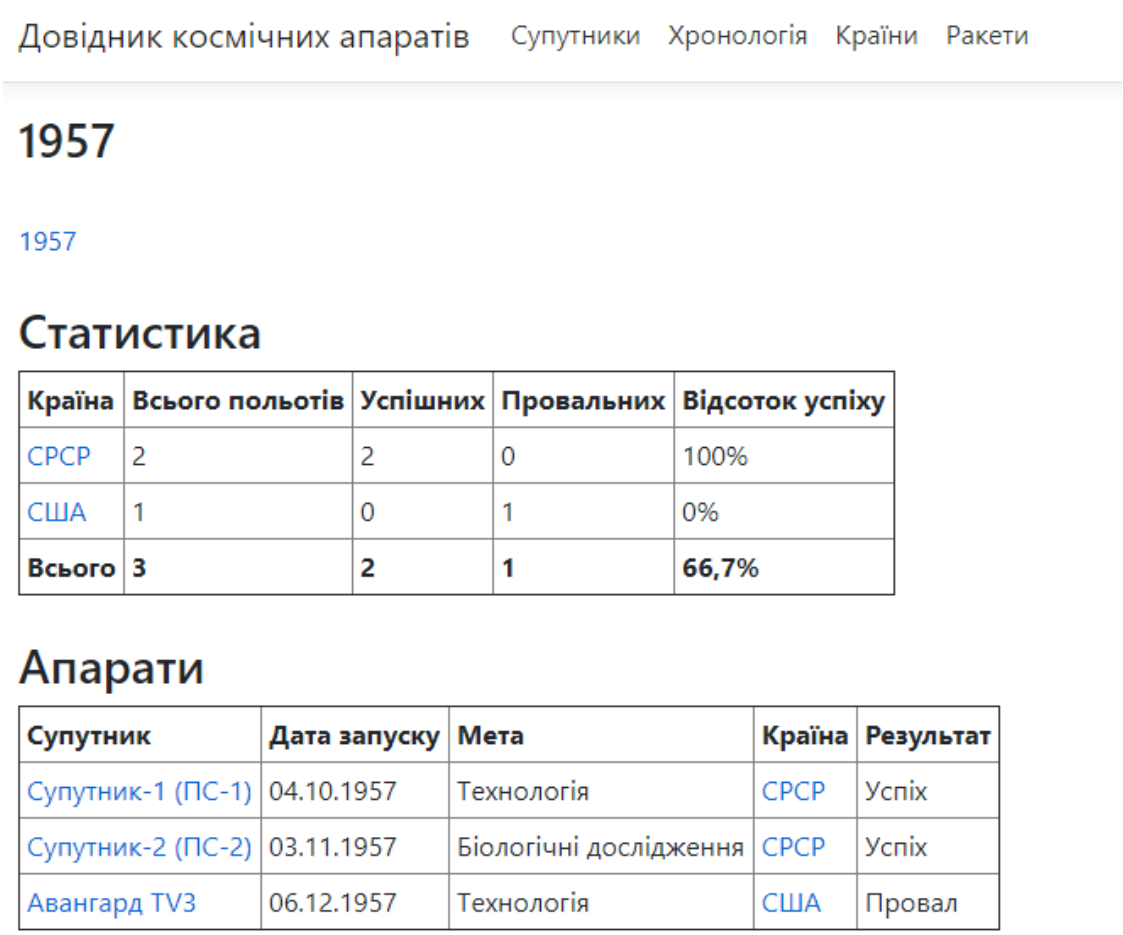


Рис. 3.23. Веб-сторінка «Year»

Призначення сторінки «Countries» — відображення інформації про кількість запусків різних країн та відсоток успішних запусків. Для цього використовується сервіс «StatisticsService».

На сторінці не показана детальна інформація. Завдання цієї сторінки в тому, щоб ознайомити відвідувача з країнами, що здійснюють запуски (або здійснювали раніше) космічних апаратів.

При натисканні на таблицю відвідувач зможе потрапити на сторінку зі списком космічних апаратів вказаної країни.

Ця сторінка показана на рисунку 3.24.

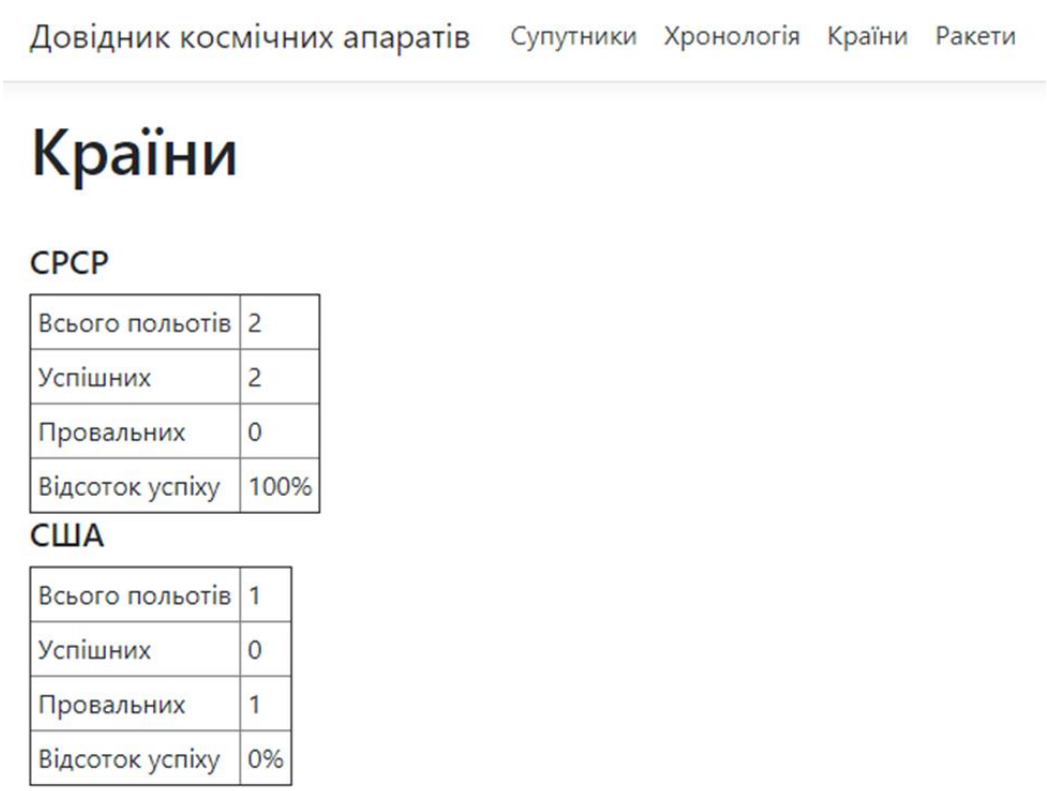


Рис. 3.24. Веб-сторінка «Countries»

Призначення сторінки «Country» — відображення таблиці з усіма космічними апаратами країни, та статистики успішності запусків цієї країни. Для цього використовуються сервіси «LaunchService» та «StatisticsService». Ця сторінка показана на рисунку 3.25.

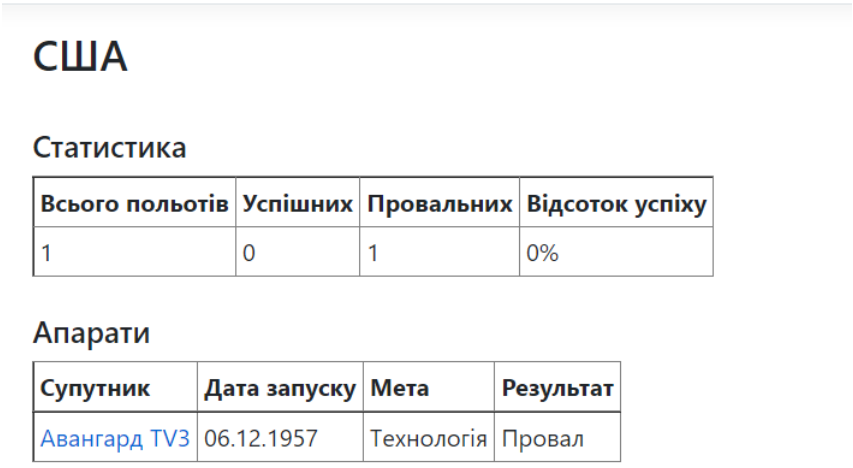


Рис. 3.25. Веб-сторінка «Country»

Призначення сторінки «Rockets» — відображення списку всіх ракет-носіїв і деякої статистики. У списку космічних апаратів було сортування за датою запуску, що зручно. Оскільки у різних моделей ракет-носіїв може бути багато запусків, зручніше за все буде відсортувати їх по країні створення.

Для роботи сторінці необхіден сервіс «RocketService». Ця сторінка показана на рисунку 3.26.

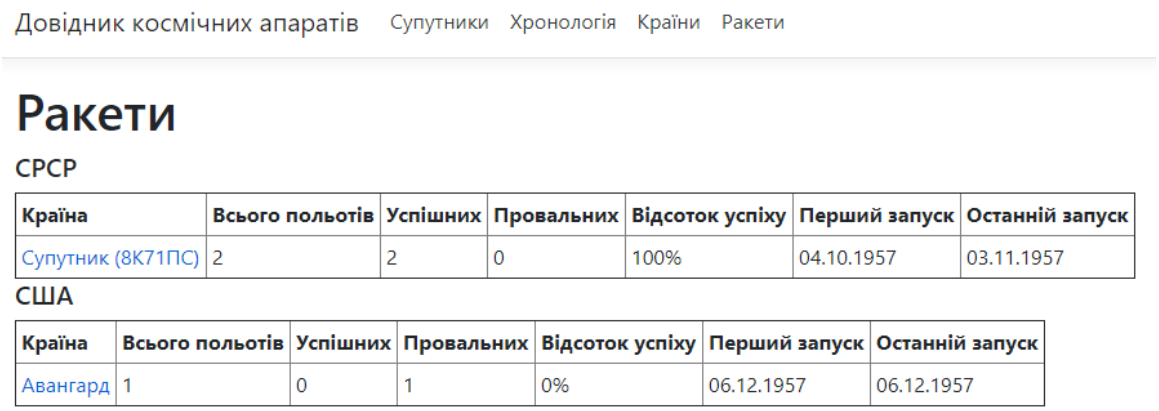


Рис. 3.26. Веб-сторінка «Rockets»

Призначення сторінки «Rocket» — надання відвідувачу даних про одну з ракет-носіїв. Для цього на сторінці використовуються декілька таблиць. У першій таблиці подаються загальні характеристики ракети: країна створення, виробник, маса та висота ракети. Друга таблиця містить перелік ступенів ракети. У третій таблиці подається статистична інформація про експлуатацію ракети-носія та дати першого і останнього запуску. Нарешті, в четвертій таблиці є список всіх запусків ракети-носія і перелік космічних апаратів, які були виведені на орбіту цією ракетою.

Сервіс «RocketService» надає всі методи, необхідні для заповнення цієї веб-сторінки. Сторінка показана на рисунку 3.27.

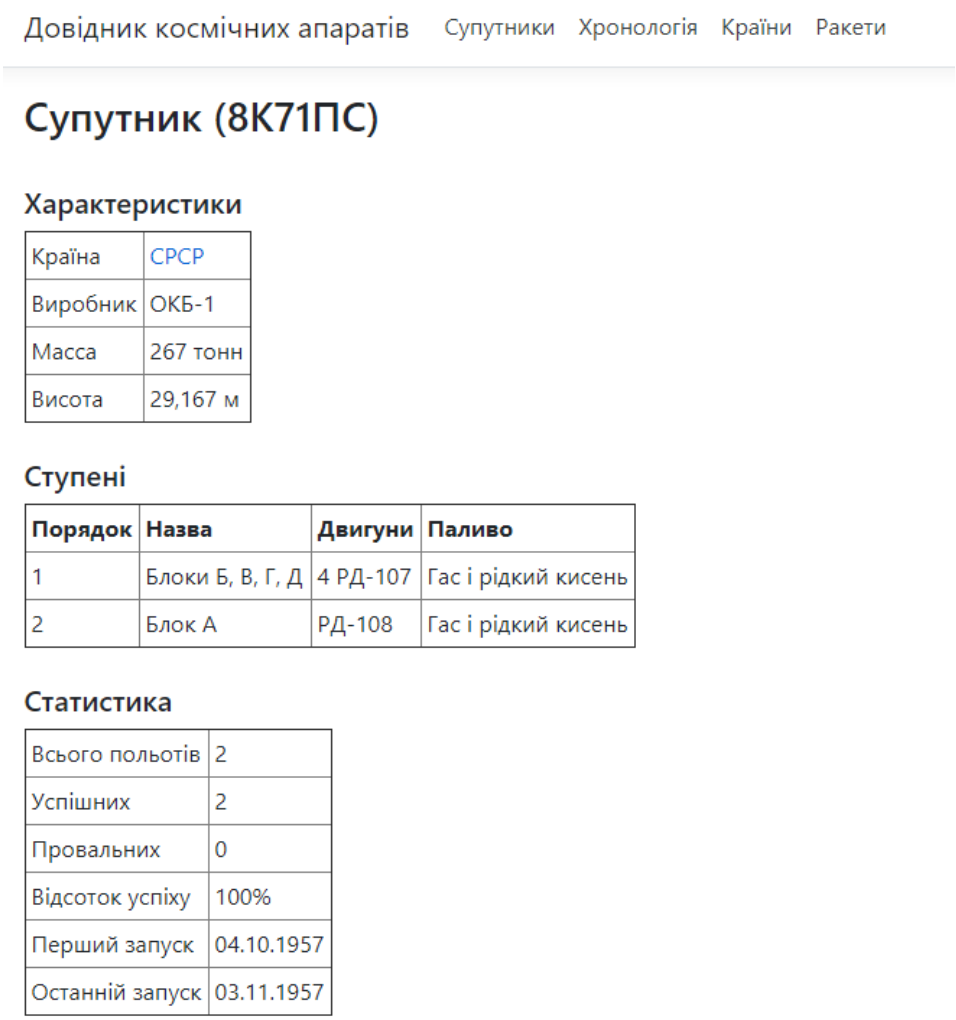


Рис. 3.27. Веб-сторінка «Rocket»

Скориставшись системою ASP .NET Core Identity, можна додати в проект попередньо створені сторінки для управління обліковим записом. Для цього в проекті буде створена окрема папка Identity/Pages/Account. В стандартні сторінки було необхідно внести невеликі зміни. Також система ASP .NET Core Identity створює менеджери для управління обліковими записами «UserManager» і «SignInManager». В цьому проекті вони використовуються в базовій конфігурації.

Призначення сторінки «Login» — надання адміністратору веб-сайту можливості увійти в систему. Попередня конфігурація проекту в класі «Startup» дозволить перенаправити користувача на цю сторінку, якщо він не був авторизований. Сторінка показана на рисунку 3.28.

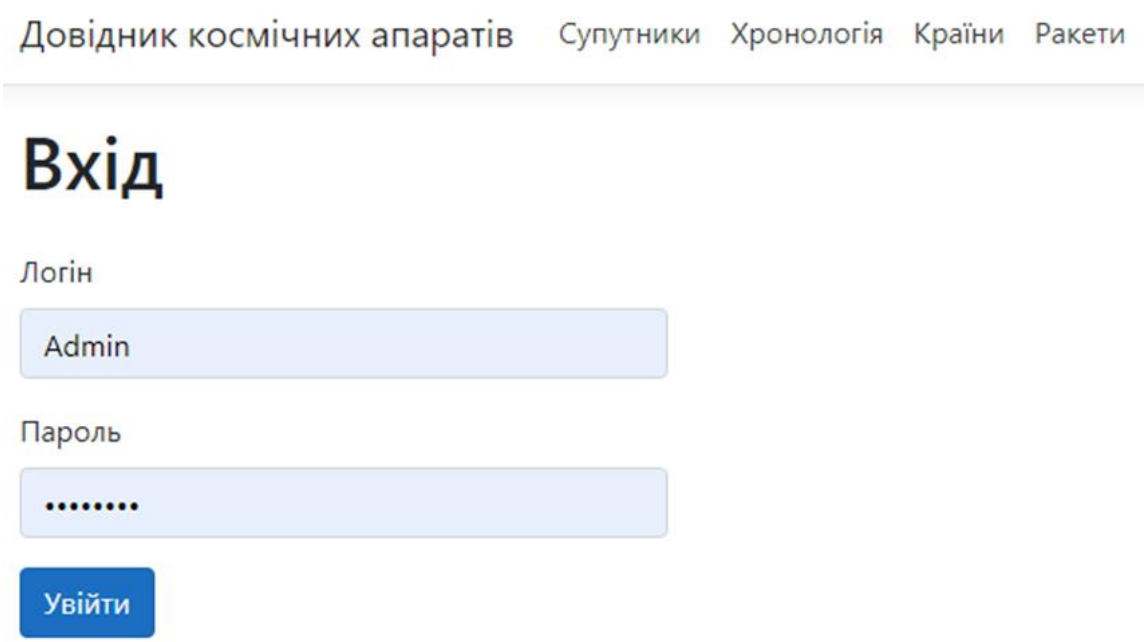


Рис. 3.28. Веб-сторінка «Login»

Призначення сторінки «ChangePassword» — дати можливість адміністратору змінити пароль від свого облікового запису. Сторінка, показана на рисунку 3.29.

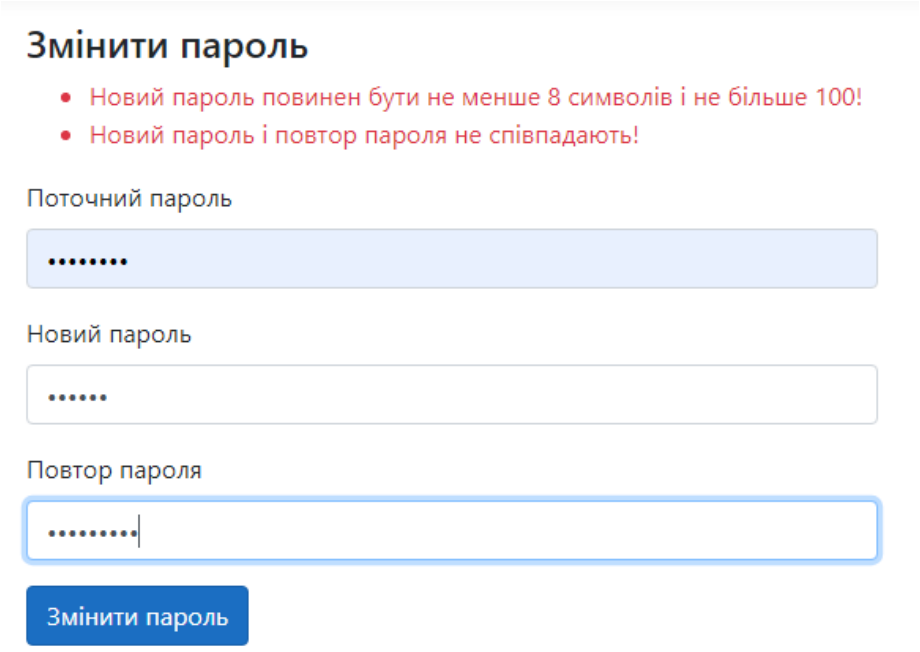


Рис. 3.29. Веб-сторінка «ChangePassword»

Пароль має відповідати мінімальним вимогам безпеки, бути не менше восьми символів, але не довше сотні. Вимоги до пароля перевіряються самою сторінкою в методах, що обробляють дані з форми.

Призначення сторінки «Register» — дати адміністратору можливість створити новий обліковий запис. Завдяки цьому декілька адміністраторів веб-сайту зможуть розділити між собою навантаження. В обов'язки адміністраторів входить створення нових сторінок, оновлення і редагування даних.

Сторінка «Register» показана на рисунку 3.30.

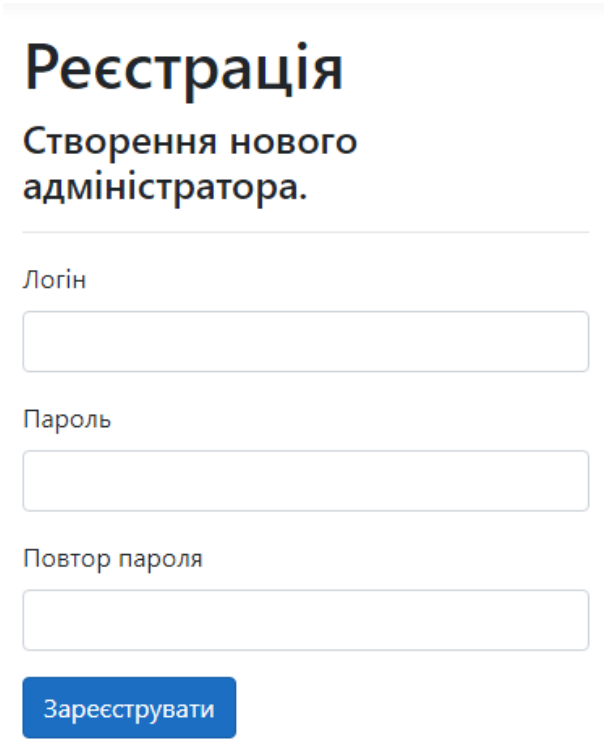


Рис. 3.30. Веб-сторінка «Register»

Призначення сторінки «EditSpacecraft» — зміна інформації про космічний апарат, що вже знаходиться в базі даних. Під час завантаження сторінки існуюча інформація про космічний апарат записується в текстові поля на сторінці. Адміністратор може змінювати вміст текстових полів. Коли адміністратор закінчить вносити зміни, він повинен натиснути кнопку

«Зберегти». Для збереження інформації використовується сервіс «SpacecraftService».

Сторінка показана на рисунку 3.31.

Рис. 3.31. Веб-сторінка «EditSpacecraft»

Призначення сторінки «CreateSpacecraft» — дати можливість створити запис про новий космічний апарат. Сторінка показана на рисунку 3.32.

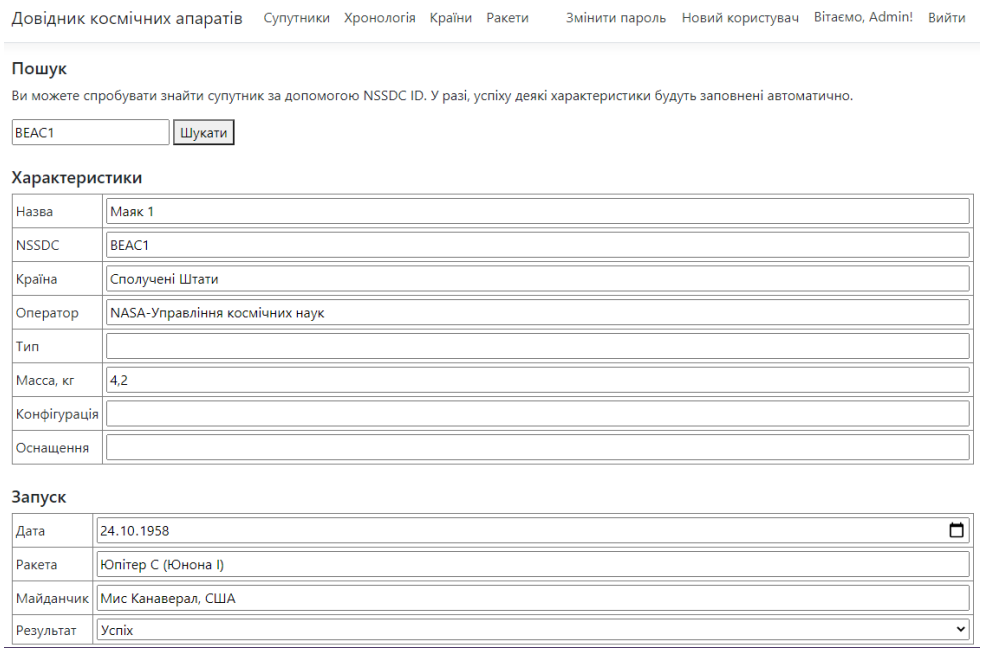


Рис. 3.32. Веб-сторінка «CreateSpacecraft»

Також сторінка дозволяє виконувати пошук інформації про космічний апарат за допомогою NSSDC ID. Сторінка містить текстові поля, призначені для різноманітної інформації про космічний апарат. Деякі з них можна залишити порожніми. Для збереження даних в базу потрібно натиснути на кнопку «Зберегти». Для пошуку відомостей про космічний апарат, необхідно ввести NSSDC ID в спеціально призначене для цього на сторінці текстове поле, і натиснути кнопку «Шукати». Тоді запит буде оброблено сервісом «SoloSearcher». Він передасть знайдену і перекладену інформацію класу сторінки, що відобразить її в браузері користувача. Після остаточного редагування інформацію можна зберігати в базу даних.

3.6 Тестування готової програми

Тестування програмного забезпечення — це перевірка властивостей та поведінки тестованого програмного забезпечення. Також тестування допомагає об’єктивно і незалежно оцінити програмне забезпечення, щоб дозволити компанії оцінити ризики впровадження програмного забезпечення. Окрім оцінки якості коду і перевірки правильності роботи програми в тестування також можуть входити перевірки на відповідність вимогам замовника. Чим більший і складніший програмний продукт, тим дорожчою буде його розробка, і більша імовірність помилок. Тому сьогодні на всіх проектах на тестування виділяється додатковий час і фінансові ресурси, щоб впевнитися, що закінчена програма відповідає вимогам замовника і не містить критичних помилок.

Основною метою тестування є виявлення помилок в роботі програмного забезпечення. Як правило, тестування не може гарантувати, що продукт функціонуватиме належним чином за будь-яких умов, але дозволяє перевірити роботу в тривіальних умовах і виявити найбільш массові помилки. Обсяг тестування залежить від ресурсів команди розробки і може включати огляд коду, виконання цього коду в різних середовищах і умовах.

Тестування часто починається після хоча б часткової готовності програмного коду, а інколи й до його написання (розробка через тестування). Тоді інформація, отримана під час тестування, може бути використана для коригування процесу розробки програмного забезпечення.

Деякі види тестування можуть самостійно визначити правильність роботи програми. Для інших необхідне порівняння результатів роботи програми зі специфікаціями, до яких належать контракти, минулі версії програми, вимоги замовників, стандарти, закони, та інші документи.

За всю історію розробки програмного забезпечення виникло чимало підходів до тестування, різних за масштабом, складністю та призначенням.

Димове тестування — перевірка базових можливостей програми для виявлення явних помилок, що виконується самим програмістом. Допомагає виявити явні помилки, наприклад, незапланованe завершення роботи програми.

Юніт-тестування, або модульне тестування — тестування на правильність роботи окремих модулів програми. В об’єктно-орієнтованих мовах програмування таким модулем найчастіше є метод. Юніт-тести це спосіб тестування коду на найнижчому рівні. Вони дуже прості, тому легко можуть бути автоматизовані.

Інтеграційне тестування — це тестування багатьох модулів системи разом. Метою інтеграційного тестування є перевірка правильності спільної роботи модулів системи, зв’язку між ними.

Системне тестування — тестування готової системи в цілому. На цьому етапі тестуються такі параметри як продуктивність, зручність інтерфейсу, повнота документації та сумісність з оточенням.

Для веб-сайту, створеного в ході написання цієї дипломної роботи, було проведено системне тестування, направлене на виявлення помилок в інтерфейсі користувача, та перевірку використання ресурсів комп’ютера.

Проект займає 52 мегабайта пам’яті на жорсткому диску, хоча варто пам’ятати, що розмір бази даних може дуже сильно збільшуватися при наповненні, а сам програмний продукт також може займати більше місця при додаванні нових функцій або виправленні помилок в ході підтримки. Під час роботи серверна частина вимагає до 500 мегабайт пам’яті. Хоча це може змінюватися в залежності від навантаження, ці характеристики сьогодні не є чимось надзвичайним навіть для персональних комп’ютерів. Таким чином, тестування підтвердило, що система працює без помилок, принаймні, видимих для користувача, і допомогло визначити необхідні ресурси сервера, на якому система буде запущена.

Висновок до розділу 3

В цьому розділі було описано створений під час роботи над проектом веб-сайт. За основу для планування архітектури були взяті вимоги до проекту, поставлені в розділі 2 та існуючі практики створення веб-сайтів.

Трирівнева архітектура є типовою для клієнт-серверних систем. В таких системах база даних, рівень доступу до даних і рівень бізнес-логіки знаходяться на потужному серверному комп’ютері, що дозволяє виконувати запити від багатьох клієнтських комп’ютерів. Користувацький рівень знаходиться на комп’ютері клієнта і призначений, як правило, не для самостійної обробки інформації, а для надсилання запитів на сервер і представлення відповідей користувачу.

В цьому розділі наданий детальний опис структури кожного з архітектурних рівнів веб-сайту. Також обгрунтовано використання і призначення кожної з використаних програмних платформ і технологій.

Розроблений інтерфейс користувача є простим та зрозумілим. В цьому розділі надано інструкцію для користування веб-сайтом. Всі веб-сторінки та елементи інтерфейсу описані, їх призначення розкрито.

РОЗДІЛ 4
РОЗРОБКА СТАРТАП-ПРОЕКТУ

4.1 Інформаційна карта проекту

Таблиця 4.1

Інформаційна карта проекту

1. Назва проекту	Веб-портал про космічні апарати і технології
2. Автори проекту	Кіпріянов Гліб
3. Коротка анотація	Мета проекту – розробка веб-порталу, присвяченого космічним технологіям і апаратам. Веб-портал надаватиме можливість користувачам Інтернету, які цікавляться дослідженням і освоєнням космосу, інформацію українською мовою. Для спрощення роботи адміністраторів веб-портал матиме можливість парсингу даних і перекладу на українську мову.
4. Термін реалізації проекту	6 місяців
5. Необхідні ресурси	Фінансові: – заробітна плата для задіяних спеціалістів: менеджера, бекенд-розробника, фронтенд-розробника і девопса; – витрати на обладнання та матеріали, необхідні для роботи: робоче приміщення, ноутбуки, комунальні послуги, кава, енергетики;

Таблиця 4.1 (продовження)

Інформаційна карта проекту

5. Необхідні ресурси	– витрати, пов'язані з веденням бізнесу: податки, юридичні витрати, збори, тощо. Інтелектуальні: В розробці стартапу використовуються технології з відкритим кодом, що розповсюджуються безкоштовно. Тому додаткові витрати на ліцензії не потрібні. Людські: необхідна команда досвідчених спеціалістів для управління невеликою компанією, або хоча б командою розробників: – менеджер з умінням управляти компанією та орієнтуватися у ринковій ситуації; – middle бекенд-розробник; – middle фронтенд-розробник; – девопс; – опціонально бізнес-аналітик, тестувальник, бухгалтер.
6. Опис проблеми, яку вирішує проект	Проект дасть можливість знайти інформацію про космічні технології, орбітальні і міжпланетні апарати українською мовою. Сьогодні така інформація часто неактуальна інколи суперечлива. Україномовних Інтернет-ресурсів на таку тему майже не існує.

Таблиця 4.1 (продовження)

Інформаційна карта проекту

7. Головні цілі та завдання проекту	Ціль – розробка веб-порталу, присвяченого космічним технологіям і апаратам. Веб-портал надаватиме можливість користувачам Інтернету, які цікавляться дослідженням і освоєнням космосу, інформацію українською мовою. Завдання: – створення веб-порталу для надання детальної, актуальної, своєчасної інформації про космічні технології, апарати, запуски, ракети-носії; – розробка привабливого та зручного інтерфейсу; – розгортання веб-порталу в мережі Інтернет.
8. Очікувані результати	Веб-портал, присвячений космонавтиці, що буде додавати і перекладати частину інформації. Це допоможе зняти частину навантаження з адміністраторів і прискорить роботу.

4.2 Технологічний аудит ідеї проекту

Перед початком розробки стартап-проекту варто провести аудит технологій. Правильний вибір технологій для розробки веб-сайтів дозволить

реалізувати більше корисних для клієнта функцій, а також спростить і пришвидшить розробку.

Обраний технологічний стек добре підходить для розробки веб-сайтів, добре задокументований, підтримується і регулярно оновлюється.

Таблиця 4.2 демонструє основні завдання проекту та доступність технології, яку можна використати для її реалізації.

Таблиця 4.2

Технологічна здійсненність завдань проекту

Завдання проекту	Технологія реалізації	Наявність технології	Доступність технології
Зберігання даних про космічні апарати та облікові записи адміністраторів, доступ до даних	Система управління базами даних SQLite, ASP .NET Core Entity Framework	Наявна	Розповсюджується безкоштовно
Серверна частина додатку, обробка запитів, обчислення статистики	Мова програмування C#, ASP .Net Core,	Наявна	Розповсюджується безкоштовно
Відображення інтерфейсу, відправка запитів на сервер	ASP .Net Razor Pages	Наявна	Розповсюджується безкоштовно
Обробка та переклад результатів пошуку	Бібліотека «HTML Agility Pack», Google-перекладач	Наявна	Розповсюджуються безкоштовно

Аутентифікація та авторизація	ASP .Net Core Identity	Наявна	Розповсюджується безкоштовно

Завдяки таблиці вище добре видно, що всі завдання проекту можуть бути реалізовані за допомогою існуючих програмних технологій. Обрані технології є надійними, швидкими, легким в освоєнні. Їх використання може полегшити працю розробників. Масштабованість обраних платформ дозволяє в майбутньому покращити проект і розширити функціонал. Всі вибрані технології розповсюджуються безкоштовно, що дозволить зменшити вартість розробки. Проте, в разі комерційного успіху проекту, можна буде підключити платні технології, якщо їх використання буде доцільнішим.

4.3 Аналіз ринкових можливостей запуску

Перед початком розробки та запуском в експлуатацію стартап-проекту варто провести аналіз ринкових можливостей. В аналіз рикових можливостей входить пошук потенційних клієнтів, можливостей та загроз. Також потрібно оцінити конкурентоспроможність стартапу в порівнянні з головними гравцями на ринку. У таблиці 4.3 показані основні характеристики потенційного ринку.

Таблиця 4.3

Загальна характеристика потенційного ринку проекту

№ п/п	Показники стану ринку	Характеристика
1.	Кількість головних гравців	1
2.	Загальний обсяг продаж	Як правило, некомерційні органцізації
3.	Динаміка ринку (якісна оцінка)	Зростає
4.	Наявність обмежень для входу	Наявність початкового капіталу, необхідного для розробки

5.	Специфічні вимоги до стандартизації та сертифікації	Формально відсутні

Для того, щоб визначити, чи користуватиметься проект попитом на ринку, необхідно здійснити пошук клієнтів, сформувати з них групи цільової аудиторії. Це дозволить визначити їхню зацікавленість в послугах, вимоги, та можливі зміни в продукт або в ринкову стратегію відповідно до зацікавленості користувачів. Загальні характеристики потенційних клієнтів надані в таблиці 4.4.

Таблиця 4.4

Характеристика потенційних клієнтів стартап-проекту

Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
Веб-портал для зберігання та отримання даних про космічні апарати та технології	Вищі навчальні заклади	Різний об’єм даних, зацікавленість в різній інформації, можливі додаткові вимоги	Мережева система для пошуку та зберігання інформації з забезпеченням доступу через мережу Інтернет. Простота та швидкість роботи. Зручний інтерфейс.

Відштовхуючись від основних завдань стартап-проекту, від характеристик ринкового середовища, та від аналізу цільових груп потенційних клієнтів можна визначити основні фактори загроз. Суть потенційних факторів загроз та можлива реакція компанії для їх виникнення показані в таблиці 4.5.

Таблиця 4.5

Фактори загроз

Фактор	Зміст загрози	Можлива реакція компанії
Конкуренція	Деякі організації вже мають схожі проекти, що можуть завадити виходу стартапу на ринок.	Запуск маркетингової програми, покращення якості послуг за рахунок тіснішої співпраці з клієнтом.
Відсутність інвестицій	Відсутність інвестицій не дасть змогу вкластися в розробку, розгортання та розвиток стартапу.	Укладання контракту на розробку з клієнтом або пошук додаткових джерел фінансування.
Ціна розробки та розгортання системи	Для створення системи необхідний певний обсяг фінансів, пропорційний складності розробки та розгортання.	Залучення додаткових коштів, або використання інших технологій з метою зменшення складності розробки і розгортання.

Окрім факторів загроз, аналіз ринкового середовища дає можливість оцінити фактори можливостей. Фактори можливостей – це потенційні сприятливі умови ринку, які можна використати для підвищення конкурентноспроможності стартап-проекту. Для того, щоб якнайкраще користуватися факторами можливостей, необхідно заздалегідь оцінити можливість виникнення сприятливої події та розробити стратегію дій

компанії. Фактори можливостей, що мають велике значення для цього стартапу, показані в таблиці 4.6.

Таблиця 4.6

Фактори можливостей

Фактор	Зміст можливості	Можлива реакція компанії
Залучення інвестицій	Отримання додаткових коштів на розробку.	Виділення додаткових фінансів на пришвидшення розробки або покращення якості системи.
Підписання контракту з замовником	Тісна співпраця з замовником для додаткового фінансування і більшої відповідності системи його вимогам.	Призначення бізнес-аналітика для зв’язку з замовником, залучення додаткових коштів для розробки додаткового функціоналу.
Інтеграція в існуючу компанію	Інтеграція стартап-проекту в програмну систему великої компанії.	Залучення ресурсів компанії для розвитку стартапу в тісній інтерграції з існуючими системами.

Аналіз ринкового середовища показав, що на цьому ринку майже відсутня конкуренція. Вибір систем, подібних тій, що розробляється в цьому стартап-проекті, дуже невеликий. На ринку є клієнти, які можливо, захочуть скористатися функціоналом такої системи. При виході на ринок слід враховувати як фактори можливостей, так і фактори загроз і мати відповідний план розвитку компанії.

Для того, щоб зацікавити клієнта, стартап-проект повинен бути конкурентноспроможним, тобто бути в очах користувача принаймні, не менш привабливим, ніж конкуренти. Одним зі способів підвищення конкурентноспроможності є технології, використані для розробки. Іншими факторами конкурентноспроможності є актуальність інформації, зручність використання системи, привабливість інтерфейсу, орієнтація на потреби користувача. Фактори конкурентноспроможності показані в таблиці 4.7.

Таблиця 4.7

Обгрунтування факторів конкурентоспроможності

Фактор конкурентоспроможності	Обгрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
Актуальність інформації	Для того, щоб мати цінність для користувача, інформація про космічні апарати та технології повинна залишатися актуальною та достовірною.
Зручність використання системи	Зручність полегшує роботу користувача з системою.
Привабливість інтерфейсу	Привабливість інтерфейсу також полегшує роботу користувача з системою. Зрозумілість інтерфейсу дозволяє користувачу одразу розуміти послідовність дій в роботі з системою, що підвищує ефективність роботи.
Орієнтація на потреби користувача	Орієнтація на потреби користувача дозволить краще їх задовільнити. Гнучким, масштабованим системам вдається впоратися з цим краще.

Таблиця 4.8 допоможе визначити сильніші та слабші сторони стартап-проекту в порівнянні з основними конкурентами на ринку.

Таблиця 4.8

Порівняльний аналіз сильних та слабких сторін стартап-проекту

Критерій конкурентно-спроможності	Оцінка 1-30	Оцінка конкурентів в порівнянні зі стартап-проектом						
		-3	-2	-1	0	+1	+2	+3
Актуальність інформації	28	+						
Зручність використання системи	28			+				
Привабливість інтерфейсу	25					+		
Орієнтація на потреби користувача	30	+						

Для завершення аналізу ринкових можливостей можна підбити підсумки за допомогою SWOT-таблиці. SWOT-аналіз (Strengths, Weaknesses, Opportunities, Threats) — це спосіб аналізу середовища (особливо часто бізнес-середовища), що з’явився в 1963 році. В ході SWOT-аналізу складається таблиця, що позначає внутрішні особливості проекту як сильні сторони (Strengths) та слабкі сторони (Weaknesses). Особливості зовнішнього середовища, що можуть допомогти в реалізації, називаються можливостями (Opportunities), а ті, що можуть завадити — загрозами (Threats). Для проведення SWOT-аналізу потрібно виписати виявлені сильні та слабкі сторони, можливості та загрози в таблицю. SWOT-таблиця стартап-проекту зображена за номером 4.9.

Таблиця 4.9

SWOT-табиця стартап-проекту

Сильні сторони: – актуальність інформації; – зручність використання; – привабливість інтерфейсу; – орієнтація на потреби користувача.	Слабкі сторони: - необхідність знайти фінансування для запуску; - невідомість серед користувачів.
Можливості: – залучення інвестицій; – підписання контракту з замовником; – інтеграція в існуючу компанію.	Загрози: – конкуренція з існуючими рішеннями.

4.4 Ринкова стратегія

Проведений аналіз ринку розкрив потреби користувачів, рівень конкурентів, потенційних клієнтів. Були визначені сильні та слабкі сторони стартап-проекту в порівнянні з конкурентами, можливості та загрози ринкового середовища. На основі аналізу ринку можна будувати ринкову стратегію компанії.

Потенційними клієнтами проекту вважаються вищі навчальні заклади, яким може бути потрібна клієнт-серверна система для пошуку, перекладу та зберігання інформації про космічні апарати та технології. В ринкових умовах, що склалися, доцільним буде використання однієї з трьох ринкових стратегій. Це самостійне створення системи та продаж доступу до неї, створення системи на замовлення вищого навчального закладу, або привернення уваги великої ІТ-компанії та створення системи під її

керівництвом. Складність різних етапів реалізації цих стратегій показана в таблиці 4.10.

Таблиця 4.10

Можливі ринкові стратегії стартап-проекту

Стратегія	Легкість виходу на ринок	Конкуренція	Фінансування	Можливості для розвитку
Самостійний вихід на ринок	Складна	Середня	За рахунок стартового капіталу	Високі
Розробка системи для замовника	Середня	Низька	За рахунок замовника	Низькі
Інтеграція з великою IT-компанією	Низька	Середня	За рахунок компанії	Середні

4.5 Команда стартапу

Для того, щоб визначити оптимальний склад команди для створення проекту, необхідно спочатку виділити основні етапи роботи та підібрати для них підходящих спеціалістів. Основні етапи роботи, та спеціалісти, що можуть її виконувати, вказані в таблиці 4.11.

Таблиця 4.11

Етапи розробки проекту та необхідні спеціалісти

Етап розробки	Спеціаліст
Дослідження ринкової ситуації та пошук інвесторів	Менеджер, бізнес-аналітик
Створення компанії та юридичне забезпечення	Менеджер
Узгодження технічного завдання	Менеджер, бізнес-аналітик

Створення, налаштування та підтримка середовища для розробки	Системний адміністратор

Таблиця 4.11

Етапи розробки проекту та необхідні спеціалісти

Розробка серверної частини веб-сайту	Бекенд-розробник
Розробка клієнтської частини веб-сайту та інтерфейсу користувача	Фронтенд-розробник
Тестування	Розробник, тестувальник
Розгортання та налаштування системи на сервері	Системний адміністратор
Супровід в ході експлуатації	Системний адміністратор, розробник
Взаємодія з клієнтами та аналіз результатів	Менеджер

З наведеної вище таблиці можна визначити, що необхідними спеціалістами для створення стартап-проекту будуть менеджер, бекенд-розробник, фронтенд-розробник і девопс.

Висновок до розділу 4

У цьому розділі був проведений технічний аудит проекту, виявлені сильні та слабкі сторони проекту, проаналізована ситуація на ринку. Аналіз ринкового середовища показав, що на цьому ринку майже відсутня конкуренція. На ринку є клієнти, які можливо, захочуть скористатися функціоналом такої системи. При виході на ринок слід враховувати як фактори можливостей, так і фактори загроз і мати відповідний план розвитку компанії. Проведений аналіз показав, що стартап може бути реалізований на ринку.

Була складена стратегія розвитку для компанії, що захоче взятися за реалізацію цього проекту. Був підібраний оптимальний склад команди розробників. Також були визначені основні можливості і загрози, які менеджменту компанії варто враховувати.

ВИСНОВКИ

Магістерська дисертація присвячена розробці веб-порталу про космічні апарати і технології.

Для визначення вимог до проекту був проведений детальний огляд предметної області. Були виділені головні особливості предметної області і проаналізовані можливості створення подібного проекту. В ході аналізу існуючих рішень виявилося, що більшість з них не відповідає вимогам, і не містить інформації українською мовою.

Для підготовки до розробки були розглянуті різні варіанти побудови архітектури програмних систем. Кожна архітектурна модель має свої переваги і недоліки, орієнтується на досягнення певних якостей програмної системи і тому використовуються для певного типу програм. Для створення веб-сайту добре підійде трирівнева архітектурна модель. Така модель забезпечить легке розгортання веб-сайту на сервері, знизить навантаження на комп’ютер клієнта і спростить розробку, логічно розділивши проект на частини.

Перед початком розробки проекту було проаналізовано існуючі технології для створення веб-сайтів. Оскільки веб-сайти є досить поширеним типом програм, для їх розробки було створено чимало зручних інструментів. При виборі технологій для розробки перевага надавалася в першу чергу надійним, швидким, легким в освоєнні технологіям, добре пристосованим до розробки веб-сайтів. Веб-сайт буде створений на платформі ASP .NET Core. Основною мовою програмування веб-застосунку буде C#. В якості середовища розробки обрано Microsoft Visual Studio 2022 Community Edition. Це універсальне і одне із найкраще адаптованих під вище зазначені платформи середовищ.

Проектом передбачена необхідність довготривалого зберігання даних. Тому для цього на сервверному боці необхідно розгорнути систему управління базою даних. Для цього була обрана SQLite — ситема, відома

своєю простотою використання, швидкодією, легкістю в розгортанні, портативністю.

Була створена автоматизована система пошуку та перекладу інформаціїї. Її використання дозволяє адміністратору швидко знайти і перекласти інформацію про космічний апарат, який необхідно внести в базу даних.

Інтерфейс користувача був розроблений з розрахунком на зручність та зрозумілість. Інтерфейс застосунку є простим, проте забезпечує всі необхідні для користування програмою функції.

Створений веб-сайт має великий потенціал для покращення. Архітектура спроектована з урахуванням цих можливостей. Можна розробити нову систему пошуку, або створити кращий інтерфейс користувача.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Вікіпедія. [Електронний ресурс]. — Режим доступу: https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B3%D0%BB%D0%B0%D0%B2%D0%BD%D0%B0%D1%8F_%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B8%D1%86%D0%B0 — Дата доступу: Жовтень 2022.

2. Gunter`s Space Page. [Електронний ресурс]. — Режим доступу: https://space.skyrocket.de/index.html — Дата доступу: Жовтень 2022.

3. The Spacecraft Encyclopedia. [Електронний ресурс]. — Режим доступу: http://claudelafleur.qc.ca/Spacecrafts-index.html — Дата доступу: Жовтень 2022.

4. NSSDCA Master Catalog. [Електронний ресурс]. — Режим доступу: https://nssdc.gsfc.nasa.gov/nmc/ — Дата доступу: Жовтень 2022.

5. Керівництво з програмування на C#. [Електронний ресурс]. — Режим доступу: https://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/ — Дата доступу: Жовтень 2022.

6. Stack Overflow. [Електронний ресурс]. — Режим доступу: https://stackoverflow.com/ — Дата доступу: Жовтень 2022.

7. Роб Майлс. The C# Programming Yellow Book [Текст] / Роб Майлс. — 2019.

8. Джеффрі Ріхтер. CLR via C#. Програмування на платформі Microsoft .NET Framework 4.5 на мові C#. 4-е вид [Текст] / Джеффрі Ріхтер. — 2019.

9. Керівництво з програмування на ASP.NET Core. [Електронний ресурс]. — Режим доступу: https://metanit.com/sharp/aspnet6/ — Дата доступу: Жовтень 2022.

10. Керівництво з програмування на C#. [Електронний ресурс]. — Режим доступу: https://metanit.com/sharp/tutorial/1.1.php — Дата доступу: Жовтень 2022.

11. Керівництво з використання Entity Framework Core Identity. [Електронний ресурс]. — Режим доступу: https://metanit.com/sharp/aspnet6/13.1.php — Дата доступу: Листопад 2022.

12. Керівництво з налаштування Entity Framework Core Identity. [Електронний ресурс]. — Режим доступу: https://docs.microsoft.com/ru-ru/aspnet/core/security/authentication/identity-configuration?view=aspnetcore-5.0 — Дата доступу: Листопад 2022.

13. ASP .NET Forums. [Електронний ресурс]. — Режим доступу: https://forums.asp.net/ — Дата доступу: Листопад 2022.

14. Список космічних запусків в 2022 році - Вікіпедія. [Електронний ресурс]. — Режим доступу: https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_%D0%BA%D0%BE%D1%81%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D1%85_%D0%B7%D0%B0%D0%BF%D1%83%D1%81%D0%BA%D0%BE%D0%B2_%D0%B2_2022_%D0%B3%D0%BE%D0%B4%D1%83 — Дата доступу: Жовтень 2022.

15. Платформа Google Cloud. [Електронний ресурс]. — Режим доступу: https://cloud.google.com/ — Дата доступу: Листопад 2022.

16. Інструкція для підключення до API Гугл-перекладача. [Електронний ресурс]. — Режим доступу: https://cloud.google.com/translate/docs/setup — Дата доступу: Листопад 2022.

17. Html Agility Pack — удобный .NET парсер HTML / Хабр. [Електронний ресурс]. — Режим доступу: https://habr.com/ru/post/112325/ — Дата доступу: Листопад 2022.

18. Дослідження космосу - Вікіпедія. [Електронний ресурс]. — Режим доступу: https://uk.wikipedia.org/wiki/%D0%94%D0%BE%D1%81%D0%BB%D1%96%D0%B4%D0%B6%D0%B5%D0%BD%D0%BD%D1%8F_%D0%BA%D0%BE%D1%81%D0%BC%D0%BE%D1%81%D1%83 — Дата доступу: Жовтень 2022.

19. SpaceX - Вікіпедія. [Електронний ресурс]. — Режим доступу: https://uk.wikipedia.org/wiki/SpaceX — Дата доступу: Жовтень 2022.

20. Emirates Mars Mission - Вікіпедія. [Електронний ресурс]. — Режим доступу: https://uk.wikipedia.org/wiki/Emirates_Mars_Mission — Дата доступу: Жовтень 2022.

21. Китайська космічна програма - Вікіпедія. [Електронний ресурс]. — Режим доступу: https://uk.wikipedia.org/wiki/%D0%9A%D0%B8%D1%82%D0%B0%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BA%D0%BE%D1%81%D0%BC%D1%96%D1%87%D0%BD%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B0 — Дата доступу: Жовтень 2022.

22. Артеміда (космічна програма) - Вікіпедія. [Електронний ресурс]. — Режим доступу: https://uk.wikipedia.org/wiki/%D0%90%D1%80%D1%82%D0%B5%D0%BC%D1%96%D0%B4%D0%B0_(%D0%BA%D0%BE%D1%81%D0%BC%D1%96%D1%87%D0%BD%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B0) — Дата доступу: Жовтень 2022.

23. Common Language Runtime - Вікіпедія. [Електронний ресурс]. — Режим доступу: https://uk.wikipedia.org/wiki/Common_Language_Runtime — Дата доступу: Листопад 2022.

24. Visual Studio — Основной инструмент разработчика на платформе .NET / Хабр. [Електронний ресурс]. — Режим доступу: https://habr.com/ru/hub/vs/ — Дата доступу: Листопад 2022.

25. Entity Framework - Вікіпедія. [Електронний ресурс]. — Режим доступу: https://uk.wikipedia.org/wiki/Entity_Framework — Дата доступу: Листопад 2022.

26. SQLite - Вікіпедія. [Електронний ресурс]. — Режим доступу: https://uk.wikipedia.org/wiki/SQLite — Дата доступу: Листопад 2022.

27. SQLite — замечательная встраиваемая БД (часть 1) / Хабр. [Електронний ресурс]. — Режим доступу: https://habr.com/ru/post/149356/ — Дата доступу: Листопад 2022.

28. Features of SQLite. [Електронний ресурс]. — Режим доступу: https://www.sqlite.org/features.html — Дата доступу: Листопад 2022.

29. Recommended Formats Statement - Datasets. [Електронний ресурс]. — Режим доступу: https://www.loc.gov/preservation/resources/rfs/data.html — Дата доступу: Листопад 2022.

30. Entity Framework – DbContext. [Електронний ресурс]. — Режим доступу: https://www.c-sharpcorner.com/article/entity-framework-dbcontext/ — Дата доступу: Листопад 2022.

31. Архітектура програмного забезпечення - Вікіпедія. [Електронний ресурс]. — Режим доступу: https://uk.wikipedia.org/wiki/%D0%90%D1%80%D1%85%D1%96%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F — Дата доступу: Листопад 2022.

32. Сервісно-орієнтована архітектура - Вікіпедія. [Електронний ресурс]. — Режим доступу: https://uk.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D1%96%D1%81%D0%BD%D0%BE-%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B0_%D0%B0%D1%80%D1%85%D1%96%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0 — Дата доступу: Листопад 2022.

33. Просто о микросервисах / Хабр. [Електронний ресурс]. — Режим доступу: https://habr.com/ru/company/raiffeisenbank/blog/346380/ — Дата доступу: Листопад 2022.

34. Мікорсервіси - Вікіпедія. [Електронний ресурс]. — Режим доступу: https://uk.wikipedia.org/wiki/%D0%9C%D1%96%D0%BA%D1%80%D0%BE%D1%81%D0%B5%D1%80%D0%B2%D1%96%D1%81%D0%B8 — Дата доступу: Листопад 2022.

35. Лучшая архитектура для MVP: монолит, SOA, микросервисы или бессерверная?. Часть 1 / Хабр. [Електронний ресурс]. — Режим доступу: https://habr.com/ru/company/otus/blog/476024/ — Дата доступу: Листопад 2022.

36. Многоуровневая архитектура в ASP .NET MVC 5. [Електронний ресурс]. — Режим доступу: https://metanit.com/sharp/mvc5/23.5.php — Дата доступу: Листопад 2022.

37. Software Architecture and Design Introduction. [Електронний ресурс]. — Режим доступу: https://www.tutorialspoint.com/software_architecture_design/introduction.htm — Дата доступу: Листопад 2022.

38. Processing static web pages. [Електронний ресурс]. — Режим доступу: http://etutorials.org/Macromedia/Dream+Weaver+Online+Help/Getting+Started+with+Dreamweaver/Understanding+Web+Applications/How+a+web+application+works/Processing+static+web+pages/ — Дата доступу: Листопад 2022.

39. UML — диаграмма вариантов использования (use case diagram) / Хабр. [Електронний ресурс]. — Режим доступу: https://habr.com/ru/post/47940/ — Дата доступу: Листопад 2022.

40. Клієнт-серверна архітектура - Вікіпедія. [Електронний ресурс]. — Режим доступу: https://uk.wikipedia.org/wiki/%D0%9A%D0%BB%D1%96%D1%94%D0%BD%D1%82-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80%D0%BD%D0%B0_

%D0%B0%D1%80%D1%85%D1%96%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0 — Дата доступу: Листопад 2022.

ДОДАТОК А

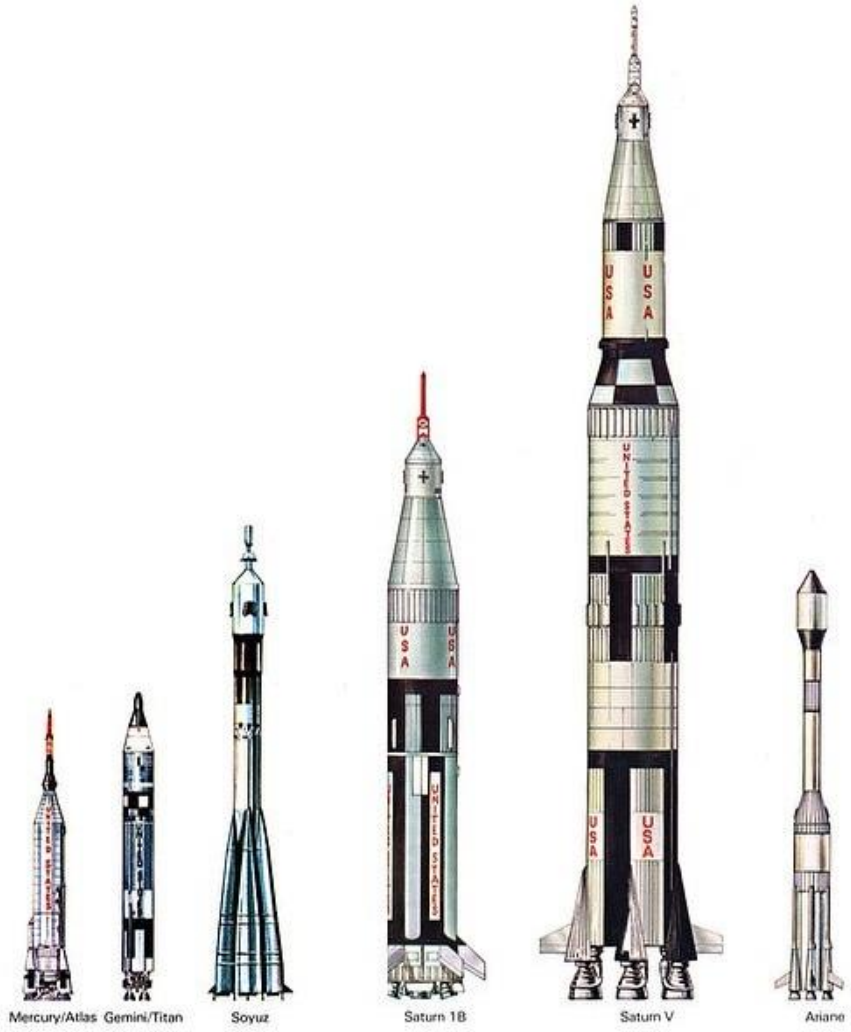
Короткий огляд історії досліджень космосу

У той час як спостереження за об’єктами в космосі, відомі як астрономія, виникли ще до письменної історії, саме розробка великих і відносно ефективних ракет у середині двадцятого століття дозволила фізичному дослідженню космосу стати реальністю.

Першою у світі масштабною експериментальною ракетною програмою була Opel-RAK під керівництвом Фріца фон Опеля та Макса Вальєра наприкінці 1920-х років, що призвело до створення перших пілотованих ракетних автомобілів і ракетопланів. Програма Opel-RAK і видовищні демонстрації наземних і повітряних транспортних засобів привертали великі натовпи, а також викликали глобальний ажіотаж громадськості, так званий «Rocket Rumble» і мали великий вплив на першопроходців космічних польотів: Корольова, Глушко, Вернера фон Брауна. Ракетні програми Радянського Союзу привели до появи реактивних снарядів та прототипів ракетних літаків. Першим рукотворним об’єктом, який досяг відкритого космосу, стала німецька ракета V-2. Під час випробувального запуску 20 червня 1944 року ракета перетнула лінію Кармана і піднялася на висоту 176 кілометрів. Після війни німецькі ракетні технології і спеціалісти стали основою для розвитку військових і космічних ракетних програм Радянського Союзу та Сполучених Штатів Америки.

«Космічна гонка» між Радянським Союзом і Сполученими Штатами стала рушійною силою для досягнень раннього періоду досліджень космосу. Запуски космічних апаратів вважалися необхідними для демонстрації прогресу в гонці ракетних і ядерних озброєнь та інших військових технологій, підтримки технологічного та міжнародного престижу держави. За межі цього періоду часто беруть запуск першого рукотворного об’єкта, який вийшов на орбіту Землі, радянського «ПС-1» 4 жовтня 1957 року, і

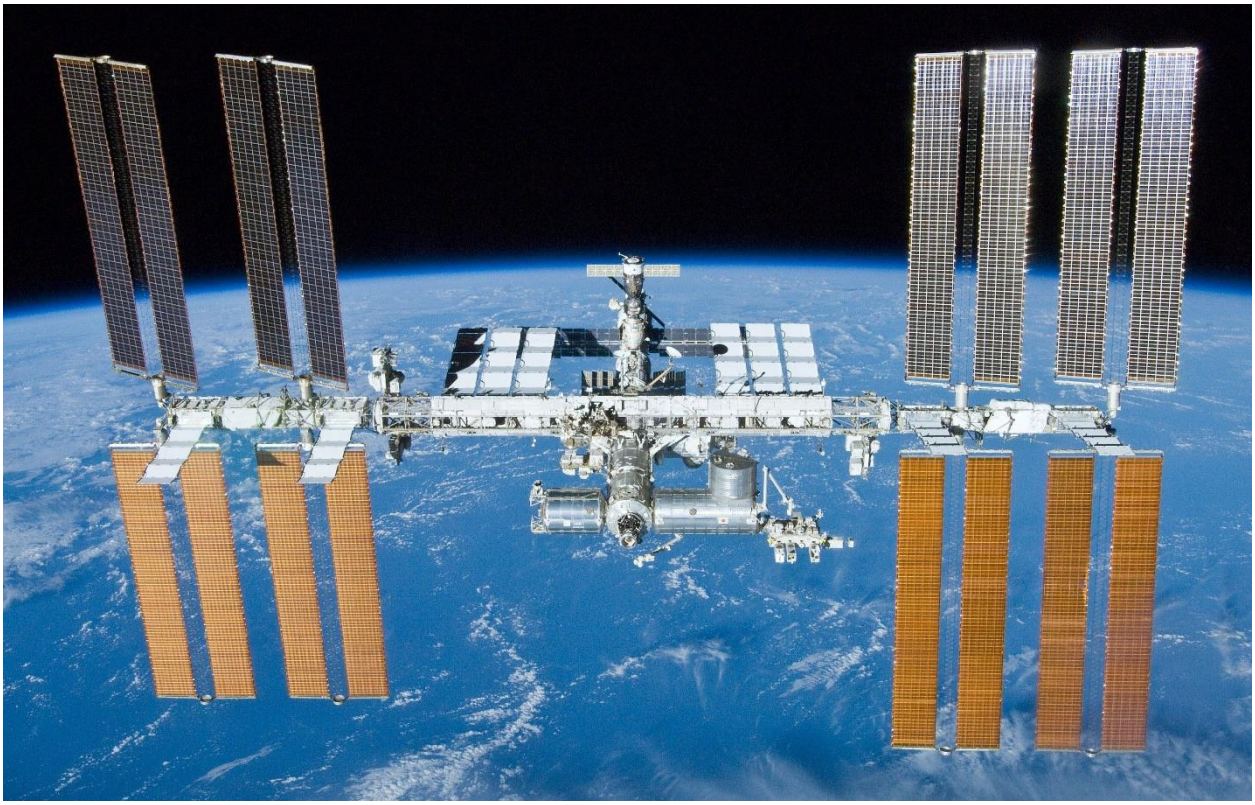
першу пілотовану висадку на Місяць американської місії «Аполлон-11» 20 липня 1969 року.



Порівняння розмірів американських ракет часів космічної гонки.

Радянська космічна програма отримала першість у багатьох досягненнях. Серед них: перша жива істота на орбіті, собака Лайка, в 1957 році, перший політ людини в космос, Юрій Гагарін на борту корабля «Восток-1», у 1961 році, перший вихід у відкритий космос, Олексій Леонов на борту «Восход-2», 18 березня 1965 року, перша автоматична посадка на інше небесне тіло в 1966 році та запуск першої космічної станції «Салют-1» в 1971 році.

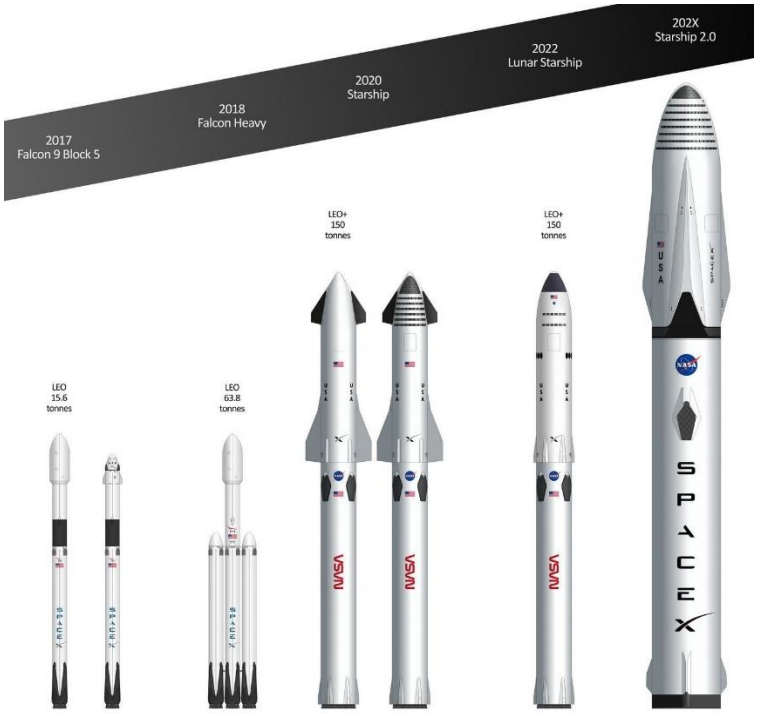
Після перших 20 років досліджень на заміну одноразовим ракетам частково прийшли багаторазові космічні кораблі, такі як «Space Shuttle». Після закінчення холодної війни космічні агенства перейшли від в більшості випадків конкуренції до співпраці, завдяки чому з’явилися міжнародні космічні місії та була побудована Міжнародна космічна станція (МКС).



Міжнародна космічна станція.

З закінченням холодної війни і подальшим розвитком технологій до дослідження космосу приєдналися нові великі світові держави, що стали розробляти свої ракети-носії. У 2000-х роках Китай реалізував успішну програму пілотованих космічних польотів і побудував декілька власних космічних станцій. Європейський Союз і Японія також планують майбутні пілотовані польоти. Китай, Росія та Японія виступають за пілотовані польоти на Місяць у 21 столітті. Європейське Космічне Агенство виступає за пілотовані місії на Місяць і на Марс у 21 столітті.

Крім цього у 2000-х роках приватні космічні компанії, що раніше користувалися виключно державними засобами доставки, всерйоз узялися за розробку власних систем доставки вантажів на орбіту. Приватні інтереси почали фінансувати обмежені програми розвитку, але пізніше, з закінченням програми «Space Shuttle», уряд Сполучених Штатів спонсорував серію ініціатив, щоб стимулювати та заохочувати приватні компанії почати пропонувати вантажні, а пізніше і пасажирські космічні транспортні послуги.



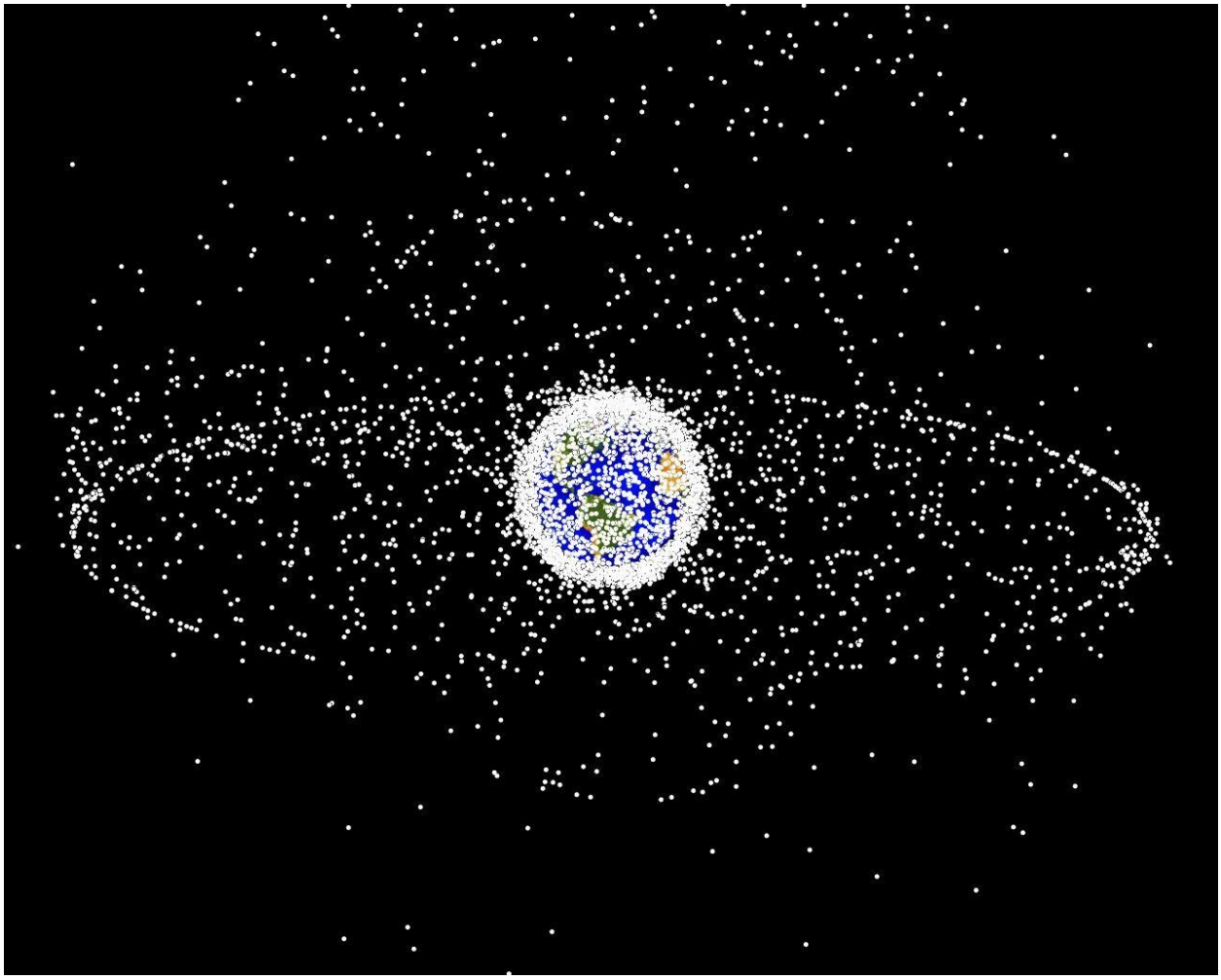
Лінійка сучасних і майбутніх ракет SpaceX – найуспішнішої на сьогодні приватної космічної компанії.

Поступово фокус приватних компаній змістилася з надлегких ракет, що забезпечували мінімальну ціну за один запуск, до важких ракет, що в сучасних умовах мініатюризації компонентів, підвищення надійності, стандартизації і максимального полегшення космічних апаратів, дозволяє запускати багато супутників за раз, і потім розводити їх на необхідні орбіти. Окрім цього, приватні компанії змогли реалізувати і використовувати

багаторазові космічні системи, на противагу «Space Shuttle» США та «Бурану» Радянського Союзу, що в основному не виправдали надій.

Зниження цін на послуги запуску після 2010 року призвели до небаченої раніше конкуренції на ринку космічних запусків. У цій новій парадигмі для вантажних перевезень на низьку навколоземну орбіту в НАСА уклали контракти на послуги з перевезення вантажів на космічних кораблях, розроблених приватно. Для МКС використовуються як приватні, так і державні кораблі постачання. Російські кораблі «Союз» і «Прогрес», а також ATV Європейського космічного агентства (до 2014 року) і японський «Куноторі» (до 2021 року) все ще використовуються.

Люди досягли опосередкованої присутності по всій Сонячній системі, але наймасштабніша діяльність розгорнута на навколоземній орбіті. Люди постійно перебувають на орбіті з 2000 року в складі екіпажів Міжнародної космічної станції, і з невеликими перервами з кінця 1980-х років в складі космічної станції «Мир». Розширення діяльності людини в навколоземному космічному просторі, окрім переваг, також призвело до утворення космічного сміття, що потенційно може призвести до так званого синдрому Кесслера. Синдром Кесслера — це гіпотетичний сценарій, за яким щільність об’єктів на низькій навколоземній орбіті настільки висока, що зіткнення між об’єктами породить каскад уламків, що генерує подальші зіткнення і блокує доступ до орбіти. Це, в свою чергу, створює необхідність в пом’якшенні наслідків та законодавчому регулюванні для забезпечення сталого доступу до космічного простору.



Схема, що показує розташування рукотворних об’єктів у навколоземному просторі, які можливо відстежити.

ДОДАТОК Б

Вихідний код сервісу, що відповідає за пошук та переклад інформації

```
using System;
using System.Linq;
using System.Collections.Generic;
using System.Text.RegularExpressions;
using System.Net.Http;
using System.Threading.Tasks;
using SpacecraftData.Entities;
using SpacecraftSearcher.Interfaces;
using Google.Cloud.Translation.V2;
using HtmlAgilityPack;

namespace SpacecraftSearcher.Concrete
{
    public class SoloSearcher : ISoloSearcher
    {
        public async Task<Spacecraft> SearchForSpacecraft(string NSSDCA)
        {
            Spacecraft spacecraft = new Spacecraft{NSSDC = NSSDCA};

            System.Environment.SetEnvironmentVariable("GOOGLE_APPLICATION_CREDENTIALS",
"sateliteproject-f52711ea42f5.json");

            using (var client = new HttpClient())
            {
                var nasaSearchResult = await
client.GetAsync("https://nssdc.gsfc.nasa.gov/nmc/spacecraft/display.action?id=" +
NSSDCA);
                var searchResultHtml =
nasaSearchResult.Content.ReadAsStringAsync().Result;
                bool error = !nasaSearchResult.IsSuccessStatusCode ||
searchResultHtml.Contains("A valid spacecraft ID must be specified.");
                {
                    return new Spacecraft { Name = "NoData" };
                }

                TranslationClient tclient = TranslationClient.Create();

                spacecraft.Name =
tclient.TranslateTextAsync(GetSpacecraftName(searchResultHtml), LanguageCodes.Ukrainian,
LanguageCodes.English).Result.TranslatedText;
                spacecraft.Country = new Country { Name =
tclient.TranslateTextAsync(GetSpacecraftCountry(searchResultHtml),
LanguageCodes.Ukrainian, LanguageCodes.English).Result.TranslatedText };
                spacecraft.Operator =
tclient.TranslateTextAsync(GetSpacecraftOperator(searchResultHtml),
LanguageCodes.Ukrainian, LanguageCodes.English).Result.TranslatedText;
                spacecraft.Type =
tclient.TranslateTextAsync(GetSpacecraftType(searchResultHtml), LanguageCodes.Ukrainian,
LanguageCodes.English).Result.TranslatedText;
                spacecraft.Mass = GetSpacecraftMass(searchResultHtml);
                spacecraft.Mission =
tclient.TranslateTextAsync(GetSpacecraftMission(searchResultHtml),
LanguageCodes.Ukrainian, LanguageCodes.English).Result.TranslatedText;
                spacecraft.Launch = new Launch
                {
                    Rocket = new Rocket { Name =
tclient.TranslateTextAsync(GetSpacecraftLaunchVehicle(searchResultHtml),
LanguageCodes.Ukrainian, LanguageCodes.English).Result.TranslatedText },
```

```
                    LaunchDate = GetSpacecraftLaunchDate(searchResultHtml),
                    LaunchSite =
tclient.TranslateTextAsync(GetSpacecraftLaunchSite(searchResultHtml),
LanguageCodes.Ukrainian, LanguageCodes.English,
TranslationModel.ServiceDefault).Result.TranslatedText,
                    Success = true
                };
            }
            return spacecraft;
        }

        private string GetSpacecraftName(string html)
        {
            var htmlDocument = new HtmlDocument();
            htmlDocument.LoadHtml(html);
            return htmlDocument.DocumentNode.SelectSingleNode("//h1").InnerText;
        }

        private string GetSpacecraftCountry(string html)
        {
            if (!html.Contains("<h2>Funding Agency</h2><ul><li>")) return "";
            var country = html.Substring(html.IndexOf("<h2>Funding Agency</h2><ul><li>")
+ 31);
            country = country.Substring(country.IndexOf("(") + 1, country.IndexOf(")") -
(country.IndexOf("(") + 1));
            return country;
        }

        private string GetSpacecraftOperator(string html)
        {
            if (!html.Contains("<h2>Funding Agency</h2><ul><li>")) return "";
            var Operator = html.Substring(html.IndexOf("<h2>Funding Agency</h2><ul><li>")
+ 31);
            Operator = Operator.Substring(0, Operator.IndexOf("("));
            return Operator.Trim();
        }

        private string GetSpacecraftType(string html)
        {
            if (!html.Contains("<h2>Disciplines</h2><ul>")) return "";
            var type = html.Substring(html.IndexOf("<h2>Disciplines</h2><ul>") + 24);
            type = type.Substring(0, type.IndexOf("</ul>"));
            var typesHtml = new HtmlDocument();
            typesHtml.LoadHtml(type);
            string types = "";
            foreach (var htmlNode in typesHtml.DocumentNode.ChildNodes)
            {
                types = types + htmlNode.InnerText + ", ";
            }
            types = types.Substring(0, types.Length - 3);
            return types;
        }

        private double GetSpacecraftMass(string html)
        {
            if (!html.Contains("<strong>Mass:</strong>")) return 0.0;
            string MassString = html.Substring(html.IndexOf("<strong>Mass:</strong>") +
23);
            MassString = MassString.Substring(0, MassString.IndexOf("<"));
            MassString = Regex.Replace(MassString, "[^0-9.]", "");
            MassString = MassString.Replace(".", ",");
            var Mass = Convert.ToDouble(MassString);
            return Mass;
        }
```

```
        private string GetSpacecraftMission(string html)
        {
            var missionHtml = new HtmlDocument();
            missionHtml.LoadHtml(html);
            var missions =
missionHtml.DocumentNode.SelectSingleNode(".//*[contains(@class,'urone')]").ChildNodes.To
List();
            missions.RemoveAt(0);
            string mainText = "";
            foreach (var textNode in missions)
            {
                mainText = mainText + textNode.InnerText;
            }
            return mainText;
        }

        private DateTime GetSpacecraftLaunchDate(string html)
        {
            if (!html.Contains("<strong>Launch Date:</strong>")) return new DateTime();
            string LaunchDateString = html.Substring(html.IndexOf("<strong>Launch
Date:</strong>") + 30);
            LaunchDateString = LaunchDateString.Substring(0,
LaunchDateString.IndexOf("<"));
            var date = Convert.ToDateTime(LaunchDateString);
            return date;
        }

        private string GetSpacecraftLaunchVehicle(string html)
        {
            if (!html.Contains("<strong>Launch Vehicle:</strong>")) return "";
            Console.WriteLine(html.Contains("<strong>Launch Vehicle:</strong>"));
            string LaunchVehicle =
html.Substring(html.IndexOf("<strong>Launch Vehicle:</strong>") + 33);
            LaunchVehicle = LaunchVehicle.Substring(0, LaunchVehicle.IndexOf("<"));
            return LaunchVehicle;
        }

        private string GetSpacecraftLaunchSite(string html)
        {
            if (!html.Contains("<strong>Launch Site:</strong>")) return "";
            string LaunchSite =
html.Substring(html.IndexOf("<strong>Launch Site:</strong>") + 30);
            LaunchSite = LaunchSite.Substring(0, LaunchSite.IndexOf("<"));
            return LaunchSite;
        }
    }
       }
```