

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

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

До захисту допущено:

Завідувач кафедри

_____ Олександр РОЛІК

«__» _____ 2025 р.

**Дипломний проєкт
на здобуття ступеня бакалавра
за освітньо-професійною програмою «Інформаційне забезпечення
робототехнічних систем»
спеціальності 126 «Інформаційні системи та технології»
на тему: «Система оптимізації та підбору компонентів роботизованих
платформ»**

Виконав :

студент IV курсу, групи ІК-13

Маковецький Сергій Олександрович

Керівник:

Асистент кафедри ІСТ

Шинкевич Микола Костянтинович

Рецензент:

к.т.н., доцент каф. ОТ

Верба Олександр Андрійович

Засвідчую, що у цьому дипломному
проєкті немає запозичень з праць інших
авторів без відповідних посилань.

Студент _____

Київ – 2025 року

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

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 126 «Інформаційні системи та технології»

Освітньо-професійна програма «Інформаційне забезпечення робототехнічних систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Олександр РОЛІК

«__» _____ 2025 р.

ЗАВДАННЯ
на дипломний проєкт студенту
Маковецькому Сергію Олександровичу

1. Тема проєкту «Система оптимізації та підбору компонентів роботизованих платформ», керівник проєкту асистент кафедри ІСТ Шинкевич. М. К., затверджені наказом по університету від «23» травня 2025 р. № 1705-с
2. Термін подання студентом проєкту: 09.06.2025.
3. Вихідні дані до проєкту: вимоги до функціоналу інформаційної системи підбору компонентів роботизованих платформ, вимоги до точності розрахунку параметрів конфігурацій роботизованої платформи.
4. Зміст пояснювальної записки: опис предметної області, аналіз існуючих рішень, формулювання вимог до системи, вибір технологій розробки, розробка підсистеми, тестування підсистеми, висновки.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо): діаграма послідовності роботи, діаграма таблиць бази даних, схема класів програми, діаграма прецедентів, архітектурна схема, структурна схема.

6. Дата видачі завдання: 03.02.2025

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Опис предметної області	14.04.25	
2	Формулювання вимог до системи	21.04.25	
3	Вибір необхідних технологій	28.04.25	
4	Опис робототехнічних комплексів	05.05.25	
5	Реалізація інформаційної системи	12.05.25	
6	Розгортання та тестування системи	19.05.25	
7	Оформлення пояснювальної записки	26.05.25	

Студент

Сергій МАКОВЕЦЬКИЙ

Керівник

Микола ШИНКЕВИЧ

АНОТАЦІЯ

Система оптимізації та підбору компонентів роботизованих платформ.

Проект містить 61 с. тексту, 30 рисунків, посилання на 20 літературних джерел, 1 додаток та 6 конструкторських документів.

ІНФОРМАЦІЙНА СИСТЕМА, БАЗИ ДАНИХ, КОНФІГУРАЦІЯ ДРОНА, АВТОМАТИЗАЦІЯ ВИБОРУ КОМПОНЕНТІВ, КОРИСТУВАЦЬКИЙ ІНТЕРФЕЙС, АЛГОРИТМИ СУМІСНОСТІ, АНАЛІТИКА ДАНИХ.

Об'єктом роботи є процес автоматизованого підбору компонентів для дрона.

Мета роботи – підвищення ефективності, швидкості та точності вибору комплектуючих для безпілотних літальних апаратів.

Об'єктом дослідження є процес узгодження даних у розподілених нереляційних документо-орієнтованих базах даних.

У дипломному проекті розроблено інформаційну підсистему для підбору компонентів дрона, що автоматизує процес пошуку, сортування та вибору комплектуючих. Актуальність роботи зумовлена складністю підбору сумісних елементів та необхідністю зменшення витрат на закупівлі. Система дозволяє переглядати доступні компоненти, здійснювати пошук, фільтрувати за технічними параметрами та формувати оптимальні конфігурації. Реалізація базується на C# в Unity, використовується System.Data.SqlClient для інтеграції з SQL Server.

Описано архітектуру, базу даних, зв'язки між сутностями та ключові точки API. Після розробки проведено тестування CRUD-операцій та перевірку продуктивності SQL-запитів.

SUMMARY

System for Optimization and Selection of Components for Robotic Platforms.

The project contains 61 pages of text, 30 illustrations, references to 20 literary sources, 1 appendice and 6 design documents.

Keywords: information system, databases, drone configuration, automated component selection, user interface, compatibility algorithms, data analytics.

The object of development is the process of automated selection of components for a drone.

The aim of the work is to enhance the efficiency, speed, and accuracy of selecting components for unmanned aerial vehicles.

The subject of the research is the process of data reconciliation in distributed, non-relational, document-oriented databases.

In the diploma project, an information subsystem for the selection of drone components has been developed to automate the process of searching, sorting, and choosing parts. The relevance of the work is due to the complexity of selecting compatible components and the need to reduce procurement costs. The system allows users to view available components, perform searches, filter by technical parameters, and generate optimal configurations. The implementation is based on C# in Unity, using System.Data.SqlClient for integration with SQL Server. The project describes the system's architecture, database, entity relationships, and key API endpoints. After development, thorough testing of CRUD operations and SQL query performance was conducted.

№ рядка	Формат	Позначення	Найменування	Кіл. аркушів	№ екз.	Примітка
1			Документація загальна			
2						
3			Знову розроблена			
4						
5	A4	ІК13.160БАК.006 ПЗ	Система оптимізації та підбору	61		
6			компонентів роботизованих плат-			
7			форм. Пояснювальна записка			
8	A3	ІК13.160БАК.006 Д1	Система оптимізації та підбору	1		
9			компонентів роботизованих плат-			
10			форм. Структурна схема			
11	A3	ІК13.160БАК.006 Д2	Система оптимізації та підбору	1		
12			компонентів роботизованих плат-			
13			форм. Архітектурна схема			
14	A3	ІК13.160БАК.006 Д3	Система оптимізації та підбору	1		
15			компонентів роботизованих плат-			
16			форм. Діаграма прецедентів			
17	A3	ІК13.160БАК.006 Д4	Система оптимізації та підбору	1		
18			компонентів роботизованих плат-			
19			форм. Діаграма таблиць бази			
20			даних			
21	A3	ІК13.160БАК.006 Д5	Система оптимізації та підбору	1		
22			компонентів роботизованих плат-			
23			форм. Схема класів			
24	A3	ІК13.160БАК.006 Д6	Система оптимізації та підбору	1		
25			компонентів роботизованих плат-			
26			форм. Діаграма послідовності			
27						
28						
Зм.	Лист	№ докум.	Підпис			
Розробив		Маковецький С.О.				
Перевірив		Шинкевич М.К.				
Затв.						
				ІК13.160БАК.006 ТП		
				Система оптимізації та підбору		
				компонентів роботизованих плат-		
				форм.		
				Відомість дипломного проєкту		
				Літ.		
				Арк.		
				Аркушів		
				Т		
				1		
				1		
				КПІ ім. Ігоря Сікорського		
				Група ІК-13		

**Пояснювальна записка
до дипломного проєкту
на тему: «Система оптимізації та підбору компонентів
роботизованих платформ»**

Київ – 2025 року

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	5
ВСТУП	6
1 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ	8
1.1 Різновиди інформаційних систем.....	9
1.2 Різновиди баз даних	10
Висновки до розділу 1	11
2 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ.....	12
2.1 Огляд існуючих інформаційних систем для оптимізації закупівель	12
2.1.1 Загальні характеристики систем управління закупівлями.....	12
2.1.2 Сфери застосування існуючих рішень	12
2.2 Системи управління закупівлями	13
2.2.1 Аналіз SAP Ariba.....	13
2.2.2 Аналіз Oracle Procurement Cloud	13
2.2.3 Аналіз Coupa Procurement	13
2.3 CRM-системи із модулями закупівель.....	14
2.3.1 Аналіз Zoho CRM.....	14
2.3.2 Аналіз Salesforce із модулями закупівель.....	14
2.4 Спеціалізовані рішення для підбору електронних компонентів.....	15
2.4.1 Аналіз Octopart	15
2.4.2 Аналіз Digi-Key	15
2.5 Порівняльний аналіз існуючих рішень	16
2.5.1 Основні переваги існуючих рішень	16
2.5.2 Основні недоліки існуючих рішень	17
Висновки до розділу 2	18
3 ВИБІР ТЕХНОЛОГІЙ РОЗРОБКИ.....	20

					ІК13.160БАК.006 ПЗ			
Зм.	Лист	№ докум.	Підпис					
Розробив	Маковецький С.О.				Система оптимізації та підбору компонентів роботизованих плат-форм. Пояснювальна записка	Літ.	Арк.	Аркушів
Перевірив	Шинкевич М.К.					Т	2	61
						КПІ ім. Ігоря Сікорського		
Затв.						Група ІК-13		

3.1 Вибір середовища розробки.....	20
3.2 Вибір операційної системи розгортання.....	21
3.3 Система контролю версій Git.....	22
3.4 Системи управління базою даних	22
3.5 Фреймворки	23
3.5.1 .NET Framework.....	23
3.5.2 Unity UI Toolkit.....	23
3.5.3 TextMeshPro	24
3.6 Архітектурні підходи.....	24
Висновки до розділу 3	25
4 РОЗРОБКА ДІАГРАМИ ПРЕЦЕДЕНТІВ	26
5 РОЗРОБКА БАЗИ ДАНИХ	28
6 РОЗРОБКА СТРУКТУРИ ІНФОРМАЦІЙНОЇ СИСТЕМИ	32
6.1 Архітектура інформаційної системи	32
6.2 Інтеграція бази даних до інформаційної системи.....	32
6.3 Реалізація CRUD-функціоналу	37
6.5 Отримання конфігурації дрону.....	40
6.6 Розробка інтерфейсу програми.....	42
6.6.1 Розробка інтерфейсу перегляду компонентів	42
6.6.2 Розробка інтерфейсу отримання конфігурації дрону.....	43
6.6.3 Перехід між інтерфейсами	45
6.7 Створення версії для поширення.....	45
Висновки до розділу 6	47
7 ТЕСТУВАННЯ СИСТЕМИ	48
7.1 Мета випробувань	48
7.2 Тестування CRUD операцій	48
7.2.1 Тестування операції читання таблиць компонентів	48
7.2.2 Тестування операції створення.....	50
7.2.3 Тестування операції оновлення	52
7.2.4 Тестування операції видалення	54

7.3 Тестування отримання конфігурації роботизованої системи..... 56

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ..... 60

ДОДАТОК А..... 62

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – Інтерфейс програмування додатків;

Branch – Гілка в репозиторії для розділення розробки;

C# – Мова програмування для реалізації бізнес-логіки проєкту;

Commit – Збереження змін у локальному репозиторії Git;

CRUD – Операції створення, читання, оновлення та видалення даних;

DB – База даних;

Entity Framework – ORM-інструмент для роботи з базами даних у C#;

Git – Система контролю версій;

JSON – Формат обміну даними між клієнтськими та серверними додатками;

Merge – Об'єднання змін у коді між гілками Git;

Push – Надсилання локальних змін у віддалений репозиторій;

REST – Архітектурний стиль для розробки веб-сервісів;

SQL – Структурована мова запитів для управління базами даних;

SQL Server – Система управління реляційною базою даних

System.Data.SqlClient – Бібліотека .NET для взаємодії з SQL Server;

TextMeshPro – Технологія для якісного відображення тексту в UI;

UI – Користувацький інтерфейс;

Unity – Ігровий рушій, що використовується для розробки інтерфейсу;

WebGL – Технологія для запуску додатків у браузері

ВСТУП

Інформаційна система оптимізації закупівель компонентів дронів є складовою сучасних інформаційних технологій, що забезпечує автоматизацію процесів підбору, аналізу та придбання комплектуючих для безпілотних літальних апаратів. Вона дозволяє ефективно обробляти великі обсяги даних щодо технічних характеристик, цінових пропозицій та наявності товарів, сприяючи прийняттю обґрунтованих рішень при формуванні технічної комплектації дронів.

На сучасному етапі розвитку безпілотних технологій попит на дрони зростає в різних галузях: від аграрного сектору до промислового моніторингу та безпеки. З огляду на широкий асортимент компонентів на ринку, процес їх підбору стає дедалі складнішим. Часті зміни у характеристиках товарів, поява нових постачальників, динаміка цін та різноманітність вимог до технічних параметрів дронів зумовлюють необхідність створення спеціалізованих інформаційних систем, що автоматизують процес аналізу і оптимізації закупівель. Тому розробка такої системи є важливим завданням для підвищення ефективності розробки та експлуатації безпілотних літальних апаратів.

Необхідність використання подібних систем проявляється у наступних випадках:

конструювання індивідуальних дронів. В умовах, коли необхідно створити апарат під специфічні вимоги замовника, інформаційна система забезпечує швидкий підбір найбільш відповідних компонентів;

освітні та наукові проєкти. Для навчальних закладів та дослідницьких організацій система допомагає оперативно формувати комплектуючі для практичних і теоретичних завдань із урахуванням обмеженого бюджету;

масове виробництво дронів. У випадку серійного виробництва, оптимізація закупівель дозволяє зменшити витрати, підвищити рентабельність проєкту та швидко реагувати на зміни ринку;

охорона та безпека. Компанії, що забезпечують відеомоніторинг та патрулювання територій, часто потребують дронів із специфічними камерами,

					ІК13.160БАК.006 ПЗ	Арк.
						6
Зм.	Лист	№ докум.	Підпис	Дата		

датчиками та системами зв'язку. Оптимізація закупівель знижує витрати на технічне оснащення та підвищує ефективність охоронних заходів;

екологічний моніторинг. Організації, що займаються дослідженням змін клімату, якості повітря, стану лісових масивів або водних ресурсів, використовують дрони для збору даних. Автоматизована система допомагає обирати комплектуючі, що відповідають потребам дослідників.

Метою дипломної роботи є розробка інформаційної системи для оптимізації процесу закупівель компонентів безпілотних літальних апаратів шляхом автоматизації аналізу технічних характеристик, вартості та наявності товарів.

Об'єктом дослідження є процес підбору та закупівлі комплектуючих для безпілотних літальних апаратів.

Предметом дослідження є методи та алгоритми автоматизованої обробки даних, що використовуються для оптимізації вибору комплектуючих у рамках інформаційних систем.

					ІК13.160БАК.006 ПЗ	Арк.
						7
Зм.	Лист	№ докум.	Підпис	Дата		

1 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

Сучасний світ безпілотних літальних апаратів (дронів) розвивається стрімкими темпами, що призводить до значного розширення асортименту доступних компонентів і ускладнення процесу їх вибору та закупівлі. Ефективність створення дронів напряду залежить від правильного підбору комплектуючих, які відповідають вимогам проекту за технічними характеристиками, вартістю, надійністю та сумісністю.

У сфері комплектування дронів виділяють основні категорії компонентів:

- двигуни та пропелери, які визначають потужність, маневреність і вантажопідйомність апарата;
- акумулятори, що впливають на тривалість польоту, вагу та енергозабезпечення системи;
- контролери польоту, відповідальні за стабільність, балансування та виконання автоматичних маневрів;
- сенсори (GPS, гіроскопи, камери, далекоміри та інші), необхідні для навігації, збору даних та взаємодії з навколишнім середовищем;
- рами та конструктивні елементи, що формують основу апарата і визначають його міцність та вагові характеристики;
- системи зв'язку для передачі команд і даних між дроном і оператором або центром управління.

Окрім базових вимог, сучасні задачі вимагають індивідуалізації дронів під спеціалізовані сценарії використання, такі як моніторинг сільськогосподарських угідь, забезпечення безпеки об'єктів, картографування місцевості чи доставка вантажів. Це ще більше ускладнює процес підбору компонентів, оскільки потрібно враховувати специфічні потреби місії.

Зростання ринку дронів також супроводжується збільшенням кількості виробників і постачальників комплектуючих, що створює додаткові труднощі:

- необхідність швидкої актуалізації інформації про ціни, наявність та технічні характеристики;

					ІК13.160БАК.006 ПЗ	Арк.
						8
Зм.	Лист	№ докум.	Підпис	Дата		

- оцінка надійності постачальників і брендів;
- автоматизоване порівняння альтернативних пропозицій за заданими критеріями;
- обробка великих обсягів даних про товари у режимі реального часу;
- уніфікація форматів даних із різних джерел;
- адаптація до динаміки ринку, включаючи тимчасову відсутність або заміну моделей;
- забезпечення сумісності між новими та вже наявними компонентами.

Особливої актуальності набуває інтеграція інформаційних систем із зовнішніми базами даних виробників, маркетплейсів та аналітичних сервісів, що дозволяє забезпечити оперативність, точність і повноту інформації для ухвалення оптимальних рішень.

Таким чином, інформаційна система для оптимізації закупівель компонентів дронів має вирішувати комплекс завдань:

- забезпечувати зручний пошук і фільтрацію компонентів за технічними і економічними параметрами;
- автоматично оновлювати дані про наявність і ціни;
- формувати рекомендації щодо сумісних комбінацій комплектуючих;
- підтримувати аналіз ринку та тенденцій попиту.

Створення такої системи дозволяє значно скоротити час на формування замовлень, підвищити якість кінцевих продуктів і зменшити ризики, пов'язані з вибором неякісних або несумісних деталей, що є критично важливим для ефективного функціонування сучасних безпілотних технологій.

1.1 Різновиди інформаційних систем

Інформаційні системи класифікуються за призначенням:

- операційні системи (Operational Systems) — реєструють і обробляють транзакції (продажі, платежі), забезпечуючи актуальні дані для управління;

- інформаційно-аналітичні системи (MIS) — узагальнюють дані з операційних систем, формують звіти, допомагають у плануванні та контролі;
- системи підтримки прийняття рішень (DSS) — аналізують дані та моделюють сценарії для керівних рішень у складних ситуаціях;
- системи керування знаннями (KMS) — зберігають та поширюють експертні знання, підвищуючи ефективність бізнес-процесів;
- експертні системи (Expert Systems) — імітують мислення експертів у вузьких галузях, пропонуючи рекомендації та діагностику;
- інтегровані системи управління підприємством (ERP) — об'єднують бізнес-процеси в єдиній платформі, автоматизуючи взаємодію між відділами;
- системи управління взаємовідносинами з клієнтами (CRM) — аналізують дані про клієнтів, автоматизують комунікацію, покращують маркетинг;
- системи бізнес-аналітики (BI) — агрегують дані, надають аналітичні інструменти для оцінки ефективності й підтримки стратегічних рішень.

1.2 Різновиди баз даних

Бази даних є основною складовою будь-якої сучасної інформаційної системи, забезпечуючи організоване зберігання та ефективний доступ до даних. З розвитком технологій і зростанням обсягів оброблюваної інформації виникла необхідність у використанні різних моделей баз даних, кожна з яких орієнтована на вирішення специфічних задач. Від правильного вибору типу бази даних залежить ефективність роботи інформаційної системи, зокрема системи оптимізації закупівель компонентів для дронів, де важливими є швидкість обробки запитів, масштабованість і гнучкість структури даних.

На сьогодні найбільш поширеними різновидами баз даних є реляційні та нереляційні бази даних.

Висновки до розділу 1

Було проведено аналіз предметної області. Було виявлено основні потреби користувачів у цьому процесі, зокрема: необхідність врахування сумісності компонентів, орієнтація на бюджет користувача, вагові та енергоспоживчі характеристики дронів, а також відсутність зручного інструменту для систематизації й підбору обладнання.

У результаті аналізу було сформульовано вимоги до інформаційної системи, яка має на меті надати користувачеві можливість вводити параметри дрона, отримувати список сумісних комплектуючих відповідно до технічних і вартісних обмежень, а також отримувати рекомендації на основі оптимізації заданих критеріїв. Такий підхід дозволить зменшити час на підбір обладнання, знизити ризик помилок і забезпечити більш ефективне використання бюджету.

					ІК13.160БАК.006 ПЗ	Арк.
						11
Зм.	Лист	№ докум.	Підпис	Дата		

2 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

2.1 Огляд існуючих інформаційних систем для оптимізації закупівель

2.1.1 Загальні характеристики систем управління закупівлями

Інформаційні системи управління закупівлями спрямовані на автоматизацію всіх етапів закупівельної діяльності: від пошуку постачальників до укладення контрактів та контролю постачання товарів і послуг. Вони забезпечують:

- централізовану базу даних постачальників і товарів;
- механізми аналізу витрат і ефективності закупівель;
- інструменти для автоматизації запитів пропозицій (RFP) та оцінки постачальників;
- можливості інтеграції із системами бухгалтерського обліку та управління складом.

Такі системи орієнтовані переважно на корпоративний сектор і великі підприємства, де контроль закупівельних процесів має критичне значення для оптимізації витрат і забезпечення ефективності логістичних ланцюгів [1].

2.1.2 Сфери застосування існуючих рішень

В галузі безпілотних літальних апаратів готові рішення застосовуються обмежено, через те що більшість платформ не передбачають технічної оцінки сумісності компонентів або моделювання майбутнього апарату на основі підібраних елементів.

Існуючі системи управління закупівлями переважно застосовуються у наступних сферах:

- промисловість (закупівля сировини, обладнання);
- будівництво (закупівля матеріалів, техніки);
- охорона здоров'я (закупівля медичних товарів);
- енергетика (управління тендерами на великі постачання).

2.2 Системи управління закупівлями

2.2.1 Аналіз SAP Ariba

SAP Ariba є одним з лідерів серед корпоративних платформ управління закупівлями. Основні можливості системи:

- пошук постачальників на глобальному ринку;
- управління контрактами і тендерами;
- автоматизація процесів узгодження закупівель;
- розширена аналітика витрат.

SAP Ariba орієнтована на великі підприємства та корпорації. Одним із недоліків для використання у проєкті закупівель компонентів дронів є складність налаштування системи під конкретну галузеву специфіку, а також висока вартість ліцензування [2].

2.2.2 Аналіз Oracle Procurement Cloud

Oracle Procurement Cloud пропонує хмарне рішення для централізованого управління закупівлями [3]. Функціональність включає:

- автоматизовану генерацію замовлень;
- інтеграцію з фінансовими і складськими системами;
- інструменти для аудиту і контролю витрат.

Платформа добре підходить для уніфікації закупівельних процесів у великих компаніях, однак, як і SAP Ariba, не має вбудованих механізмів для технічного підбору сумісних комплектуючих .

2.2.3 Аналіз Coupa Procurement

Coupa Procurement є гнучкою системою управління витратами та закупівлями [4], яка дозволяє:

- автоматизувати процеси закупівель;

					ІК13.160БАК.006 ПЗ	Арк.
						13
Зм.	Лист	№ докум.	Підпис	Дата		

- аналізувати ефективність постачальників;
- планувати бюджети на основі реальних закупівельних даних.

Coopra вирізняється серед конкурентів простотою впровадження і нижчою вартістю володіння у порівнянні з SAP та Oracle. Проте й ця система не враховує специфіки інженерного підбору компонентів для безпілотних літальних апаратів, що є критичним для успішної оптимізації закупівель у даній предметній області.

2.3 CRM-системи із модулями закупівель

2.3.1 Аналіз Zoho CRM

Zoho CRM є потужною платформою для управління взаємовідносинами з клієнтами, яка також має розширення для закупівельних процесів [5]. Основні можливості:

- створення профілів постачальників;
- управління історією замовлень;
- налаштування автоматичних нагадувань щодо оновлення контактів чи необхідності перезакупівлі;
- інтеграція з аналітичними модулями для оцінки постачальників.

Zoho CRM зручно використовувати для організації контактної бази постачальників та контролю статусів співпраці. Проте система не має вбудованого функціоналу для технічного аналізу або автоматичної перевірки сумісності придбаних компонентів.

2.3.2 Аналіз Salesforce із модулями закупівель

Salesforce є однією з найпотужніших CRM-платформ, яка пропонує можливість кастомізації для управління постачальниками та закупівлями [6]. Основні можливості:

- створення кастомних об'єктів для управління каталогами товарів;
- відстеження взаємодії з постачальниками;

					ІК13.160БАК.006 ПЗ	Арк.
						14
Зм.	Лист	№ докум.	Підпис	Дата		

- налаштування процесів узгодження закупівель;
- потужні аналітичні інструменти для оцінки витрат.

Salesforce дозволяє інтегрувати закупівельний процес із загальною CRM-стратегією компанії, проте вартість використання платформи є високою, а налаштування вимагає значних часових і технічних ресурсів.

2.4 Спеціалізовані рішення для підбору електронних компонентів

2.4.1 Аналіз Octopart

Octopart є пошуковою системою для електронних компонентів, яка агрегує інформацію від великої кількості постачальників [7]. Основні можливості:

- пошук компонентів за назвою, частковим кодом або характеристиками;
- порівняння цін і наявності у реальному часі;
- перегляд документації (datasheets) і технічних специфікацій;
- можливість інтеграції з інженерними CAD-системами.

Octopart є особливо корисним у випадках, коли необхідно знайти електронні модулі для проектування безпілотних апаратів, такі як контролери польоту, сенсори або плати живлення. Проте платформа не забезпечує повноцінного механізму побудови або валідації сумісних систем з кількох компонентів.

2.4.2 Аналіз Digi-Key

Digi-Key — один із найбільших світових дистриб'юторів електронних компонентів, який пропонує також онлайн-інструменти для підбору товарів [8]. Основні функції:

- каталог товарів із докладними технічними параметрами;
- онлайн-калькулятори для розрахунку електронних характеристик;
- інтеграція з платформами проектування (наприклад, KiCad, Altium).

					ІК13.160БАК.006 ПЗ	Арк.
						15
Зм.	Лист	№ докум.	Підпис	Дата		

Для створення інформаційної системи оптимізації закупівель компонентів дронів досвід використання Digi-Key може бути цінним з точки зору організації пошуку технічної інформації та відображення сумісних елементів.

2.5 Порівняльний аналіз існуючих рішень

Для обґрунтування необхідності розробки власної інформаційної системи було здійснено порівняльний аналіз існуючих рішень за ключовими критеріями, релевантними для цілей проекту оптимізації закупівель компонентів дронів.

Основні критерії оцінки:

- спеціалізація на роботі з технічними компонентами для безпілотних систем;
- можливість автоматичної перевірки сумісності компонентів;
- наявність функціоналу гнучкого підбору за технічними параметрами (вага, розміри, енергоспоживання тощо);
- гнучкість інтеграції в кастомні рішення;
- доступність використання (вартість, простота впровадження).

2.5.1 Основні переваги існуючих рішень

Готові рішення мають низку переваг:

- наявність широкої бази постачальників і товарів (Alibaba, Global Sources);
- інтеграція процесів закупівлі та фінансового обліку (SAP Ariba, Oracle Procurement Cloud);
- наявність розвиненої аналітики витрат (Coupa, Salesforce);
- зручність пошуку електронних компонентів (Octopart, Digi-Key).

Ці можливості можуть бути використані як приклади для проєктування функціоналу власної системи, наприклад, реалізації розширеного пошуку чи інтеграції аналітики закупівель.

					ІК13.160БАК.006 ПЗ	Арк.
						16
Зм.	Лист	№ докум.	Підпис	Дата		

2.5.2 Основні недоліки існуючих рішень

Разом із тим, більшість існуючих систем мають такі недоліки:

- відсутність спеціалізації під потреби безпілотних літальних апаратів;
- нездатність перевіряти сумісність окремих компонентів автоматично;
- висока вартість впровадження (SAP Ariba, Oracle Procurement Cloud);
- необхідність значних технічних ресурсів для кастомізації (Salesforce).

Ці недоліки є визначальними факторами на користь розробки власної адаптивної інформаційної системи, орієнтованої на специфіку проектування та закупівель дронів.

Результати аналізу існуючих систем наведені у таблиці 1.1. Даний аналіз показав, що хоча окремі системи пропонують розвинені функції управління закупівлями або пошуку товарів, жодна з них не задовольняє повною мірою специфічні вимоги, пов'язані з оптимізацією закупівель комплектуючих для безпілотних літальних апаратів.

Таблиця 1.1 – Порівняльний аналіз існуючих рішень

Назва рішення	Спеціалізація під дрони	Автоматичний аналіз сумісності	Гнучкість налаштування	Простота інтеграції	Вартість
SAP Ariba	Ні	Ні	Висока	Середня	Висока
Oracle Procurement Cloud	Ні	Ні	Висока	Середня	Висока
Coupa Procurement	Ні	Ні	Висока	Висока	Висока
Alibaba	Ні	Ні	Середня	Висока	Низька

Назва рішення	Спеціалізація під дрони	Автоматичний аналіз сумісності	Гнучкість налаштування	Простота інтеграції	Вартість
ThomasNet	Ні	Ні	Низька	Середня	Безкоштовно
Global Sources	Ні	Ні	Середня	Висока	Низька
Zoho CRM	Ні	Ні	Середня	Висока	Середня
Salesforce	Ні	Ні	Висока	Середня	Висока
Octopart	Частково	Ні	Висока	Висока	Безкоштовно / Платно
Digi-Key	Частково	Ні	Висока	Висока	Безкоштовно

Результати проведеного аналізу обґрунтовують доцільність розробки спеціалізованої інформаційної системи оптимізації закупівель компонентів дронів, яка буде враховувати особливості побудови безпілотних апаратів, забезпечуватиме гнучкий підбір елементів за технічними характеристиками, перевірку їх сумісності та ефективне управління даними про постачальників і товари.

Висновки до розділу 2

Було проведено аналіз існуючих інформаційних систем, що орієнтовані на автоматизацію закупівельних процесів, а також спеціалізованих платформ для пошуку технічних компонентів. Розглянуто можливості таких рішень, як SAP Ariba,

Oracle Procurement Cloud, Coupa Procurement, Alibaba, Zoho CRM, Octopart та інших.

Деякі функції існуючих рішень можна використати як приклад для створення функцій у інформаційній системі, а саме:

- зберігати дані про постачальників, ціни та наявність;
- мати зручний і гнучкий інтерфейс для кінцевого користувача;
- підтримувати можливість розширення функціональності у майбутньому;
- мати можливість інтеграції зі стороніми сервісами.

Розробка такої системи дозволить оптимізувати витрати часу і ресурсів на етапі проектування та закупівель, а також забезпечить більшу ефективність підбору комплектуючих для безпілотних літальних апаратів.

					ІК13.160БАК.006 ПЗ	Арк.
						19
Зм.	Лист	№ докум.	Підпис	Дата		

3 ВИБІР ТЕХНОЛОГІЙ РОЗРОБКИ

3.1 Вибір середовища розробки

Під час дослідження були розглянуті найпоширеніші сучасні середовища розробки, серед яких особливу увагу було приділено таким, як Unity [9], Unreal Engine [10], Godot [11] та CryEngine [12].

Unreal Engine є надзвичайно потужним інструментом для створення програм із високоякісною 3D-графікою [10]. Втім, аналіз Unreal Engine у рамках поточного проєкту виявив певні обмеження, які унеможливають його використання для розробки інформаційної системи. Дані обмеження включають: великий об'єм пам'яті потрібний для готового продукту та складність у підключенні бази даних.

Godot Engine має відкритий кодом і просту структуру роботи [11]. На даний момент середовище перебуває в активному розвитку, тому у нього повністю відсутній функціонал, що дозволяє використовувати бази даних.

CryEngine було популярним середовищем для створення ігрових застосунків у минулому [12]. CryEngine зовсім не призначений для реалізації інформаційних систем і не містить інструменти для роботи з формами, таблицями чи взаємодії з базами даних.

Unity Engine є одним із найпопулярніших та найбільш гнучких інструментів для розробки інтерактивних застосунків [9]. Дане середовище має найбільше позитивних аспектів для створення інформаційної системи, такі як:

- компактний розмір середовища та готового продукту;
- вбудовані засоби роботи з таблицями;
- можливості інтеграції зовнішніх бібліотек (наприклад, для роботи з базами даних).

Таким чином, для розробки інформаційної системи Unity є найкращим середовищем завдяки легкості створених програм, модульній архітектурі, універсальності, можливості підключення різних бібліотек та нативними можливостями роботи з таблицями.

3.2 Вибір операційної системи розгортання

Під час аналізу потенційних платформ були розглянуті такі варіанти: Windows, Web (браузерні додатки), Android, iOS.

Розробка під Windows дозволяє швидко тестувати функціональність без потреби у додаткових емуляторах чи складній конфігурації. Більшість користувачів уже вміють працювати з цією платформою, отже розроблений застосунок буде одразу зрозумілим та доступним цільовій аудиторії.

Web-платформа дозволяє запускати програму безпосередньо у браузері без потреби в установці. Перевагою такого підходу є легкий доступ до системи з будь-якого пристрою. Проте WebGL має бути набагато більше захищена від кібератак. Дана платформа поширення може бути розглянута як додаткова, але не як основна на етапі первинної реалізації.

Платформа Android надає великі можливості для створення портативних застосунків.. Проте відображення інтерфейсу на мобільних пристроях потребує його суттєвої зміни, а також адаптації логіки під сенсорне управління.

Платформа iOS також була розглянута, однак вона має ще вищий поріг входження — для розгортання застосунку потрібна реєстрація в платній програмі розробника Apple Developer Program, а також суворіші вимоги до оптимізації й безпеки. Зважаючи на це, iOS була відхилена як платформа для першого релізу.

У підсумку, основною платформою для реалізації системи було обрано Windows, оскільки вона є найпопулярнішою платформою для користувачів [14]. В майбутньому, за наявності попиту з боку користувачів, буде можливим адаптувати систему під Web [15] або Android з урахуванням відповідних змін в архітектурі інтерфейсу.

3.3 Система контролю версій Git

Система контролю версій Git дозволяє зберігати всю історію змін у локальному репозиторії, що забезпечує максимальну автономність під час роботи [16].

З метою централізованого збереження та резервного копіювання репозиторію, для цього проєкту використовується хостинг GitHub. Завдяки GitHub з'являється можливість організовувати завдання та сповіщення через Issues та Projects, а також надавати доступ до проєкту іншим членам команди (у разі переходу проєкту на командну розробку). (Додаток А)

У контексті розробки інформаційної системи оптимізації закупівель компонентів дронів Git дозволяє ізолювати роботу над складними модулями, такими як підключення бази даних, візуалізація елементів або генерація інтерфейсу, а потім інтегрувати їх до загального репозиторію без порушення цілісності проєкту.

Таким чином, використання Git у поєднанні з GitHub є важливим інструментом для ефективної та безпечної розробки дипломного проєкту.

3.4 Системи управління базою даних

Для реалізації підсистеми підбору компонентів дрона були розглянуті різні варіанти реляційних баз даних, серед яких Microsoft SQL Server, MySQL, PostgreSQL та SQLite.

Microsoft SQL Server було обрано для зберігання інформації про компоненти дрона. Ця СУБД підтримує складні SQL-запити, забезпечує високу продуктивність при обробці даних, має широкі можливості безпеки, включаючи рівневе управління доступом та шифрування. Крім того, вона легко інтегрується з платформою .NET через використання бібліотеки ADO.NET.

ADO.NET містить набори класів, таких як SqlConnection, SqlCommand, SqlDataReader та інші, які забезпечують встановлення з'єднання з сервером баз даних, виконання SQL-запитів, зчитування результатів та управління транзакціями. Використовуючи ці компоненти, програми може безпосередньо звертатись до бази даних, отримувати потрібну інформацію про компоненти дрона та передавати її у бізнес-логіку для подальшої обробки.

3.5 Фреймворки

Фреймворки грають важливу роль у процесі розробки програмного забезпечення оскільки вони надають готові інструменти та засоби для вирішення типових завдань, таких як робота з даними, побудова інтерфейсів та реалізація бізнес-логіки. Використання фреймворків сприяє стандартизації коду, пришвидшує розробку та полегшує підтримку системи.

3.5.1 .NET Framework

.NET Framework є повноцінною платформою для розробки застосунків, яка забезпечує стабільне середовище виконання коду [17]. Завдяки широкій бібліотеці класів розробники отримують доступ до готових рішень для роботи з файлами, мережею, базами даних та іншими компонентами. Ця платформа тісно інтегрується з мовою C#, що дозволяє писати об'єктно-орієнтований код, спрямований на створення надійних і масштабованих додатків.

3.5.2 Unity UI Toolkit

Unity UI Toolkit є сучасним інструментом для створення користувацьких інтерфейсів у середовищі Unity. Завдяки декларативному підходу до розробки інтерфейсу, розробники можуть побудувати складну ієрархію візуальних елементів, подібну до DOM у веб-розробці. Використання стилів у вигляді USS

					ІК13.160БАК.006 ПЗ	Арк.
						23
Зм.	Лист	№ докум.	Підпис	Дата		

(Unity Style Sheets) дає змогу централізовано керувати зовнішнім виглядом усіх UI-компонентів. Інтерактивність, адаптивність та швидке тестування інтерфейсу безпосередньо в Unity Editor сприяють створенню зручного та естетично привабливого інтерфейсу.

3.5.3 TextMeshPro

TextMeshPro — це фреймворк для рендерингу тексту в Unity, який значно покращує якість відображення текстових елементів. Завдяки цій технології текст залишається чітким і читабельним навіть на високих масштабах та пристроях з високою роздільною здатністю. Розширене форматування, можливість налаштування кернінгу і інтервалів між символами дозволяють розробникам створювати динамічні текстові блоки, що гармонійно інтегруються в загальний дизайн інтерфейсу. TextMeshPro сприяє як технічній, так і візуальній складовій проекту, допомагаючи передати інформацію користувачам.

3.6 Архітектурні підходи

Для розробки інформаційної системи було прийнято рішення про впровадження багаторівневої архітектури, що забезпечує чітке розділення презентаційного, логічного та даних рівнів. Такий підхід спрощує написання та підтримку коду, а також полегшує інтеграцію із зовнішніми сервісами та розширення функціональних можливостей системи в майбутньому.

Також будуть використані принципи модульного підходу та патерн «розділення обов’язків». Модульна архітектура дозволяє виділити окремі логічні блоки, які взаємодіють між собою через чітко визначені інтерфейси. Така організація допомагає ізолювати потенційні проблеми, а також дає можливість розробляти і впроваджувати нові функції без кардинальних змін в існуючій системі.

					ІК13.160БАК.006 ПЗ	Арк. 24
Зм.	Лист	№ докум.	Підпис	Дата		

Використання підходів Model-View-Controller (MVC) дозволяє чітко розмежувати логіку презентації та взаємодії користувача з внутрішньою логікою системи. Це сприяє утриманню інтерфейсу в окремому шарі, що полегшує адаптацію під різні цільові платформи і дозволяє оперативно оновлювати UI, не порушуючи логіку роботи додатку.

Загалом, архітектурні підходи, застосовані у цьому проєкті, спрямовані на досягнення високої модульності, гнучкості та розширюваності системи.

Висновки до розділу 3

Для розробки інформаційної системи було обрано середовища розробки Unity Engine. Основною платформою для поширення буде Windows з можливістю подальшого розширення на Web. Для контролю версій буде використано GitHub. За якість написання коду буде відповідати .NET Framework. Фреймворки Unity UI Toolkit та TextMeshPro вибрані для роботи з інтерфейсом.

Структурна схема програми наведена на кресленику ІК13.160БАК.006 Д1.

					ІК13.160БАК.006 ПЗ	Арк.
						25
Зм.	Лист	№ докум.	Підпис	Дата		

4 РОЗРОБКА ДІАГРАМИ ПРЕЦЕДЕНТІВ

Діаграма прецедентів наведена у кресленику ІК13.160БАК.006 Д2.

У програмі передбачено два рівні доступу: Користувач та Адміністратор, кожен із яких має свій набір функцій. Реалізовано повноцінні механізми аутентифікації та авторизації, які забезпечують контрольований доступ користувачів до функціоналу програми відповідно до їхніх прав. Аутентифікація здійснюється шляхом введення унікальних облікових даних, які перевіряються на відповідність інформації в таблиці користувачів бази даних, де паролі зберігаються у вигляді криптографічно захищених хешів. У разі успішної аутентифікації система переходить до етапу авторизації, визначаючи роль користувача — «Користувач» або «Модератор» — та надаючи відповідний набір функціональних можливостей. Такий підхід гарантує, що кожен користувач працює лише в межах дозволених йому дій, а спроби доступу до функцій вищого рівня блокуються системою без шкоди для загальної роботи застосунку.

Користувач після входу до системи отримує доступ до перегляду технічних характеристик компонентів та розрахунку ключових параметрів конфігурацій. Тауож йому надано право редагування даних у локальній базі, що дозволяє змінювати або зберігати альтернативні конфігурації та компоненти в рамках власного робочого середовища. Однак усі спроби додавання нового компоненту чи модифікації на глобальному рівні супроводжуються формуванням запиту на внесення змін. Такий запит передається модератору для подальшого підтвердження, редагування або відхилення, що дозволяє поєднувати гнучкість користувача із централізованим контролем даних.

Модератор має більш з розширені права доступу. Він має доступ до всіх можливостей звичайного користувача, включаючи перегляд компонентів, формування конфігурацій дронів та виконання розрахунків.

Окремим функціоналом модератора є обробка запитів на зміну глобальної бази даних, які надходять від користувачів. Кожен запит містить пропозицію на

					ІК13.160БАК.006 ПЗ	Арк.
						26
Зм.	Лист	№ докум.	Підпис	Дата		

додавання нового компоненту, зміну параметрів або виправлення неточностей. Модератор переглядає зміст запиту, аналізує запропоновані зміни та приймає рішення — затвердити або відхилити. Після затвердження зміни автоматично вносяться до глобальної бази та стають доступними для всіх користувачів системи. Таким чином, модератор виступає посередником між пропозиціями користувачів і централізованими даними, підтримуючи баланс між відкритістю системи та її структурованістю.

Крім розгляду запитів, модератор може самостійно ініціювати оновлення або видалення компонентів, які втратили актуальність, або не відповідають технічним вимогам. Для зручності модератора передбачено інтерфейс керування базою даних із фільтрами, сортуванням і можливістю швидкого редагування параметрів. Всі дії модератора фіксуються в системному журналі, що дозволяє відстежувати історію змін і забезпечує прозорість адміністрування. Такий підхід гарантує не лише оперативність оновлення даних, а й збереження їхньої цілісності та достовірності в умовах постійного розвитку системи.

5 РОЗРОБКА БАЗИ ДАНИХ

Діаграма таблиць та їх взаємопов'язаності у базі даних описана у кресленику ІК13.160БАК.006 ДЗ.

База даних складається з наступних таблиць: Motors (Мотори), Propellers (Пропелери), DroneFrames (Рами дронів), FlightControllers (Польотні контролери), Receiver (Приймачі), VideoTranslator (Відеопередавачі), Camera (Камери).

Таблиця Motors містить характеристики двигунів, а саме:

- ID - унікальний ідентифікатор мотора;
- Name - назва моделі мотора;
- Price - ціна мотора;
- StatorSize - розмір статора;
- KV - коефіцієнт обертів на вольт;
- OperatingVoltage - робоча напруга;
- MaxCurrent - максимально допустимий струм;
- MountSize - розмір кріплення для установки на раму;
- PropellerMount - тип або стандарт кріплення пропелера;
- PurchaseLink - посилання на сторінку з покупкою.

Таблиця Propellers параметри пропелерів, які встановлюються на мотори. У цій таблиці є поля:

- ID – унікальний ідентифікатор пропелера;
- Name – назва моделі;
- Price – вартість;
- Diameter – діаметр пропелера в дюймах;
- Pitch – крок лопаті (визначає, скільки повітря проходить за один оберт);
- BladeCount – кількість лопатей;
- MountType – тип кріплення до мотора;
- Construction – матеріал або тип виготовлення (наприклад, вуглепластик);
- PurchaseLink – посилання на товар.

					ІК13.160БАК.006 ПЗ	Арк.
						28
Зм.	Лист	№ докум.	Підпис	Дата		

Таблиця DroneFrames містить параметри каркасів для дронів та має наступні поля:

- ID – унікальний ідентифікатор рами;
- Model – назва моделі;
- Price – ціна;
- PropellerSize – максимальний діаметр пропелера, який поміщається в рамі;
- MotorSpacing – відстань між осями моторів;
- MotorMountSize – розмір отворів під мотори;
- ElectronicsMount – розмір монтажного отвору під польотний контролер;
- CameraMount – тип/розмір кріплення під камеру;
- BeamPlateThickness – товщина пластин рами;
- Material – матеріал конструкції;
- Weight – маса рами в грамах;
- PurchaseLink – посилання на покупку.

Таблиця FlightControllers описує дані польотних контролерів та містить наступні поля:

- ID – унікальний ідентифікатор;
- Name – назва контролера;
- Price – ціна;
- MountSize – розмір монтажних отворів;
- Processor – тип мікроконтролера;
- UARTPorts – кількість UART-портів (для підключення модулів);
- PWMOutputs – кількість PWM-виходів (для моторів);
- MotorOutputs – кількість виходів під двигуни;
- Barometer – логічне поле, що вказує на наявність барометра (1 — є, 0 — немає);
- Wireless – наявність бездротового з'єднання (Wi-Fi, Bluetooth);
- PurchaseLink – посилання на товар.

					ІК13.160БАК.006 ПЗ	Арк.
						29
Зм.	Лист	№ докум.	Підпис	Дата		

Таблиця Receiver містить характеристики приймачів сигналу. Вона складається з наступних полів:

- ID – унікальний ідентифікатор;
- Name – назва моделі;
- Price – вартість;
- Connection – тип сигналу (наприклад, PWM, SBUS, CRSF);
- Frequency – частота роботи;
- Power – потужність передавання;
- BodySize – габарити корпусу;
- AntennaLength – довжина антени;
- Polarization – тип поляризації (лінійна, кругова);
- AntennaConnector – тип роз'єму антени;
- PurchaseLink – посилання на покупку.

Таблиця VideoTranslator відповідає за передачу відеосигналу з камери до наземної станції. Ця таблиця складається з наступних полів:

- ID – унікальний ідентифікатор;
- Name – назва передавача;
- Price – вартість;
- Connection – тип підключення (аналоговий, цифровий);
- Frequency – діапазон частот;
- Power – потужність передавання;
- BodySize – розміри корпусу;
- AntennaLength – довжина антени;
- Polarization – тип поляризації сигналу;
- AntennaConnector – роз'єм антени;
- PurchaseLink – посилання на покупку.

Таблиця Camera зберігає параметри камер, що використовуються на роботизованій платформі. Ця таблиця складається з наступних полів:

- ID – унікальний ідентифікатор;

					ІК13.160БАК.006 ПЗ	Арк.
						30
Зм.	Лист	№ докум.	Підпис	Дата		

- Name – назва моделі;
- Price – ціна;
- Connection – тип відеовиходу (аналоговий / цифровий);
- Resolution – роздільна здатність (наприклад, 720p, 1080p)
- FieldOfView – кут огляду в градусах.;
- BodySize – габарити корпусу;
- Sensitivity – світлочутливість у люксах;
- PurchaseLink – посилання на товар.

Створена база даних має гнучку структуру, що дозволяє легко її розширювати для інших роботизованих платформ, зокрема наземних автономних роботів, дронів для промислового моніторингу та розумних автоматизованих систем.

					ІК13.160БАК.006 ПЗ	Арк.
						31
Зм.	Лист	№ докум.	Підпис	Дата		

6 РОЗРОБКА СТРУКТУРИ ІНФОРМАЦІЙНОЇ СИСТЕМИ

6.1 Архітектура інформаційної системи

Було визначено п'ять рівнів архітектури інформаційної системи:

- користувацький інтерфейс;
- основний логічний рівень;
- рівень підключення до бази даних;
- рівень збереження даних;
- контроль версій.

Користувацький інтерфейс відповідає за взаємодію користувача із системою, відображення вхідних та вихідних даних, а також навігацію по функціоналу програми.

Основний логічний рівень відповідає за обробку запитів, реалізацію функціональних сценаріїв системи, взаємодію між інтерфейсом та сховищами даних.

Рівень підключення до бази даних забезпечує передачу структурованих запитів від логіки до бази даних і назад.

Рівень збереження даних відповідає за збереження інформації системи. Він включає використання класичної реляційної бази даних для основних структурованих даних, а також резервного збереження у вигляді JSON файлу.

Контроль версій забезпечує відстеження змін у програмному коді, підтримку історії змін і можливість командної розробки.

Повна архітектурна схема інформаційної системи зображена на кресленнику ІК13.160БАК.006 Д4.

6.2 Інтеграція бази даних до інформаційної системи

Згідно з архітектурною моделлю MVC, було створено C# класи, що реалізують компонент Модель (Model). Вони відповідають за представлення сутностей реляційної бази даних та забезпечують взаємозв'язок між об'єктно-орієнтованою бізнес-логікою застосунку та даними, що зберігаються у SQL Server. Ці моделі містять

					ІК13.160БАК.006 ПЗ	Арк.
						32
Зм.	Лист	№ докум.	Підпис	Дата		

поля, що відображають структуру таблиць бази даних, і слугують основою для обміну даними між рівнями застосунку.

На рисунку 6.1 представлено клас, що представляє сутність даних таблиці «Camera» з реляційної бази даних SQL Server.

```
public class Camera
{
    public int ID { get; set; }
    public string Name { get; set; }
    public int Price { get; set; }
    public string Connection { get; set; }
    public string Resolution { get; set; }
    public int FieldOfView { get; set; }
    public string BodySize { get; set; }
    public double SensitivityLux { get; set; }
    public string Link { get; set; }

    public string GetInfo()
    {
        return $"Name: {Name}\n" +
            $"Price: {Price} грн\n" +
            $"Connection: {Connection}\n" +
            $"Resolution: {Resolution}\n" +
            $"Field of View: {FieldOfView}°\n" +
            $"Body Size: {BodySize}\n" +
            $"Sensitivity: {SensitivityLux} Lux\n" +
            $"Purchase Link: {Link}";
    }
}
```

Рисунок 6.1 – Клас, що представляє сутність таблиці «Camera»

Рисунок 6.2 демонструє клас, що представляє сутність даних таблиці «DroneFrame» з реляційної бази даних SQL Server.

Рисунок 6.3 демонструє клас, що представляє сутність даних таблиці «FlightController» з реляційної бази даних SQL Server.

Рисунок 6.4 демонструє клас, що представляє сутність даних таблиці «Motor» з реляційної бази даних SQL Server.

Рисунок 6.5 демонструє клас, що представляє сутність даних таблиці «Propeller» з реляційної бази даних SQL Server.

Рисунок 6.6 демонструє клас, що представляє сутність даних таблиці «Receiver» з реляційної бази даних SQL Server.

```

public class DroneFrame
{
    public int ID { get; set; }
    public string Model { get; set; }
    public string Price { get; set; }
    public string PropellerSize { get; set; }
    public string MotorSpacing { get; set; }
    public string MotorMountSize { get; set; }
    public string ElectronicsMount { get; set; }
    public string CameraMount { get; set; }
    public string BeamPlateThickness { get; set; }
    public string Material { get; set; }
    public string Weight { get; set; }
    public string PurchaseLink { get; set; }

    public string GetInfo()
    {
        return $"Model: {Model}\n" +
            $"Price: {Price}\n" +
            $"Propeller Size: {PropellerSize}\n" +
            $"Motor Spacing: {MotorSpacing}\n" +
            $"Motor Mount Size: {MotorMountSize}\n" +
            $"Electronics Mount: {ElectronicsMount}\n" +
            $"Camera Mount: {CameraMount}\n" +
            $"Beam Plate Thickness: {BeamPlateThickness}\n" +
            $"Material: {Material}\n" +
            $"Weight: {Weight}\n" +
            $"Purchase Link: {PurchaseLink}";
    }
}

```

Рисунок 6.2 – Клас, що представляє сутність даних таблиці «Camera»

```

public class FlightController
{
    public int ID { get; set; }
    public string Name { get; set; }
    public int Price { get; set; }
    public string MountSize { get; set; }
    public string Processor { get; set; }
    public int UARTPorts { get; set; }
    public int PWMOutputs { get; set; }
    public int MotorOutputs { get; set; }
    public bool Barometer { get; set; }
    public string Wireless { get; set; }
    public string PurchaseLink { get; set; }

    public string GetInfo()
    {
        return $"Name: {Name}\n" +
            $"Price: {Price} грн\n" +
            $"Mount Size: {MountSize}\n" +
            $"Processor: {Processor}\n" +
            $"UART Ports: {UARTPorts}\n" +
            $"PWM Outputs: {PWMOutputs}\n" +
            $"Motor Outputs: {MotorOutputs}\n" +
            $"Barometer: {(Barometer ? "Yes" : "No")}\n" +
            $"Wireless: {Wireless}\n" +
            $"Purchase Link: {PurchaseLink}";
    }
}

```

Рисунок 6.3 – клас, що представляє сутність даних таблиці «FlightController»

```

public class Motor
{
    public int ID { get; set; }
    public string Name { get; set; }
    public int Price { get; set; }
    public int StatorSize { get; set; }
    public int KV { get; set; }
    public double OperatingVoltage { get; set; }
    public int MaxThrust { get; set; }
    public int MaxCurrent { get; set; }
    public int MountSize { get; set; }
    public string PropellerMount { get; set; }
    public string PurchaseLink { get; set; }

    public string GetInfo()
    {
        return $"Name: {Name}\n" +
            $"Price: {Price} грн\n" +
            $"Stator Size: {StatorSize} мм\n" +
            $"KV: {KV}\n" +
            $"Operating Voltage: {OperatingVoltage} V\n" +
            $"Max Thrust: {MaxThrust} г\n" +
            $"Max Current: {MaxCurrent} А\n" +
            $"Mount Size: {MountSize} мм\n" +
            $"Propeller Mount: {PropellerMount}\n" +
            $"Purchase Link: {PurchaseLink}";
    }
}

```

Рисунок 6.4 – Клас, що представляє сутність даних таблиці «Motor»

```

public class Propeller
{
    public int ID { get; set; }
    public string Name { get; set; }
    public int Price { get; set; }
    public double Diameter { get; set; }
    public double Pitch { get; set; }
    public int BladeCount { get; set; }
    public string MountType { get; set; }
    public string Construction { get; set; }
    public string Link { get; set; }

    public string GetInfo()
    {
        return $"Name: {Name}\n" +
            $"Price: {Price} грн\n" +
            $"Diameter: {Diameter} см\n" +
            $"Pitch: {Pitch} см\n" +
            $"Blade Count: {BladeCount}\n" +
            $"Mount Type: {MountType}\n" +
            $"Construction: {Construction}\n" +
            $"Link: {Link}";
    }
}

```

Рисунок 6.5 – Клас, що представляє сутність даних таблиці «Propeller»

Рисунок 6.7 демонструє клас, що представляє сутність даних таблиці «VideoTranslator» з реляційної бази даних SQL Server.

```

public class Receiver
{
    public int ID { get; set; }
    public string Name { get; set; }
    public int Price { get; set; }
    public string CommunicationProtocol { get; set; }
    public string Frequency { get; set; }
    public int TelemetryPower { get; set; }
    public string ReceiverSize { get; set; }
    public string Link { get; set; }

    public string GetInfo()
    {
        return $"Name: {Name}\n" +
            $"Price: {Price} грн\n" +
            $"Communication Protocol: {CommunicationProtocol}\n" +
            $"Frequency: {Frequency}\n" +
            $"Telemetry Power: {TelemetryPower} mW\n" +
            $"Receiver Size: {ReceiverSize}\n" +
            $"Purchase Link: {Link}";
    }
}

```

Рисунок 6.6 – Клас, що представляє сутність даних таблиці «Receiver»

```

public class VideoTranslator
{
    public int ID { get; set; }
    public string Name { get; set; }
    public int Price { get; set; }
    public string Connection { get; set; }
    public string Frequency { get; set; }
    public int Power { get; set; }
    public string BodySize { get; set; }
    public int AntennaLength { get; set; }
    public string Polarization { get; set; }
    public string AntennaConnector { get; set; }
    public string Link { get; set; }

    public string GetInfo()
    {
        return $"Name: {Name}\n" +
            $"Price: {Price} грн\n" +
            $"Connection: {Connection}\n" +
            $"Frequency: {Frequency}\n" +
            $"Power: {Power} mW\n" +
            $"Body Size: {BodySize}\n" +
            $"Antenna Length: {AntennaLength} мм\n" +
            $"Polarization: {Polarization}\n" +
            $"Antenna Connector: {AntennaConnector}\n" +
            $"Purchase Link: {Link}";
    }
}

```

Рисунок 6.7 – Клас, що представляє сутність даних таблиці «VideoTranslator»

6.3 Реалізація CRUD-функціоналу

Основна концепція CRUD-функціоналу полягає у чотирьох базових операціях: створення (Create), читання (Read), оновлення (Update) та видалення (Delete). Для реалізації цих операцій використовуються спеціалізовані методи, які в свою чергу звертаються до SQL Server через класичні SQL-запити.

Операція читання (Read) здійснюється через виконання SQL-запиту типу SELECT, завдяки якому дані з відповідної таблиці зчитуються та перетворюються у список об'єктів, що представляють конкретні елементи дрона. Після читання даних з таблиці вони конвертуються у формат, вказаний у певному класі. Приклад імплементації даної операції для таблиці «Motor» наведена на рисунку 6.8. Для читання інших таблиць використовуються схожі методи з іншим SQL-запитом та конвертацією у інший формат даних.

```
public void ReadMotors()
{
    MotorList.Clear();
    using (SqlConnection connection = new SqlConnection(connectionString))
    {
        try
        {
            connection.Open();
            string query = "SELECT ID, Name, Price, StatorSize, KV, OperatingVoltage, MaxThrust, " +
                           "MaxCurrent, MountSize, PropellerMount, PurchaseLink FROM dbo.Motors";

            using (SqlCommand command = new SqlCommand(query, connection))
            using (SqlDataReader reader = command.ExecuteReader())
            {
                while (reader.Read())
                {
                    Motor tempMotor = new Motor
                    {
                        ID = reader.GetInt32(0),
                        Name = reader.GetString(1),
                        Price = reader.GetInt32(2),
                        StatorSize = reader.GetInt32(3),
                        KV = reader.GetInt32(4),
                        OperatingVoltage = reader.GetDouble(5),
                        MaxThrust = reader.GetInt32(6),
                        MaxCurrent = reader.GetInt32(7),
                        MountSize = reader.GetInt32(8),
                        PropellerMount = reader.GetString(9),
                        PurchaseLink = reader.GetString(10)
                    };
                    MotorList.Add(tempMotor);
                }
            }
        }
        catch (SqlException ex)
        {
            Debug.LogError($"*SQL помилка: {ex.Message}*");
        }
    }

    MotorList = MotorList.OrderByDescending(camera => camera.Price).ToList();
    if (motorTableUI != null) motorTableUI.FillMotorTable(MotorList);
}
```

Рисунок 6.8 – Приклад імплементації операції читання елементів у для таблиці «Motor»

Операція створення (Create) реалізована через SQL-запит типу INSERT, який дозволяє додати новий запис до таблиці. Для виконання даної операції користувач

повинен внести дані до певного компоненту та натиснути кнопку, яка б викликала метод додавання цього компонента до бази даних. Власне метод створення має приймати об'єкт певного типу як вхідний параметр, параметризує його у SQL-запиті для зменшення ризику SQL-ін'єкцій. Приклад імплементації методу для таблиці «Motor» наведено на рисунку 6.9. Для методів створення елементів інших таблиць було створено схожі методи з іншими SQL-запитами та вхідними даними.

```
public void AddMotor(Motor motor)
{
    using (SqlConnection connection = new SqlConnection(connectionString))
    {
        try
        {
            connection.Open();
            string query = "INSERT INTO dbo.Motors (Name, Price, StatorSize, KV," +
                "OperatingVoltage, MaxThrust, MaxCurrent, MountSize, PropellerMount, PurchaseLink) " +
                "VALUES (@Name, @Price, @StatorSize, @KV, @OperatingVoltage" +
                ", @MaxThrust, @MaxCurrent, @MountSize, @PropellerMount, @PurchaseLink)";

            using (SqlCommand command = new SqlCommand(query, connection))
            {
                command.Parameters.AddWithValue("@Name", motor.Name);
                command.Parameters.AddWithValue("@Price", motor.Price);
                command.Parameters.AddWithValue("@StatorSize", motor.StatorSize);
                command.Parameters.AddWithValue("@KV", motor.KV);
                command.Parameters.AddWithValue("@OperatingVoltage", motor.OperatingVoltage);
                command.Parameters.AddWithValue("@MaxThrust", motor.MaxThrust);
                command.Parameters.AddWithValue("@MaxCurrent", motor.MaxCurrent);
                command.Parameters.AddWithValue("@MountSize", motor.MountSize);
                command.Parameters.AddWithValue("@PropellerMount", motor.PropellerMount);
                command.Parameters.AddWithValue("@PurchaseLink", motor.PurchaseLink);

                command.ExecuteNonQuery();
                Debug.Log("Мотор успішно додано!");
            }
        }
        catch (SqlException ex)
        {
            Debug.LogError($"SQL помилка: {ex.Message}");
        }
    }
}
```

Рисунок 6.9 – Приклад імплементації методу створення

Операція оновлення (Update) дозволяє модифікувати існуючі записи. Використовуючи SQL-запити типу UPDATE, система проводить зміну полів конкретного елемента, що визначається його унікальним ідентифікатором. Для виконання даної операції користувач повинен внести оновлені дані до певного компоненту та натиснути кнопку, яка б викликала метод оновлення компонента у базі даних. Власне метод створення має приймати об'єкт певного типу та його індекс у базі

даних як вхідні параметри. Приклад імплементації методу оновлення для таблиці «Motor» наведений на рисунку 6.10. Для методів оновлення елементів інших таблиць було створено схожі методи з іншими SQL-запитами та вхідними даними.

```
public void UpdateMotor(int motorID, Motor updatedMotor)
{
    using (SqlConnection connection = new SqlConnection(connectionString))
    {
        try
        {
            connection.Open();
            string query = "UPDATE dbo.Motors SET Name = @Name, Price = @Price, StatorSize = @StatorSize, KV = @KV, " +
                "OperatingVoltage = @OperatingVoltage, MaxThrust = @MaxThrust, MaxCurrent = @MaxCurrent, " +
                "MountSize = @MountSize, PropellerMount = @PropellerMount, PurchaseLink = @PurchaseLink WHERE ID = @ID";

            using (SqlCommand command = new SqlCommand(query, connection))
            {
                command.Parameters.AddWithValue("@ID", motorID);
                command.Parameters.AddWithValue("@Name", updatedMotor.Name);
                command.Parameters.AddWithValue("@Price", updatedMotor.Price);
                command.Parameters.AddWithValue("@StatorSize", updatedMotor.StatorSize);
                command.Parameters.AddWithValue("@KV", updatedMotor.KV);
                command.Parameters.AddWithValue("@OperatingVoltage", updatedMotor.OperatingVoltage);
                command.Parameters.AddWithValue("@MaxThrust", updatedMotor.MaxThrust);
                command.Parameters.AddWithValue("@MaxCurrent", updatedMotor.MaxCurrent);
                command.Parameters.AddWithValue("@MountSize", updatedMotor.MountSize);
                command.Parameters.AddWithValue("@PropellerMount", updatedMotor.PropellerMount);
                command.Parameters.AddWithValue("@PurchaseLink", updatedMotor.PurchaseLink);

                int rowsAffected = command.ExecuteNonQuery();
                if (rowsAffected > 0)
                {
                    Debug.Log($"Motor with ID {motorID} successfully updated!");
                }
                else
                {
                    Debug.LogWarning($"No motor found with ID {motorID}. Ensure the ID is valid.");
                }
            }
        }
        catch (SqlException ex)
        {
            Debug.LogError($"SQL помилка: {ex.Message}");
        }
    }
}
```

Рисунок 6.10 – Приклад імплементації методу оновлення

Остання операція - видалення (Delete) - здійснюється через SQL-запити типу DELETE, що дозволяють повністю видалити запис із таблиці за заданим ідентифікатором. Для виконання даної операції користувач повинен вибрати компонент та натиснути кнопку, яка б викликала метод його видалення у базі даних. Власне метод видалення має приймати індекс певного об'єкту у базі даних як вхідний параметр. Приклад імплементації методу видалення для таблиці «Motor» наведений на рисунку 6.11. Для методів видалення елементів інших таблиць було створено схожі методи з іншими SQL-запитами.

```

public void DeleteMotor(int motorID)
{
    using (SqlConnection connection = new SqlConnection(connectionString))
    {
        try
        {
            connection.Open();
            string query = "DELETE FROM dbo.Motors WHERE ID = @ID";

            using (SqlCommand command = new SqlCommand(query, connection))
            {
                command.Parameters.AddWithValue("@ID", motorID);

                int rowsAffected = command.ExecuteNonQuery();
                if (rowsAffected > 0)
                {
                    Debug.Log($"Motor with ID {motorID} successfully deleted!");
                }
                else
                {
                    Debug.LogWarning($"No motor found with ID {motorID}. Ensure the ID exists.");
                }
            }
        }
        catch (SqlException ex)
        {
            Debug.LogError($"SQL помилка: {ex.Message}");
        }
    }
}

```

Рисунок 6.11 – Приклад імплементації методу видалення

Загалом усі вищенаведені коди складають клас MotorsReader, який відповідає за повну роботу з таблицею «Motor». Загалом було написано сім таких класів – по одному на кожну таблицю бази даних.

Всі операції виконуються в рамках встановленого з'єднання з базою даних. Для кожної CRUD-операції використовується блок try-catch, який контролює можливі винятки, що можуть виникнути під час виконання запитів, і негайно інформує розробника або користувача про проблему через системні повідомлення у консолі середовища роботи. Такий підхід гарантує стабільність роботи всієї системи, а також дозволяє оперативно реагувати на збої в з'єднанні або помилки в SQL-запитах.

6.5 Отримання конфігурації дрону

У методі GetDroneConfiguration(double requiredFlightRange, double requiredFlightTime, double requiredAdditionalPayload, int maxCost) виконується послідовний перебір усіх можливих комбінацій рами, моторів, пропелерів, контролерів,

приймачів, камер та відеопередавачів з подальшою перевіркою їхньої сумісності, технічних характеристик і вартості.

Спочатку задаються фіксовані параметри: кількість моторів (motorCount), вага одного мотора (motorWeight), вага батареї (batteryWeight), вага відеопередавача (videoTranslatorWeight), середня швидкість польоту (avgSpeed), коефіцієнт для оцінки часу польоту (kFlightTime) та безпековий коефіцієнт safetyFactor.

Для кожної рами зчитується її власна вага (frameWeight), розмір монтажних отворів (frameMountSize) і розмір рекомендованого пропелера (framePropSize). Мотор вважається сумісним із рамою лише у випадку точного співпадіння розміру монтажного фланця.

Коли рамка і мотор вибрані, обчислюється базова вага конструкції за формулою 6.1 і загальна вага з урахуванням відеопередавача за формулою 6.2:

$$\text{baseWeight} = \text{frameWeight} + \text{motorCount} * \text{motorWeight} + \text{batteryWeight} \quad (6.1)$$

$$\text{totalWeight} = \text{baseWeight} + \text{videoTranslatorWeight} \quad (6.2)$$

Сумарна тяга всіх моторів розраховується за формулою 6.3

$$\text{totalThrust} = \text{motorCount} * \text{motor.MaxThrust} \quad (6.3)$$

Перевіряється умова безпеки: $\text{totalThrust} \geq \text{safetyFactor} \times \text{totalWeight}$. Якщо тяга достатня, оцінюється час польоту (в хвилинах) за формулою 6.4, дальність (в кілометрах) за формулою 6.5 та додаткова вантажопідйомність (в грамах) за формулою 6.6.

$$\text{estimatedFlightTime} = \text{kFlightTime} * (\text{totalThrust} / \text{totalWeight}) \quad (6.4)$$

$$\text{estimatedFlightRange} = (\text{avgSpeed} * \text{estimatedFlightTime}) / 60 \quad (6.5)$$

					ІК13.160БАК.006 ПЗ	Арк.
						41
Зм.	Лист	№ докум.	Підпис	Дата		

$$\text{additionalPayloadCapacity} = (\text{totalThrust} / \text{safetyFactor}) - \text{totalWeight} \quad (6.6)$$

Ці величини порівнюються з вимогами `requiredFlightTime`, `requiredFlightRange` і `requiredAdditionalPayload`. Якщо всі умови виконано, програма перебирає усі доступні комбінації контролерів, приймачів, камер і відеопередавачів та для кожної обчислює повну вартість за формулою 6.7

$$\text{totalCost} = \text{framePrice} + \text{motor.Price} * \text{motorCount} + \text{prop.Price} + \text{cont.Price} + \text{rec.Price} + \text{cam.Price} + \text{vt.Price}. \quad (6.7)$$

Якщо `totalCost` не перевищує `maxCost`, створюється об'єкт `DroneConfiguration`, куди записуються вибрані компоненти, `totalWeight`, `totalCost`, `estimatedFlightTime`, `estimatedFlightRange` і `additionalPayloadCapacity`. Цей об'єкт повертається із методу й відображається у UI. Якщо жодна комбінація не задовольняє усіх критеріїв, метод повертає `null` і виводить повідомлення про відсутність конфігурацій, що відповідають вимогам.

Усі коди та їх взаємодію можна побачити на кресленику ІК13.160БАК.006 Д5.

6.6 Розробка інтерфейсу програми

6.6.1 Розробка інтерфейсу перегляду компонентів

У реалізації інтерфейсу таблиць для кожного компоненту використано наступні класи: `UIRow`, `TableUI`. Усі класи мають аналогічну структуру.

Приклад імплементації `UIRow` для таблиці `Camera` наведений на рисунку 6.12. Він відповідає за відображення кожного рядка таблиці камер: інкапсулює набір полів `TextMeshProUGUI` для даних камери (назва, ціна, роздільна здатність, кут огляду, тип підключення, розміри корпусу, світлочутливість, посилання) і в методі

Setup(Camera cam) заповнює їх даними з переданого об'єкта Camera.

```
public class CameraUIRow : MonoBehaviour
{
    public TextMeshProUGUI nameText;
    public TextMeshProUGUI priceText;
    public TextMeshProUGUI resolutionText;
    public TextMeshProUGUI fovText;
    public TextMeshProUGUI connectionText;
    public TextMeshProUGUI bodySizeText;
    public TextMeshProUGUI sensitivityText;
    public TextMeshProUGUI linkText;

    public void Setup(Camera cam)
    {
        nameText.text = cam.Name;
        priceText.text = cam.Price.ToString();
        resolutionText.text = cam.Resolution;
        fovText.text = cam.FieldOfView + "°";
        connectionText.text = cam.Connection;
        bodySizeText.text = cam.BodySize;
        sensitivityText.text = cam.SensitivityLux + " Lux";
        linkText.text = cam.Link;
    }
}
```

Рисунок 6.12 – Приклад імплементації класу UIRow для таблиці Camera

Клас CameraTableUI відповідає за формування та управління всією таблицею камер: у методі Start() він знаходить CameraReader, відключає зайві UI-панелі й налаштовує обробники подій для створення, оновлення та видалення; метод FillCameraTable(List<Camera>) створює екземпляри CameraUIRow, накопичує їх для очищення та наповнює два випадуючі списки для оновлення й видалення елементами з отриманого списку камер; панелі створення, оновлення та видалення активуються відповідними методами (ActivateCreationUI, ActivateUpdateUI, ActivateDeletionUI), а після кожної операції дані перераховуються (cameraReader.ReadCameras()) і таблиця оновлюється.

6.6.2 Розробка інтерфейсу отримання конфігурації дрону

В інтерфейсі для отримання конфігурації дрона виділено три ключові блоки.

Перший блок призначений для введення даних. Тут користувач задає всі необхідні параметри: бажану дальність та час польоту, обсяг додаткового навантаження й максимально доступний бюджет. Після введення цих значень система переходить до підбору оптимальних компонентів.

Другий блок відповідає за отримання більш детальної інформації про кожен обраний елемент. Після того, як алгоритм знаходить потенційні комбінації, користувач може переглядати окремі компоненти — раму, мотори, пропелери, контролер, приймач, камеру та відеопередавач. Для кожного з них передбачено відображення технічних характеристик і опису, щоб мати змогу глибше ознайомитися з особливостями та сумісністю.

Третій блок відображає результати підбору – готові характеристики дрона. Тут з'являється загальна вага конструкції, загальна вартість, розрахована дальність польоту, очікуваний час у повітрі та резерв вантажопідйомності. Усі ці дані подані у вигляді зручного текстового звіту, який одразу доступний для перегляду й подальшого прийняття рішення.

Схему вигляду інтерфейсу отримання конфігурації дрону до заповнення даними наведено на рисунку 6.13

Рисунок 6.13 – Вигляд інтерфейсу отримання конфігурації дронів

6.6.3 Перехід між інтерфейсами

Для переходу між інтерфейсами було створено додатковий метод, зображений на рисунку 6.14 що дозволяє приховувати або відображати потрібний екран.

```
public class SceneSwitcher : MonoBehaviour
{
    public GameObject scene1;

    public void SwitchScene(bool OpenFirstScene)
    {
        // Перемикаємо активний GameObject
        if (!OpenFirstScene)
        {
            scene1.SetActive(false);
        }
        else
        {
            scene1.SetActive(true);
        }
    }
}
```

Рисунок 6.14 – Метод для переходу між інтерфейсами

Він містить поле, яке посилається на перший інтерфейсний блок. Коли викликається метод SwitchScene із логічним параметром, ця функція оцінює його значення і відповідно активує або вимикає цільовий об’єкт. Якщо передано true, інтерфейс стає видимим, якщо false – приховується.

Такий підхід працює саме тому, що один інтерфейс фактично розташовується поверх іншого — тому для оптимізації достатньо активувати лише один об’єкт у кожний момент часу.

6.7 Створення версії для поширення

Для підготовки програми до поширення створюють спеціальну збірку — виконуваний пакет для обраної платформи. Під час цього процесу середовище

компілює скрипти, упаковує ресурси та застосовує оптимізації, щоб надати кінцевому користувачу готовий до запуску продукт.

Існує кілька ключових мотивів для створення збірки програми:

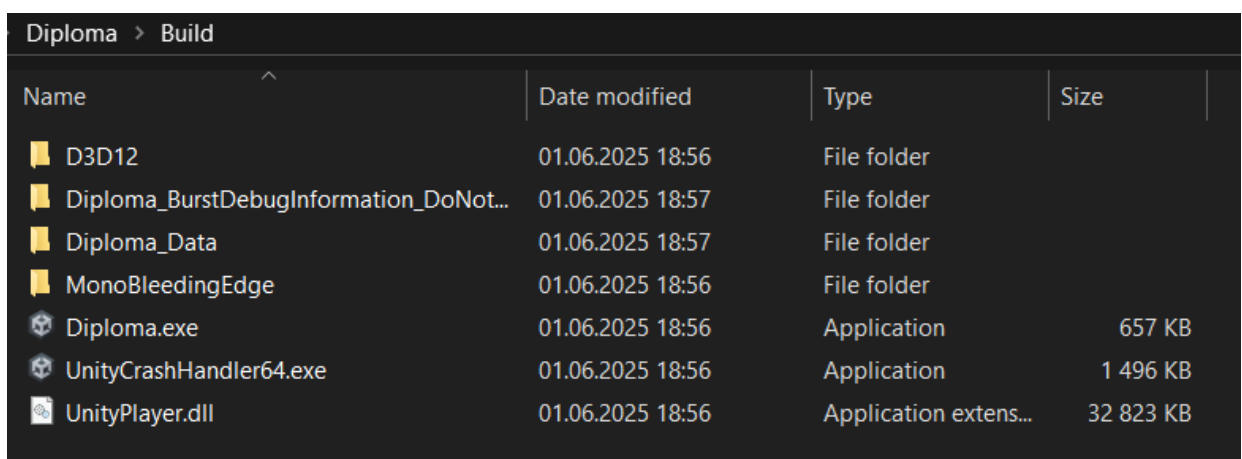
збірка гарантує сумісність із цільовою платформою — середовище перетворює код і ресурси у формат, який працює на конкретній операційній системі, без потреби додаткової ручної адаптації;

у процесі генерації виконуваного пакету відбувається оптимізація: вимикаються засоби відладки, проходять агресивніші кроки з мінімізації та компіляції, завдяки чому програма працює швидше та споживає менше ресурсів;

створення збірки підвищує безпеку коду — сценарії вбудовуються у бінарний виконуваний файл, що утруднює спроби зворотної інженерії та несанкціоновані зміни;

готовий виконуваний пакет легко розповсюджувати: його можна розмістити на веб-сервісі, в магазині додатків або передати замовнику, а також автоматизувати процес оновлень і доставки нових версій.

На рисунку 6.15 наведені файли були отримані після завершення процесу збірки програми для платформи Windows. Для запуску програми користувачу потрібно запустити файл з розширенням .exe.



Diploma > Build				
Name	Date modified	Type	Size	
D3D12	01.06.2025 18:56	File folder		
Diploma_BurstDebugInformation_DoNot...	01.06.2025 18:57	File folder		
Diploma_Data	01.06.2025 18:57	File folder		
MonoBleedingEdge	01.06.2025 18:56	File folder		
Diploma.exe	01.06.2025 18:56	Application	657 KB	
UnityCrashHandler64.exe	01.06.2025 18:56	Application	1 496 KB	
UnityPlayer.dll	01.06.2025 18:56	Application extens...	32 823 KB	

Рисунок 6.15 – Файли отримані після «збірки» програми

Висновки до розділу 6

Реалізовано повноцінний інтерфейс для роботи з компонентами й конфігурацією дрона. Розроблено табличний перегляд усіх доступних елементів із можливістю їх створення, редагування й видалення. Окремо виділено блок, де користувач вводить параметри, і блок, у якому відображаються остаточні характеристики згенерованого дрона. Для зручності навігації між різними екранами створено простий механізм перемикання між інтерфейсами. Обґрунтовано необхідність створення збірки програми для поширення.

Для візуалізації логіки взаємодії користувача з системою побудовано діаграму послідовності, що демонструє повний цикл роботи програми. Діаграму послідовності роботи з програмою наведена на кресленику ІК13.160БАК.006 Д6.

					ІК13.160БАК.006 ПЗ	Арк.
						47
Зм.	Лист	№ докум.	Підпис	Дата		

7 ТЕСТУВАННЯ СИСТЕМИ

7.1 Мета випробувань

Випробування покликані оцінити зручність користування програмою, перевірити реакцію системи на різноманітні сценарії поведінки користувача та забезпечити оптимальну взаємодію між логікою програми та базою даних. Для тестування швидкодії було модифіковано код, що б виводились логи часу витраченого для певної операції.

Основні завдання випробувань включають перевірку виконання всіх заявлених функцій, коректності роботи алгоритмів обробки даних і взаємодії з базою даних, а також правильність відображення інформації у користувацькому інтерфейсі. Тестуванню операцій CRUD, інтеграції компонентів, швидкодії, обробці помилок під час роботи з базою даних та перевірці сумісності роботи додатку на різних цільових платформах.

7.2 Тестування CRUD операцій

Під час тестування CRUD функціоналу було проведено перевірку всіх чотирьох основних операцій роботи з базою даних: створення, читання, оновлення та видалення записів.

7.2.1 Тестування операції читання таблиць компонентів

На рисунку 7.1 можна побачити результат виконання операції читання з бази даних з таблиці «Мотор».

Model	Price	Stator size	KV	Operating voltage	Max thrust	Max current	Mount size	Propeller mount	Link
Motor X1 Pro	800 грн	38 мм	2300 kV	11,1 V	500 г	30 A	20 мм	Standard	https://escadrone.com.ua/produkt/motor-x1-pro
RotorMax 2200	750 грн	36 мм	2200 kV	11,1 V	420 г	28 A	25 мм	Universal	https://dronehub.in.ua/produkt/rotormax-2200
TurboDrive 2500	900 грн	40 мм	2500 kV	11,1 V	600 г	32 A	20 мм	Compact	https://flymod.net/produkt/turbo-drive-2500
EcoSpin 2000	650 грн	34 мм	2000 kV	11,1 V	350 г	25 A	22 мм	Standard	https://escadrone.com.ua/produkt/ecospin-2000
PowerLift 2400	850 грн	38 мм	2400 kV	11,1 V	550 г	30 A	20 мм	Universal	https://dronehub.in.ua/produkt/powerlift-2400

Мотор ▼

Back to drone config

Add

Update

Delete

Рисунок 7.1 – Результат виконання операції читання з таблиці «Мотор»

Звіємо отриману таблицю з таблицею у базі даних, яку можна побачити на рисунку 7.2.

ID	Name	Price	StatorSize	KV	OperatingVoltage	MaxThrust	MaxCurrent	MountSize	PropellerMount	PurchaseLink
1	Motor X1 Pro	800	38	2300	11,1	500	30	20	Standard	https://escadrone.com.ua/produkt/motor-x1-pro
2	RotorMax 2200	750	36	2200	11,1	420	28	25	Universal	https://dronehub.in.ua/produkt/rotormax-2200
3	TurboDrive 2500	900	40	2500	11,1	600	32	20	Compact	https://flymod.net/produkt/turbo-drive-2500
4	EcoSpin 2000	650	34	2000	11,1	350	25	22	Standard	https://escadrone.com.ua/produkt/ecospin-2000
5	PowerLift 2400	850	38	2400	11,1	550	30	20	Universal	https://dronehub.in.ua/produkt/powerlift-2400

Рисунок 7.2 – Таблиця «Мотор» у базі даних

Отримані дані відображаються згідно з раніше визначеними форматом: для кожного запису інформація про модель, ціну, технічні параметри, що підтверджує правильність розподілу інформації у відповідних стовпцях таблиці.

На рисунку 7.3 видно лог системи, який показує кількість затраченого часу для виконання операції читання. З даного рисунку можемо зробити висновок, що час на операцію читання є мінімальним, оскільки було використано два різні методи перевірки часу, які повернули одне значення. Даний метод перевірки показує,

					ІК13.160БАК.006 ПЗ	Арк.
						49
Зм.	Лист	№ докум.	Підпис	Дата		

що оптимізовані SQL-запити дозволяють виконувати операції читання без затримок навіть при збільшенні бази даних.

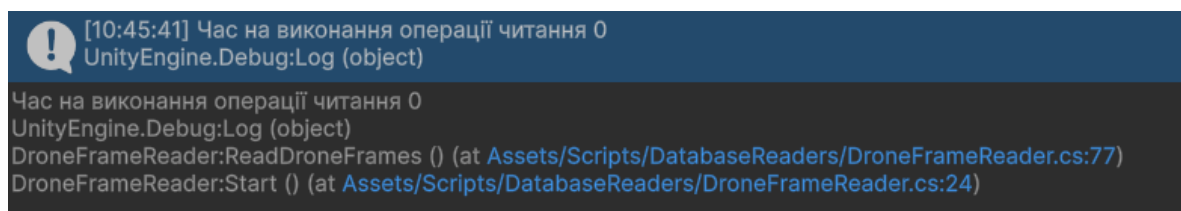


Рисунок 7.3 – Затрачений час на виконання операції читання

7.2.2 Тестування операції створення

При виконанні операції створення система не лише додає нові записи до бази даних, а й забезпечує повну валідацію та параметризацію даних на кожному етапі. Після введення інформації через інтерфейс користувача, показаний на рисунку 7.4, система автоматично перевіряє коректність заповнення полів (наприклад, відповідність числових параметрів формату integer чи float, а також правильність введення текстових даних).

Рисунок 7.4 – Інтерфейс додавання компоненту «Мотор»

Після натискання кнопки «Add» викликається спеціальна функція, яка запускає цикл валідації кожного введеного параметра. Якщо всі дані відповідають встановленим правилам, формується SQL-запит для вставки нового запису в таблицю. У цей момент система реєструє інформацію у логах, що підтверджує успішне виконання операції, а також негайно оновлює UI. Відповідна таблиця компонентів автоматично перезавантажується, і новий елемент з'являється серед уже існуючих записів, що забезпечує прозорість та оперативність роботи системи. Результат виконання операції наведений на рисунку 7.5.

Model	Price	Stator size	KV	Operating voltage	Max thrust	Max current	Mount size	Propeller mount	Link
TurboDrive 2500	900 грн	40 мм	2500 KV	11,1 V	600 г	32 A	20 мм	Compact	https://flymod.net/product/turbo-drive-2500
PowerLift 2400	850 грн	38 мм	2400 KV	11,1 V	550 г	30 A	20 мм	Universal	https://dronehub.in.ua/product/powerlift-2400
Motor X1 Pro	800 грн	38 мм	2300 KV	11,1 V	500 г	30 A	20 мм	Standard	https://escadrone.com.ua/product/motor-x1-pro
RotorMax 2200	750 грн	36 мм	2200 KV	11,1 V	420 г	28 A	25 мм	Universal	https://dronehub.in.ua/product/rotor-max-2200
EcoSpin 2000	650 грн	34 мм	2000 KV	11,1 V	350 г	25 A	22 мм	Standard	https://escadrone.com.ua/product/ecospin-2000
SomeTestModel	1 грн	2 мм	3 KV	4 V	5 г	6 A	8 мм	TestValue	SomeTestLink

Мотор ▾

Back to drone config

Add

Update

Delete

Рисунок 7.5 – Таблиця «Мотор» після операції додавання

Крім цього, система також підтримує ведення логів всіх операцій створення, що є важливим елементом для подальшого аналізу роботи додатку та виявлення потенційних проблем. Логування дозволяє відстежувати кожну успішну операцію створення, що позитивно позначається на можливості масштабування системи та її подальшій підтримці. Приклад логу наведений на рисунку 7.6.

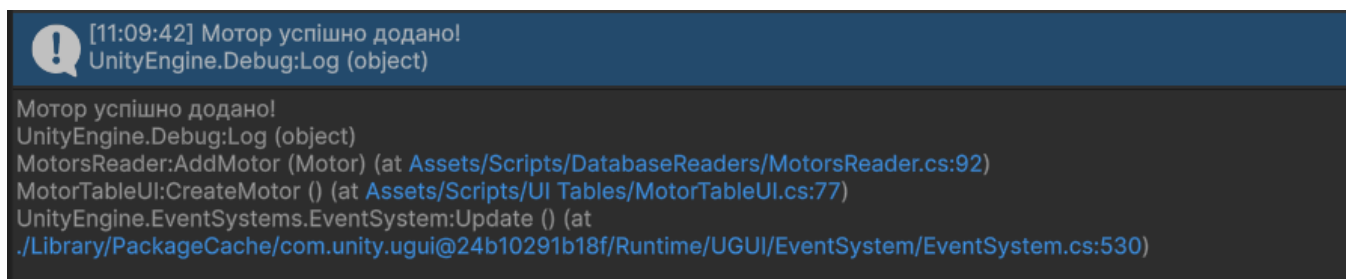


Рисунок 7.6 – Приклад логу операції додавання

7.2.3 Тестування операції оновлення

При тестуванні операції оновлення (Update) дані в уже існуючих записах відображаються миттєво, що свідчить про стабільну і ефективну роботу системи обробки даних. Приклад інтерфейсу функції оновлення наведений на рисунку 7.7.

Рисунок 7.7 –Інтерфейс функції оновлення компоненту

Відредагуємо дані мотору, створеного у попередньому пункті, як вказано на рисунку 7.8.

Model	Price	Stator size	KV	Operating voltage	Max thrust	Max current	Propeller Mount	Mount size	Link
TurboDrive 2500	900 грн	40 мм	2500 KV	11,1 V	600 г	32 A	20 мм	Compact	https://flymod.net/prod...turbo-drive-2500
PowerLift 2400	850 грн	38 мм	2400 KV	11,1 V	550 г	30 A	20 мм	Universal	https://dronehub.in.ua/product/powerlift-2400
Motor X1 Pro	800 грн	38 мм	2300 KV	11,1 V	500 г	30 A	20 мм	Standard	https://escadrones.com.ua/product/motor-x1-pro
RotorMax 2200	750 грн	36 мм	2200 KV	11,1 V	420 г	28 A	25 мм	Universal	https://dronehub.in.ua/product/rotor-max-2200
EcoSpin 2000	650 грн	34 мм	2000 KV	11,1 V	350 г	25 A	22 мм	Standard	https://escadrones.com.ua/product/ecospin-2000
SomeTestModel	19	28 мм	37 KV	46 V	55 г	64 A	82 мм	SomeOtherTestValue	SomeDifferentTestLink

Motor:

Рисунок 7.8 – Приклад редагування мотору

Після успішного оновлення даних система негайно відображає повідомлення про успішне завершення операції. Ці повідомлення підтверджують, що запис у базі даних було змінено. Одразу після цього UI оновлюється: відповідна таблиця даних демонструє, що інформація про елемент «Мотор» відображається з новими значеннями, як вказано на рисунку 7.9.

Model	Price	Stator size	KV	Operating voltage	Max thrust	Max current	Mount size	Propeller mount	Link
TurboDrive 2500	900 грн	40 мм	2500 KV	11,1 V	600 г	32 A	20 мм	Compact	https://flymod.net/prod...turbo-drive-2500
PowerLift 2400	850 грн	38 мм	2400 KV	11,1 V	550 г	30 A	20 мм	Universal	https://dronehub.in.ua/product/powerlift-2400
Motor X1 Pro	800 грн	38 мм	2300 KV	11,1 V	500 г	30 A	20 мм	Standard	https://escadrones.com.ua/product/motor-x1-pro
RotorMax 2200	750 грн	36 мм	2200 KV	11,1 V	420 г	28 A	25 мм	Universal	https://dronehub.in.ua/product/rotor-max-2200
EcoSpin 2000	650 грн	34 мм	2000 KV	11,1 V	350 г	25 A	22 мм	Standard	https://escadrones.com.ua/product/ecospin-2000
SomeTestModel	19 грн	28 мм	37 KV	46 V	55 г	64 A	82 мм	SomeOtherTestValue	SomeDifferentTestLink

Motor:

Рисунок 7.9 – Таблиця компонентів «Мотор» після оновлення елементу

7.2.4 Тестування операції видалення

Остання видалення була перевірена шляхом видалення запису про мотор, створений у попередньому пункті. Для виконання цієї операції, виберемо потрібний мотор з випадаючого списку, як вказано на рисунку 7.10. Програма вкаже які дані містить рядок бази даних з його ID, але не дозволить їх змінювати, то ж вони будуть відображатись сірим кольором.

Model	Price	KV	Operating voltage	Max thrust	Max current	Propeller Mount	Mount size	Link	
TurboDrive 2500	900 грн	40 MM	2500 kv	11,1 V	600 г	32 A	20 MM	Compact	https://fymod.net/products/tubedrive-2500
PowerLift 2400	850 грн	32 MM	2400 kv	11,1 V	500 г	30 A	20 MM	Universal	https://dronetub.in.ua/products/powerlift-2400
Motor X1 Pro	800 грн	32 MM	2400 kv	11,1 V	500 г	30 A	20 MM	Standard	https://escadrone.com.ua/products/motor-x1-pro
RotorMax 2200	750 грн	28 MM	2200 kv	11,1 V	420 г	28 A	25 MM	Universal	https://dronetub.in.ua/products/rotor-max-2200
EcoSpin 2000	650 грн	34 MM	2000 kv	11,1 V	350 г	25 A	22 MM	Standard	https://escadrone.com.ua/products/ecospin-2000
SomeTestModel	19 грн	28 MM	37 kv	46 V	55 г	64 A	32 MM	Universal	https://escadrone.com.ua/products/sometestmodel

Motor:

Back to drone config

Go back Update Delete

Рисунок 7.10 – Інтерфейс функції видалення елемента з бази даних

Після виконання операції запис негайно зникає з переліку, що гарантує актуальність інформації та узгодженість з базою даних. Результат операції показаний на рисунку 7.11.

Model	Price	Stator size	KV	Operating voltage	Max thrust	Max current	Mount size	Propeller mount	Link
TurboDrive 2500	900 грн	40 мм	2500 KV	11,1 V	600 г	32 A	20 мм	Compact	https://flymod.net/produkt/turbo-drive-2500
PowerLift 2400	850 грн	38 мм	2400 KV	11,1 V	550 г	30 A	20 мм	Universal	https://dronehub.in.ua/produkt/powerlift-2400
Motor X1 Pro	800 грн	38 мм	2300 KV	11,1 V	500 г	30 A	20 мм	Standard	https://escadrone.com.ua/produkt/motor-x1-pro
RotorMax 2200	750 грн	36 мм	2200 KV	11,1 V	420 г	28 A	25 мм	Universal	https://dronehub.in.ua/produkt/rotor-max-2200
EcoSpin 2000	650 грн	34 мм	2000 KV	11,1 V	350 г	25 A	22 мм	Standard	https://escadrone.com.ua/produkt/ecospin-2000

Мотор ▼

Back to drone config

Add

Update

Delete

Рисунок 7.11 – Результат виконання операції видалення

Також були виконані тести на видалення неіснуючих записів. Як позначено на рисунку 7.12, користувач ніяк не може вибрати неіснуючий запис, тож дана операція не може мати неправильно заданих параметрів від користувача

Model	TurboDrive 2500 ▼
Price	✓ TurboDrive 2500
Stator size	PowerLift 2400
KV	Motor X1 Pro
Operating voltage	RotorMax 2200
Max thrust	EcoSpin 2000
Max current	11,1
Propeller Mount	600
Mount size	32
Link	Compact
	20
	https://flymod.net/produkt/turk

Рисунок 7.12 – Параметри випадаючого списку операції видалення

Таким чином, тестування CRUD функціоналу доводить, що система має високий рівень стабільності та коректності при обробці даних. Це демонструє, що інформаційна система може ефективно взаємодіяти з базою даних, забезпечуючи правильну, безпечну та швидку обробку інформації.

7.3 Тестування отримання конфігурації роботизованої системи

Розглянемо функцію отримання конфігурації БПЛА за наступними параметрами: мінімальна дальність польоту — 5 км, мінімальний час польоту — 8 хвилин, мінімальна додаткова вантажопідйомність — 100 грам, максимальна вартість — 13 500 гривень.

На рисунку 7.13 зображено результат роботи модуля підбору компонентів, що відповідає заданим критеріям.

The screenshot displays a drone configuration interface with the following details:

- Left Column (Summary):**
 - Загальна маса: 1130 г
 - Вартість: 13480 грн
 - Відстань польоту: 5,90 км
 - Час польоту: 8,85 хв
 - Додаткова вантажопідйомність: 298 г
- Center Column (Components):**
 - Рама: Vortex X
 - Мотори: Motor X1 Pro
 - Пропелери: Betafpv X Gemfan 40mm-3
 - Контролер: Matek F411-WSE
 - Приймач: BETAFPV ELRS Nano 915MHz
 - Камера: RunCam Phoenix 2 SP Micro 1500TVL
 - Відеопередавач: RUSH VTX Mini 5.8G 350mW
- Right Column (Filters):**
 - Minimum flight distance: 5
 - Minimum flight time: 8
 - Minimum additional weight: 100
 - Max cost: 13500
 - View component list button

Рисунок 7.13 - Перевірка підбору компонентів БПЛА

Після введення будь-яких даних викликається метод, який повертає конфігурації бпла та виводить її на середню частину екрану.

У лівій частині відображається підсумкова інформація про сформований безпілотник. Аналіз цих значень показав, що обрана конфігурація задовільняє вимоги користувача.

У центральній частині відображено конкретні компоненти, які увійшли до цієї збірки. На рисунку 7.14 показано час (у мілісекундах), витрачений на виконання операції пошуку компонентів.

```
[11:51:34] Час витрачений на підбір компонентів 7846
UnityEngine.Debug:Log (object)

Час витрачений на підбір компонентів 7846
UnityEngine.Debug:Log (object)
DroneConfiguratorReal.GetDroneConfiguration (double,double,double,int) (at Assets/Scripts/Managers/DroneConfiguration.cs:188)
InputManager:ConfigureDrone () (at Assets/Scripts/Managers/InputManager.cs:105)
InputManager:ValidateDecimalInput (TMPPro.TMP_InputField,single&) (at Assets/Scripts/Managers/InputManager.cs:90)
InputManager:<Start>b__19_3 (string) (at Assets/Scripts/Managers/InputManager.cs:42)
UnityEngine.Events.UnityEvent`1<string>:Invoke (string)
TMPPro.TMP_InputField:SendOnEndEdit () (at
```

Рисунок 7.14 – Лог часу витраченого на операцію підбору компонентів

Була проведена перевірка ситуації, коли знаходження конфігурації БПЛА є неможливим. Результат цієї перевірки вказаний на рисунку 7.15.

Конфігурацію не знайдено!

- Рама: неможливо підібрати
- Мотори: неможливо підібрати
- Пропелери: неможливо підібрати
- Контролер: неможливо підібрати
- Приймач: неможливо підібрати
- Камера: неможливо підібрати
- Відеопередавач: неможливо підібрати

Minimum flight distance: 8

Minimum flight time: 8

Minimum additional weight: 100

Max cost: 13500

[View component list](#)

Рисунок 7.15 – Перевірка неможливості отримання конфігурації

ВИСНОВКИ

У дипломному проєкті було створено інформаційну систему оптимізації та підбору компонентів роботизованих платформ. Система автоматизує процес вибору сумісних комплектуючих на основі технічних характеристик, вартості та доступності, що дозволяє скоротити час на підбір і підвищити точність прийняття рішень у сфері закупівель.

У розділі 1 було проведено аналіз предметної області, описано основні типи інформаційних систем та баз даних, що використовуються в сучасних ІТ-рішеннях. Визначено ключові проблеми, пов'язані з вибором дронів компонентів, та обґрунтовано доцільність створення спеціалізованої інформаційної системи.

У розділі 2 було зроблено огляд існуючих програмних рішень, таких як SAP Ariba, Oracle Procurement Cloud, Octopart, Digi-Key та інші. Виявлено їхні сильні та слабкі сторони, а також проведено порівняльний аналіз, що підтвердив відсутність готового рішення, яке б повністю відповідало вимогам до підбору дронів компонентів.

У розділі 3 наведено обґрунтування вибору технологій розробки. Розглянуто та порівняно ігрові рушії, обґрунтовано вибір Unity як основного середовища. Також визначено операційну систему розгортання (Windows), інструменти контролю версій, СУБД та додаткові фреймворки, що використовуються у проєкті.

У розділі 4 було розроблено діаграму прецедентів, яка демонструє взаємодію користувача і модератора з системою, дозволяє візуалізувати функціональні можливості для кожної ролі та зрозуміти логіку доступу до різних частин функціоналу.

У розділі 5 було створено реляційну базу даних, що включає таблиці з інформацією про компоненти дронів, користувачів, ролі, запити на зміни та інші структуровані сутності. Було реалізовано їх зв'язки та підтримку запитів через SQL-запити.

У розділі 6 було описано архітектуру проєкту, використані шаблони проектування, взаємозв'язки між сутностями бази даних та їх відображення у вигляді

					ІК13.160БАК.006 ПЗ	Арк.
Зм.	Лист	№ докум.	Підпис	Дата		58

моделей. Також описано особливості взаємодії між рівнями: користувацьким, логічним та рівнем даних.

У розділі 7 описано тестування основних логічних функцій системи, включно з перевіркою зчитування даних, відображення у таблиці, генерації конфігурацій та модерації запитів. Проведені випробування підтвердили стабільність роботи програми та правильність реалізації її функціоналу.

Створена інформаційна система відповідає поставленим вимогам технічного завдання. Розроблене програмне забезпечення дозволяє автоматизувати підбір сумісних елементів на основі заданих параметрів, забезпечує зручний графічний інтерфейс користувача, ефективну взаємодію з реляційною базою даних SQL Server та підтримує багаторівневу модель доступу. Система демонструє стабільну та коректну роботу при тестуванні основних функцій, що свідчить про якісну реалізацію логіки та інтерфейсної взаємодії. Отримані результати можуть бути використані в реальних умовах для проєктування, аналізу та підбору комплектуючих дронів, а також легко масштабуються й адаптуються для подальшого використання в комерційних або освітніх цілях.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Everything You Need To Know About Procurement Systems. ProcureDesk.
URL: <https://www.procuredesk.com/procurement-system/>
2. SAP Ariba – Introduction, Modules, and Features. GeeksforGeeks. URL:
<https://www.geeksforgeeks.org/sap-ariba-introduction-modules-and-features/>
3. Oracle Procurement Cloud ERP – Oracle. URL:
<https://www.oracle.com/erp/procurement/>
4. Coupa Procurement Reviews 2025: Pricing, Features & More – SelectHub.
URL: <https://www.selecthub.com/p/procurement-software/coupa-procurement/>
5. Features of Zoho Creator Procurement. Zoho. URL:
<https://www.zoho.com/creator/procurement/features.html>
6. Procurement Software Native to Salesforce Salesforce AppExchange. URL:
<https://appexchange.salesforce.com/appxListingDetail?listingId=a0N300000003HoQQEA0>
7. Electronic Components Search Engine Octopart. URL: <https://octopart.com/>
8. Electronic Components and Parts Search Digi-Key Electronics. URL:
<https://www.digikey.com/en/products>
9. Unity Technologies. Unity User Manual. URL:
<https://docs.unity3d.com/Manual/index.html>
10. Epic Games. Unreal Engine Documentation. URL:
<https://docs.unrealengine.com/>
11. Godot Engine. Official Documentation. URL:
<https://docs.godotengine.org/en/stable/>
12. Crytek. CRYENGINE Documentation. URL:
<https://www.cryengine.com/docs>
13. Gregory, J. (2020). Game Engine Architecture. CRC Press.
14. Unity Manual: Windows Standalone Build Settings. URL:
<https://docs.unity3d.com/6000.0/Documentation/Manual/WindowsStandaloneBinaries.html>

15. Unity Manual: WebGL Getting Started. URL: <https://docs.unity3d.com/Manual/webgl-gettingstarted.html>

16. Git - About Version Control. Git SCM. URL: <https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control>

17. Microsoft. ADO.NET Overview. URL: <https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/ado-net-overview>

18. Microsoft SQL Server Documentation. URL: <https://docs.microsoft.com/en-us/sql/sql-server/>

19. Unity Technologies. TextMeshPro Documentation. URL: <https://docs.unity3d.com/Packages/com.unity.textmeshpro@latest/index.html>

System.Data.SqlClient Namespace | Microsoft Learn. URL: <https://learn.microsoft.com/en-us/dotnet/api/system.data.sqlclient?view=windowsdesktop-9.0>