МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК ТА ТЕХНОЛОГІЙ

Кафедра Комп'ютерних систем та мереж

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач випускової кафедри
_____ Ігор ЖУКОВ
«___» _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ
"МАГІСТР"
ЗА СПЕЦІАЛЬНІСТЮ 123 «КОМП'ЮТЕРНА ІНЖЕНЕРІЯ»

Тема: Комп'ютерна десктопна система для обслуговування додатків

Виконавець: Віталій КУЛІНІЧ
Керівник: Володимир АНТОНОВ
Нормоконтролер: Василь МАЛЯРЧУК

Засвідчую, що у кваліфікаційній роботі немає запозичень із праць інших авторів без відповідних посилань
Студент: Віталій КУЛІНІЧ

КИЇВ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук та технологій

Кафедра комп'ютерних систем та мереж

Напрям (спеціальність) 123 "Комп'ютерна інженерія"

ЗАТВЕРДЖУЮ

Завідувач випускової кафедри

_____ Ігор ЖУКОВ

(підпис)

«___» _____ 2023 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи

Кулініча Віталія Андрійовича

(прізвище, ім'я, по батькові випускника в родовому відмінку)

1. Тема роботи: Комп'ютерна десктопна система для обслуговування додатків затверджена наказом ректора від "29" серпня 2023 року № 1521/ст.

2.Термін виконання роботи: з 02.10.2023 до 31.12.2023

3.Вихідні дані до роботи: Технології для десктопних систем в сфері авіації

4.Зміст пояснювальної записки:
1. Огляд існуючих систем для обслуговування додатків в авіації. 2. Технології для розробки десктопних додатків. 3. Аналіз потреб авіаційної галузі. 4. Розробка десктопної системи.

5.Перелік обов'язкового графічного матеріалу:
Таблиці, рисунки, діаграми, графіки, презентація *PowerPoint*

6. Календарний план-графік

№ з/п	Завдання	Термін виконання	Підпис керівника
1	Ознайомитись з постановкою задачі дипломної роботи	04.10.2023 - 06.10.2023	
2	Вивчити спеціальну літературу і технічну документацію	07.10.2023 - 10.10.2023	
3	Провести аналіз наявних технологій передачі даних	11.10.2023 - 15.10.2023	
4	Написати розділ 1 дипломної роботи	16.10.2023 - 25.10.2023	
5	Написати розділ 2 дипломної роботи	26.10.2023 - 05.11.2023	
6	Написати розділ 3 дипломної роботи	06.11.2023- 25.11.2023	
7	Розробка десктопного застосунку	26.11.2023 - 30.11.2020	
8	Написати розділ 4 дипломної роботи	30.11.2020- 17.12.2023	
9	Підготувати всі матеріали роботи	18.12.2020- 21.12.2020	
10	Захистити дипломну роботу	22.12.2020- 31.12.2020	

7. Дата отримання завдання « 02 » жовтня 2023 р.

Керівник дипломної роботи Володимир АНТОНОВ

(підпис)

Завдання прийняв до виконання Віталій КУЛІНІЧ

(підпис студента)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи «Комп’ютерна десктопна система для обслуговування додатків»: 90 сторінку, 19 рисунків, 22 літературних джерела та 1 додаток.

Об'єктом дослідження: велика та складна система авіаційної галузі, яка включає в себе весь цикл життєвого процесу від концепції до експлуатації літаків. Особливий акцент робиться на процесі збереження та ефективного управління конструкторською документацією, яка включає в себе різноманітні дані, такі як креслення, технічні специфікації, схеми та інші важливі аспекти інженерних проектів.

Мета створення десктопної системи: розв'язання конкретних викликів та задач, які інженери та фахівці зіткнуться в процесі своєї роботи.

Методи дослідження: аналіз вимог, розробка програмного забезпечення, тестування та оцінка ефективності.

Ці методи та етапи дослідження дозволять досягти поставленої мети розробки комп'ютерної десктопної системи для обслуговування авіаційних додатків та забезпечити її надійність, безпеку та ефективність у сфері авіаційної галузі.

Результатом магістерської роботи є десктопний додаток, призначений для управління конструкторською документацією в авіаційній галузі. Додаток володіє розширеним та комплексним функціоналом, спрямованим на полегшення організації та взаємодії з документацією, а також на підвищення ефективності технічного обслуговування та безпеки польотів.

АВІАЦІЙНА ГАЛУЗЬ, ДЕСКТОПНА СИСТЕМА, КРОСПЛАТФОРМЕНІСТЬ, ШТУЧНИЙ ІНТЕЛЕКТ, ОРГАНІЗАЦІЯ ДОКУМЕНТІВ, ІНТЕРФЕЙС.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

API – Інтерфейс програмування застосунків

AI – Штучний інтелект

CI – Безперервна інтеграція

CD – Безперервна доставка

CSS – Каскадні таблиці стилів

HTML – Мова розмітки гіпертексту

HTTPS – Захищений протокол передачі гіпертексту

IoT – Інтернет речей

IDE – Інтегроване середовище розробки

JSON – Об'єктний нотаційний формат обміну даними

REST – Представлення стану переходу

SDK – Набір розробника програмного забезпечення

UI – Інтерфейс користувача

UX – Враження користувача

XML – Мова розширюваних міток

ВСТУП

В сучасному світі інформаційних технологій авіаційна галузь зазнає значних змін і вимагає постійного удосконалення. Швидкі та неперервні інновації у сфері авіаційних технологій відкривають нові горизонти для авіаційної промисловості, прискорюють перевезення та зробили авіацію більш доступною для людей з усього світу. Проте, цей постійний розвиток також створює численні виклики, особливо в контексті обслуговування додатків, що використовуються для контролю, навігації та управління повітряним транспортом [1].

Однією з ключових галузей в авіації є розробка та підтримка програмного забезпечення, яке використовується в повітряній аеронавтиці. Ці програми допомагають у забезпеченні безпеки польотів, навігації, контролю за рухом повітряних суден та багатьох інших аспектах авіаційної діяльності. Сучасні авіаційні додатки стають все більш складними та функціональними, оскільки вони повинні враховувати різні фактори, такі як зміни в аеронавтиці, потреби клієнтів та вимоги безпеки.

Однак цей рост у складності та функціональності авіаційних додатків також ставить під загрозу їхню надійність та ефективність обслуговування. Недоліки у програмному забезпеченні, несправності та збої можуть мати серйозні наслідки, включаючи аварії та втрати людських життів. Тому проблема полягає у необхідності створення комп'ютерної десктопної системи, яка забезпечить ефективне та безпечне обслуговування цих додатків в авіаційній галузі. Така система буде спрямована на забезпечення надійності, безпеки та оптимізації функціонування авіаційних програм для підтримки безпечних та ефективних польотів у сучасному інформаційному середовищі.

Актуальність теми обумовлена зростаючою складністю авіаційних додатків та постійними змінами в галузі авіації. Сучасний розвиток авіаційних технологій призвів до значного розширення функціональних можливостей додатків, які використовуються для контролю, навігації та управління повітряним транспортом.

Вони стали важливими інструментами в авіаційній діяльності, забезпечуючи безпеку та ефективність польотів.

Наприклад, зі збільшенням об'єму даних, що обробляються авіаційними додатками, зростає необхідність у вдосконаленні інструментів їхнього моніторингу та підтримки. Сучасні авіаційні системи генерують величезну кількість даних, таких як дані про польоти, навігаційну інформацію, дані про стан обладнання літака та інші, які потребують надійного збереження, обробки та передачі. Оптимізація цих процесів допоможе забезпечити реалізацію пілотажних завдань та безпеку польотів.

Крім того, забезпечення високої рівні доступності та безпеки авіаційних додатків стає пріоритетною задачею в сучасному світі. Заходи щодо кібербезпеки та захисту інформації у сфері авіації набувають все більшого значення, оскільки вони впливають на безпеку пасажирів та екіпажів. Завдяки створенню спеціалізованої десктопної системи для обслуговування авіаційних додатків, можливо забезпечити високий рівень захисту даних та зменшити вразливість перед можливими кібератаками.

Створення десктопної системи для авіаційній галузі з коментуванням є актуальним завданням з численними перевагами для інженерів та фахівців у цій сфері. Великий обсяг технічної документації в авіаційній галузі вимагає ефективного та структурованого зберігання. Додаток надає зручні інструменти для організації та збереження документів, роблячи їх доступними для інженерів [2].

Інженерні проекти вимагають спільної роботи та комунікації. Додаток надає можливість коментувати та обговорювати документацію в реальному часі, поліпшуючи ефективність роботи та сприяючи кращому розумінню завдань.

Можливість залишати реакції в додатку дозволяє швидко оцінювати правильність інформації. Це допомагає виділяти важливі аспекти та покращує якість конструкторської документації.

Використання штучного інтелекту для аналізу коментарів та автоматичного виправлення помилок є інноваційним підходом, який може значно підвищити якість та точність інженерної документації. Застосування інтеграції з AI може допомогти

автоматизувати рутинні завдання, такі як перевірка правильності даних чи виявлення потенційних помилок.

Розробка додатку, який враховує сучасні технології та потреби, дозволяє підтримувати актуальність у довгостроковій перспективі та реагувати на зміни в галузі.

Усі ці аспекти взаємодіють для створення інноваційного та ефективного інструменту, який допомагає інженерам в авіаційній галузі покращити свою роботу та досягати високих стандартів якості [3].

Отже, розробка і впровадження такої системи стає необхідністю в контексті сучасних викликів та змін у галузі авіації, і вона сприятиме підвищенню безпеки, надійності та ефективності авіаційного транспорту.

Мета створення десктопної системи є розв'язання конкретних викликів та задач, які інженери та фахівці зіткнуться в процесі своєї роботи. Основні аргументи та мети створення такого додатку включають:

1. Покращення організації та зберігання документації: Забезпечення зручного та ефективного зберігання конструкторської документації, що включає в себе креслення, схеми, технічні специфікації та інші види документів. Мета - створити структуровану систему, яка дозволяє інженерам швидко та ефективно знаходити необхідну інформацію.

2. Підтримка комунікації та спільної роботи: Створення інструментів для коментування, обговорення та взаємодії з конструкторською документацією. Мета - полегшити комунікацію між інженерами, проектними групами та іншими учасниками процесу.

3. Оцінка та Покращення Якості Документації: Забезпечення можливості залишати коментарі, лайки та дизлайки, що дозволяє швидко оцінювати правильність інформації. Мета - створити механізм для визначення важливих аспектів та вдосконалення якості конструкторської документації.

4. Інтеграція з Штучним Інтелектом: Впровадження системи штучного інтелекту для автоматичного аналізу коментарів, визначення їхньої важливості та

можливості автоматичного виправлення помилок. Мета - використання передових технологій для підвищення точності та ефективності роботи.

5. Підвищення Продуктивності та Швидкості Роботи: Забезпечення інструментів для швидкого доступу до необхідної інформації, спрощення взаємодії з документацією та підвищення продуктивності роботи інженерів.

6. Створення Інноваційного та Сучасного Інструменту: Розробка додатку, який враховує сучасні технології та вимоги галузі, що дозволяє створити інноваційний та актуальний інструмент для роботи з конструкторською документацією.

В цілому, мета полягає в створенні комплексного та ефективного інструменту, який полегшить роботу інженерів, підвищить якість та швидкість обробки документації, а також сприятиме спільній роботі та інноваціям в авіаційній галузі.

Тому розробка комп'ютерної десктопної системи, яка спростить та оптимізує обслуговування додатків в галузі авіації, покращить безпеки та надійності авіаційних додатків, зменшить час та ресурси, витрачені на їхнє обслуговування та забезпечить оптимальну продуктивність і є метою даної роботи. Ця система має сприяти підвищенню загальної ефективності та безпеки авіаційної галузі в цілому.

Об'єктом дослідження є велика та складна система авіаційної галузі, яка включає в себе весь цикл життєвого процесу від концепції до експлуатації літаків. Особливий акцент робиться на процесі збереження та ефективного управління конструкторською документацією, яка включає в себе різноманітні дані, такі як креслення, технічні специфікації, схеми та інші важливі аспекти інженерних проектів [4].

Цей об'єкт дослідження є крайньою мірою важливим, оскільки авіаційна галузь є технічно високорозвиненою та стурбованою сферою, де точність, доступність та надійність конструкторської інформації є ключовими для успішного виробництва та функціонування літаків.

Предметом вивчення є новаторський десктопний додаток, ретельно розроблений для оптимізації всіх аспектів взаємодії з конструкторською документацією в авіаційній галузі. Цей додаток виступає як ключовий інструмент для

покращення організації, взаємодії та ефективності роботи інженерів, що працюють над проектами в авіаційній сфері.

До складу цього додатку входять не лише загальноприйняті інструменти збереження і коментування, але й передові засоби колективного редагування, які дозволяють інженерам спільно працювати над документацією в режимі реального часу. Крім того, інтеграція штучного інтелекту в додаток розширює його функціональність, надаючи можливість автоматизованого аналізу коментарів та вдосконалення рішень на основі зібраної інформації.

Цей предмет дослідження є вкрай актуальним, враховуючи стрімкий технологічний розвиток в авіаційній галузі та потребу в постійних покращеннях у збереженні та обробці конструкторської інформації для ефективної роботи інженерів.

Для досягнення поставленої мети передбачено наступні завдання:

1. Провести аналіз існуючих систем обслуговування додатків в авіації з метою визначення їхніх переваг і недоліків. Це включає в себе вивчення різних підходів і рішень, які вже існують у сфері обслуговування авіаційних додатків. Порівняння і аналіз цих систем допоможуть визначити їхні сильні та слабкі сторони, а також виділити проблемні аспекти, які варто врахувати під час розробки нової системи.

2. Визначити функціональні вимоги до розроблюваної десктопної системи. В цьому завданні необхідно конкретизувати, які конкретні функції повинна виконувати система, щоб задовольняти потреби авіаційної галузі. Це може включати в себе вимоги до моніторингу, аналізу, діагностики, а також інструменти для захисту від кібератак і незаконного доступу.

3. Розробити десктопну програму, яка відповідає цим вимогам та забезпечує надійне обслуговування авіаційних додатків. На цьому етапі роботи буде створено програмне забезпечення, яке враховує функціональні вимоги та відповідає стандартам безпеки та надійності.

4. Провести тестування та оцінку ефективності розробленої системи. Після створення десктопної програми важливо піддати її тестуванню для перевірки її

працездатності та відповідності вимогам. Оцінка ефективності дозволить визначити, наскільки успішно система впоралася зі своїми завданнями та як можна її покращити.

Ці завдання спрямовані на розв'язання проблем та вдосконалення процесу обслуговування авіаційних додатків, забезпечуючи високу надійність та безпеку в авіаційній галузі.

Огляд літератури показав, що вже існують деякі розробки в області обслуговування авіаційних додатків, але більшість з них потребують подальшого вдосконалення. Існуючі системи мають свої переваги та обмеження, і важливо провести докладний аналіз цих систем для ідентифікації їхніх сильних та слабких сторін.

Дослідження деталей існуючих систем обслуговування авіаційних додатків буде проведено під час подальших розділів цієї роботи з метою визначення кращих практик та найбільш обіцяних підходів. Важливо врахувати відгуки користувачів, а також оцінки ефективності і безпеки існуючих систем.

Крім того, враховуючи швидкий розвиток технологій та зміни в галузі авіації, важливо постійно оновлювати і вдосконалювати системи обслуговування додатків. Під час подальших досліджень будуть вивчені актуальні тенденції та інновації, які впливають на обслуговування авіаційних додатків.

Отже, аналіз існуючих систем та подальша розробка докладних рекомендацій щодо їх вдосконалення є важливими кроками у досягненні мети цієї магістерської роботи, яка полягає у розробці спеціалізованої комп'ютерної десктопної системи для обслуговування авіаційних додатків.

Дослідження включатиме аналіз літератури, а також практичну розробку десктопної системи. Методи, які будуть використовуватися, включають в себе аналіз вимог, розробку програмного забезпечення, тестування та оцінку ефективності.

1. Аналіз вимог: Перший етап дослідження передбачає детальний аналіз вимог до системи. Це включає в себе визначення функціональних та нефункціональних вимог, а також вимог щодо безпеки, надійності та продуктивності системи. Важливо встановити потреби користувачів та врахувати специфіку авіаційних додатків.

2. Розробка програмного забезпечення: На основі вимог буде розроблена десктопна програма для обслуговування авіаційних додатків. Під час розробки будуть використовуватися сучасні програмні технології та методи розробки.

3. Тестування: Розроблена система буде піддана ретельному тестуванню. Це включає в себе модульне тестування, інтеграційне тестування та системне тестування. Метою тестування є виявлення та виправлення помилок та несправностей в програмному забезпеченні.

4. Оцінка ефективності: Після успішного тестування системи буде проведено оцінку її ефективності. Це включає в себе аналіз продуктивності системи, її впливу на надійність та безпеку авіаційних додатків. Результати оцінки будуть використані для покращення системи та оптимізації її роботи.

Ці методи та етапи дослідження дозволять досягти поставленої мети розробки комп'ютерної десктопної системи для обслуговування авіаційних додатків та забезпечити її надійність, безпеку та ефективність у сфері авіаційної галузі [5].

Наукова новизна отриманих результатів:

1. Впровадження інноваційних інструментів для збереження та управління конструкторською документацією в авіаційній галузі. Розроблений десктопний додаток впроваджує передові методи зберігання та структурування документації, що враховує особливості галузі та вимоги інженерів. Це дозволяє покращити організацію даних та ефективність роботи інженерів.

2. Інтеграція системи коментарів та обговорень з можливістю колективного редагування. Реалізація функціоналу коментарів та системи обговорень, що працюють у реальному часі, дозволяє інженерам обмінюватися думками та надавати спільний внесок у розробку та аналіз конструкторської документації. Це покращує комунікацію та сприяє колективній роботі.

3. Використання штучного інтелекту (*AI*) для аналізу коментарів та автоматичного виправлення помилок. Інтеграція *AI* дозволяє вдосконалити процеси аналізу та редагування документації, забезпечуючи високу точність та автоматизуючи рутинні завдання.

4. Спрощення процесів спільної роботи та створення спільних проектів. Впровадження можливості спільного редагування документів та формування груп для ефективної колективної роботи дозволяє інженерам працювати синергетично та досягати кращих результатів.

5. Інтеграція інструментів для оцінювання та вдосконалення якості документації. Введення системи лайків та дизлайків, яка взаємодіє з штучним інтелектом для аналізу важливості коментарів, сприяє швидкому оцінюванню правильності інформації та вдосконаленню якості конструкторської документації.

Отже, отримані результати визначаються впровадженням сучасних технологій та інновацій в галузь авіаційної інженерії, що покликане зробити роботу інженерів більш продуктивною, ефективною та спрощеною.

Практичним значенням отриманих результатів є полегшення організації та доступ до конструкторської документації, що призводить до підвищення ефективності інженерних процесів. Інструменти коментування та колективного редагування дозволяють інженерам обговорювати та працювати над проектами у реальному часі, сприяючи ефективній комунікації та взаємодії. Автоматизація рутинних завдань, таких як аналіз та виправлення помилок, дозволяє інженерам скорочувати часові витрати на адміністративні завдання і фокусуватися на ключових аспектах роботи. Застосування інноваційних інструментів і підходів дозволяє інженерам відповідати сучасним викликам та залишатися актуальними у швидкозмінному світі авіаційних технологій. Додаток дозволяє швидко виявляти та усувати потенційні помилки або невірності в конструкторській документації, що може значно знизити ризики неправильного розуміння та використання інженерним персоналом.

Отже, отримані результати мають значущий вплив на всі аспекти інженерної діяльності в авіаційній галузі, відзначаючись практичною вигідністю та великим потенціалом для покращення якості та продуктивності.

РОЗЛІД 1
ОГЛЯД ІСНУЮЧИХ СИСТЕМ ОБСЛУГОВУВАННЯ ДОДАТКІВ В АВІАЦІЇ

1.1. Типи авіаційних додатків

В авіаційній галузі існують різні типи додатків, які використовуються для контролю, навігації, обміну даними та управління повітряним транспортом. Розглянемо деякі з них:

Системи навігації. Ці додатки відіграють важливу роль у забезпеченні безпеки та точності польотів. Вони включають в себе програми для планування маршрутів, навігації під час польоту, моніторингу погодних умов та навколишнього повітряного простору. Наприклад, системи *GPS* та авіаційні картографічні додатки допомагають пілотам у точному плануванні та виконанні польотів [6]. Приклад системи навігації наведений на (рис. 1.1).

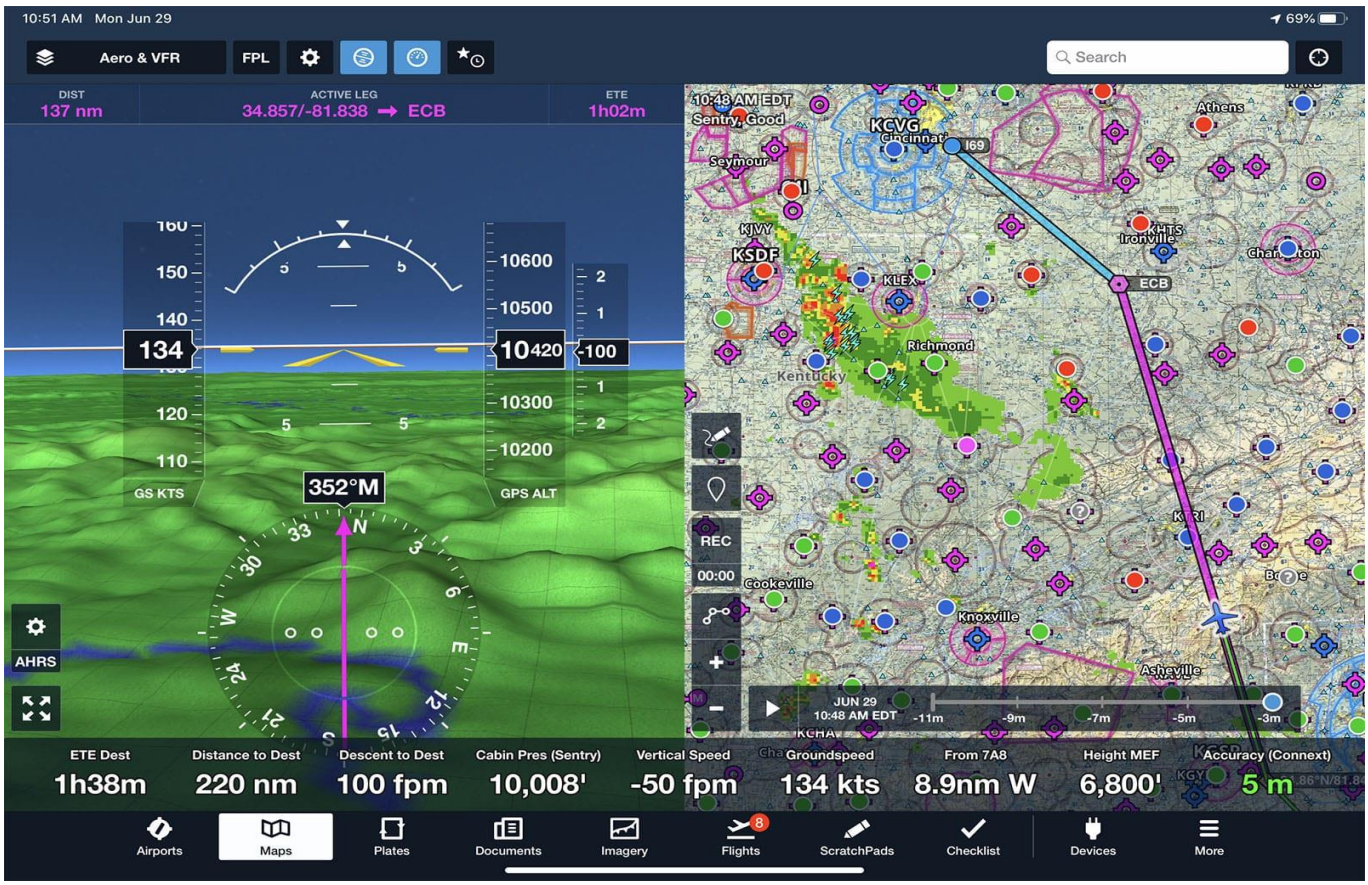


Рис. 1.1. Система навігації

Системи моніторингу та діагностики: Ці додатки спрямовані на забезпечення безпеки та надійності повітряних суден. Вони використовуються для постійного моніторингу стану літаків, виявлення будь-яких несправностей та вчасного їх виправлення. Наприклад, системи діагностики можуть відслідковувати роботу двигунів, системи керування та електроніку літака для запобігання можливим аваріям [7]. Приклад системи моніторингу та діагностики наведений на рис. 1.2.

Рис. 1.2. Система моніторингу та діагностики

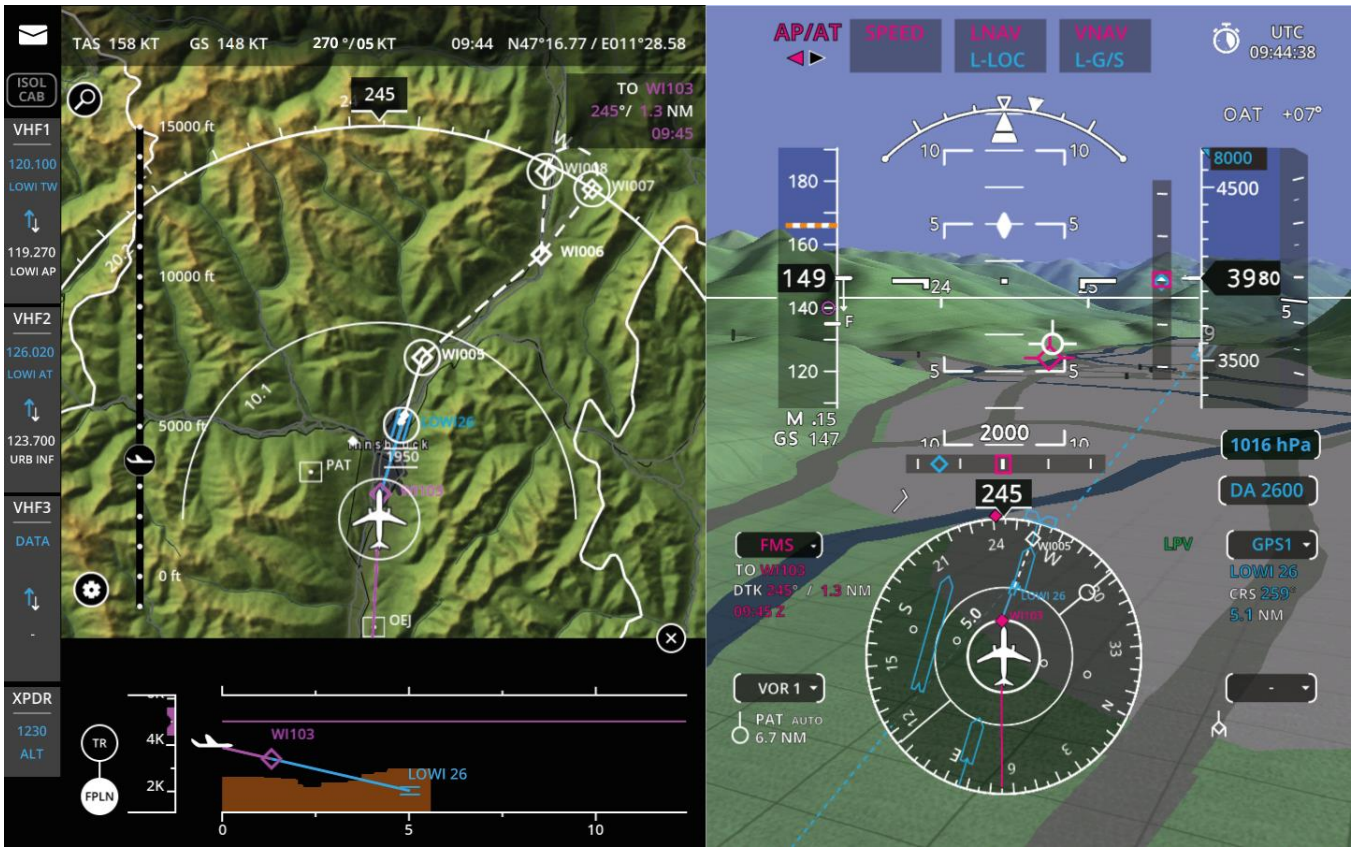
Системи управління бортовою авіонікою: Ці додатки включають в себе програми для управління різноманітними системами та пристроями на борту літака. Їх функціонал варіює від управління приладами та системами безпеки до комунікаційних систем. Наприклад, пілоти можуть використовувати додатки для керування аварійними процедурами та системами пожежогасіння [8]. Приклад системи управління бортовою авіонікою наведений на (рис. 1.3).

Системи зв'язку та обміну даними (рис 1.4). Ці додатки грають ключову роль у забезпеченні зв'язку між літаками, з контрольними центрами та землею. Вони включають в себе системи радіозв'язку, супутниковий зв'язок, системи передачі даних

та системи автоматичного звіту про місцезнаходження літака. Це дозволяє забезпечити безпеку та координацію польотів, а також вчасну передачу важливої інформації.

Рис. 1.3. Система управління бортовою авіонікою

Враховуючи різноманітність та важливість різних типів авіаційних додатків, від систем управління польотами та навігаційних засобів до програм для діагностики та



обслуговування повітряних суден, розробка спеціалізованої десктопної системи стає ключовим етапом у подальшому розвитку авіаційної індустрії. Така система виходить за рамки звичайного управління документацією та надає комплексний підхід до вирішення різноманітних завдань, що стоять перед різними гравцями в галузі. У цілому, спеціалізована десктопна система відповідає на нагальні виклики та покращує ключові аспекти авіаційної діяльності, сприяючи подальшому розвитку та безпеці галузі.

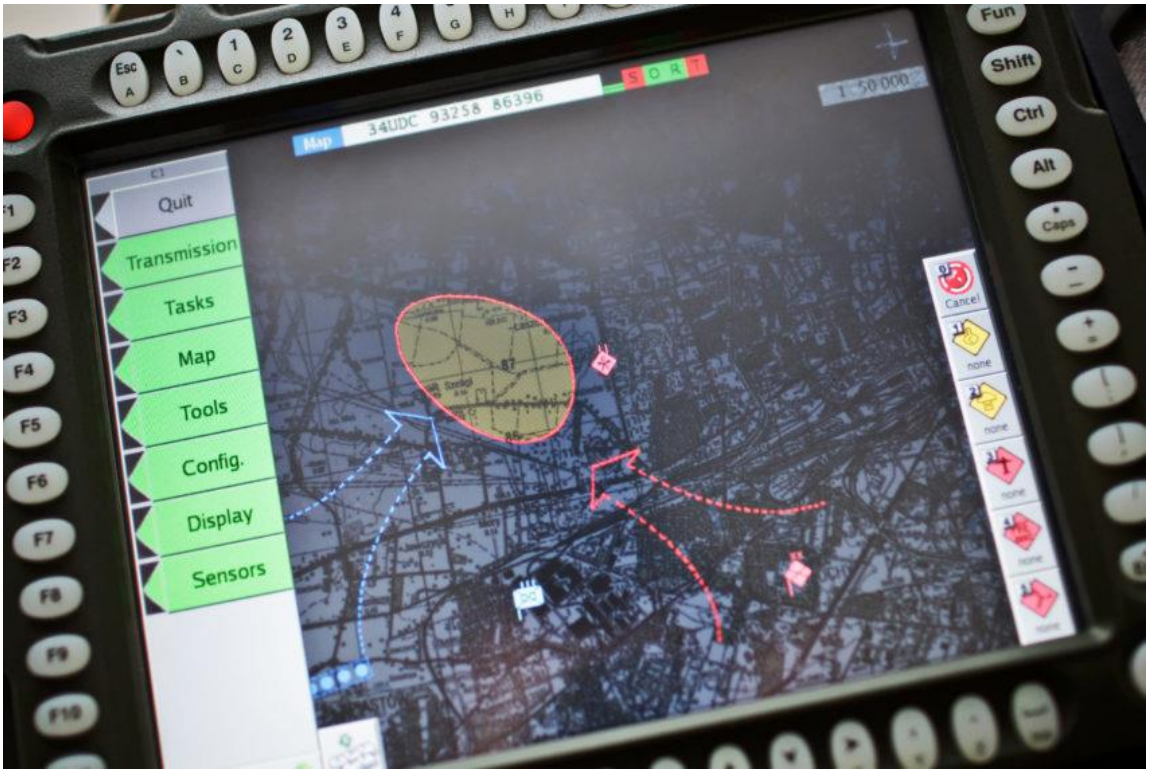


Рис. 1.4. Система зв'язку та обміну даними

1.2. Огляд існуючих систем обслуговування

Для забезпечення надійності, безпеки та ефективності авіаційних додатків існують спеціалізовані системи обслуговування. Ці системи відіграють важливу роль у гарантуванні безперебійної роботи авіаційних програм, які використовуються під час польотів та управління повітряним транспортом. Нижче детально розглянемо кожен з ключових компонентів цих систем:

1. Серверні системи:

Серверні системи є центральною складовою інфраструктури для обслуговування авіаційних додатків. Вони відповідають за зберігання, обробку та передачу даних, які використовуються в авіаційних програмах. Оскільки безперебійна робота цих додатків є критичною для безпеки польотів, серверні системи мають високий рівень надійності і доступності. Зазвичай вони встановлюються у великих дата-центрах з резервним живленням та системами аварійного відновлення для запобігання відмовам. Приклад серверні системи зображений на рисунку 2.5.

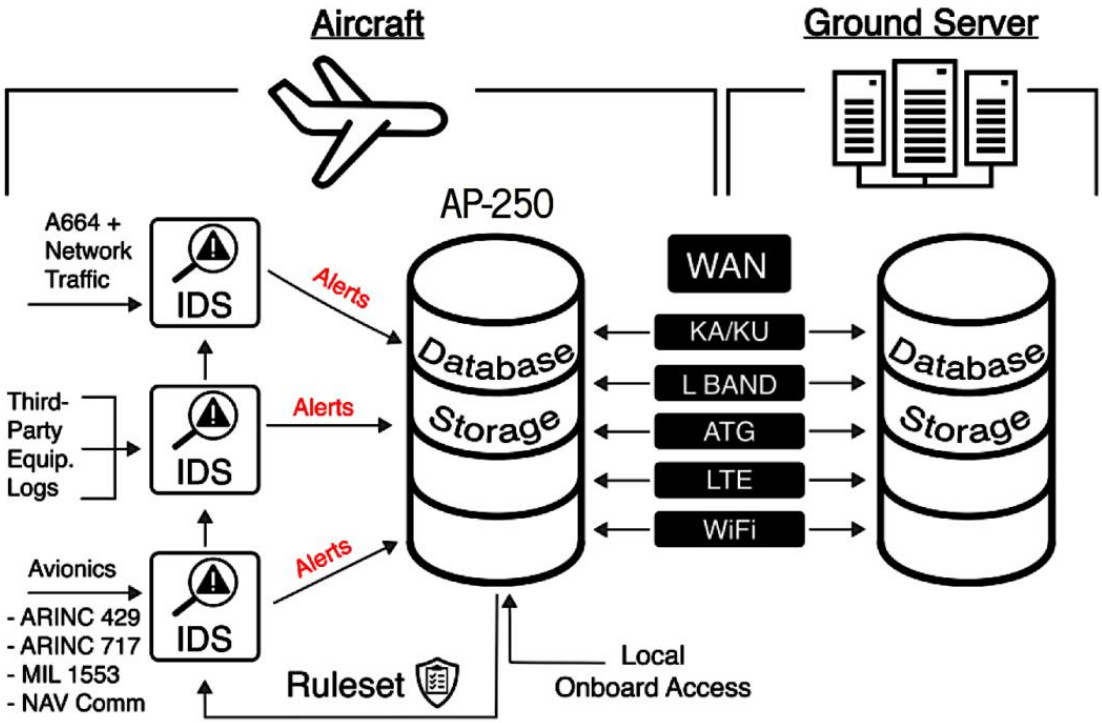


Рис. 1.5. Серверна система

2. Системи моніторингу та діагностики:

Системи моніторингу та діагностики (рис. 1.6) відіграють ключову роль у підтримці надійності та безпеки авіаційних додатків. Вони надають можливість постійно відстежувати стан програмного та апаратного забезпечення. Ці системи автоматично виявляють потенційні проблеми, включаючи аномальні параметри роботи, і надають дані для профілактичного обслуговування. Наприклад, вони можуть сповістити технічний персонал про несправності, такі як падіння продуктивності апаратних компонентів чи програмних помилок [9].

3. Інструменти для аналізу логів:

Інструменти для аналізу логів (рис. 1.7) є важливим засобом виявлення та аналізу проблем, що можуть виникнути під час роботи авіаційних додатків. Вони реєструють події та помилки, що відбуваються в системі, і дозволяють ідентифікувати причини неполадок. Аналіз логів допомагає вчасно виявити та усунути проблеми, забезпечуючи таким чином надійну роботу авіаційних додатків.

Рис. 1.6. Системи моніторингу та діагностики

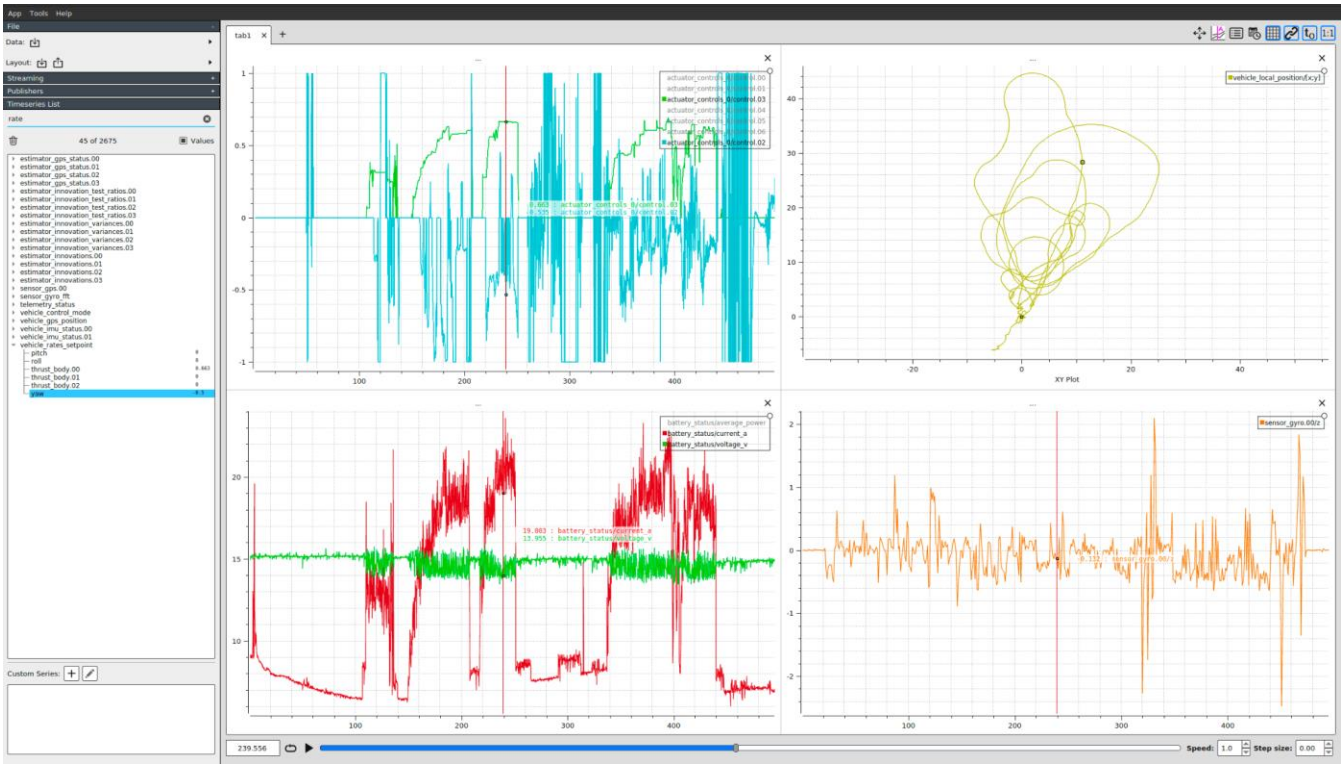


Рис. 1.7. Інструмент для аналізу логів

4. Засоби захисту та безпеки:

У галузі авіації безпека має найвищий пріоритет, і системи обслуговування авіаційних додатків повинні мати високий рівень захисту від кібератак та несанкціонованого доступу. Це включає в себе механізми аутентифікації, авторизації та аудиту, щоб гарантувати, що лише авторизовані користувачі мають доступ до системи. Шифрування даних використовується для забезпечення конфіденційності і цілісності інформації. Також розробляються системи виявлення та запобігання кібератак, оскільки збільшується загроза кіберпроникнення в авіаційну інфраструктуру.

Загалом, існуючі системи обслуговування авіаційних додатків грають важливу роль у забезпеченні безпеки та функціональності в цій критично важливій галузі. Вони вимагають постійного вдосконалення та підтримки, оскільки авіаційні додатки стають все більш складними та залежними від технологій інформаційних систем.

1.3. Проблеми та виклики

Авіаційна галузь стикається зі специфічними проблемами та викликами у зв'язку з обслуговуванням додатків, що використовуються для контролю, навігації та управління повітряним транспортом. Ці проблеми і виклики мають велике значення для забезпечення безпеки та ефективності авіаційного транспорту.

1.3.1. Надійність і доступність

24/7 доступність авіаційних додатків є вимогою, яка випливає з надзвичайно важливої ролі, яку вони відіграють в авіаційній галузі. Перш за все, авіаційні додатки використовуються для контролю та навігації повітряних судів, і будь-яка перерва у їхній роботі може призвести до серйозних наслідків, включаючи викиди з маршруту та збої у плануванні польотів. Наприклад, навігаційні системи визначають оптимальні маршрути та координують рух літаків у повітряному просторі. Короткочасна

недоступність цих систем може призвести до заторів, збільшити споживання пального та викликати нестабільність у повітряному русі.

З іншого боку, обслуговування авіаційних додатків також включає в себе надання даних та підтримки пілотам та повітряним контрольним центрам у реальному часі. Навіть кілька секунд відмови може призвести до втрати зв'язку та необхідності вживати екстрених заходів для відновлення зв'язку та координації польотів.Авіаційні додатки повинні бути доступними цілодобово, 7 днів на тиждень. Навіть короткий відмов у роботі може призвести до серйозних наслідків для безпеки пасажирів та регулярності авіаперельотів.

Надійність є ключовим аспектом в обслуговуванні авіаційних додатків. Оскільки пасажирська безпека та ефективність авіаперельотів є найважливішими пріоритетами в авіації, додатки повинні працювати без відмов навіть у найважчих умовах та стрес-тестах. Важливо враховувати, що авіаційна галузь стикається з різноманітними чинниками ризику, включаючи погодні умови, технічні збої, людські помилки та інші непередбачені обставини.

Високий рівень надійності вимагає систем, які можуть виявити та усунути помилки, використовуючи резервні або резервовані системи, а також системи аварійного відновлення. Такі системи повинні бути розроблені та підтримуватися відповідно до найвищих стандартів та норм безпеки. Вони повинні мати здатність перехоплювати роботу в разі відмови головних систем та забезпечувати неперервність обслуговування.

1.3.2. Безпека і захист даних

Конфіденційність даних є критично важливою у відомостях, оброблюваних авіаційними додатками. Серед цих даних можуть бути важливі інформація про польоти, пасажирів, критичні системи літаків та комунікаційні записи. Якщо конфіденційна інформація потрапить у невірні руки, це може мати серйозні наслідки. Наприклад, доступ до даних про пасажирів та польоти може використовуватися для здійснення терористичних атак або злочинів. Крім того, конфіденційні дані про

технічний стан літаків можуть бути використані з метою вчинення кібератак або спроб збою обладнання.

Захист конфіденційності даних включає в себе шифрування даних під час їх передачі, контроль доступу до систем, двофакторну аутентифікацію та інші заходи безпеки. Забезпечення конфіденційності даних у авіаційних додатках є важливою складовою безпеки авіаційної галузі загалом.

Забезпечення цілісності даних означає, що дані не можуть бути змінені без відома та дозволу. У важливих авіаційних додатках, таких як системи навігації та системи управління бортовою авіонікою, надзвичайно важливо, щоб дані залишалися недоторканими.

Якщо дані будуть змінені невідомим чином або несанкціоновано, це може призвести до непередбачуваних наслідків. Наприклад, зміна маршруту літака або параметрів системи управління може викликати серйозні негаразди та загрожувати безпеці польоту.

Для забезпечення цілісності даних використовуються цифрові підписи, хеш-функції та системи контролю цілісності. Ці заходи допомагають виявити будь-які незаконні зміни в даних та негайно реагувати на них, щоб запобігти можливим наслідкам.

1.3.3. Інтеграція різних систем

Авіаційна галузь включає в себе широкий спектр різних типів літаків, від малих літаків загальної авіації до великих пасажирських літаків та військових винищувачів. Кожен тип літака може мати власні особливості та вимоги до обслуговування.

Наприклад, пасажирські літаки мають високі стандарти безпеки та комфорту для пасажирів, тому системи управління та моніторингу повинні бути дуже надійними. У військових літаків можуть бути встановлені спеціальні системи для бойових операцій, що також вимагають спеціалізованого обслуговування.

Крім того, різні покоління літаків можуть мати різні технологічні характеристики та стандарти зв'язку, що створює виклик для інтеграції різних систем.

У авіаційній галузі використовуються різні програмні продукти для керування та моніторингу повітряними судами. Це включає в себе програмне забезпечення для навігації, системи керування бортовою авіонікою, системи безпеки та багато інших.

Ці програми можуть бути розроблені різними виробниками та на різних мовах програмування, що робить інтеграцію та обслуговування складним завданням. Для забезпечення надійності та сумісності цих програм необхідно виконувати тестування та верифікацію на різних платформах та обладнанні.

Крім того, в авіаційній галузі діють регуляторні стандарти, що стосуються як обладнання, так і програмного забезпечення. Ці стандарти визначають вимоги до безпеки та надійності систем та додатків.

1.3.4. Збільшення обсягу даних

Швидкий обмін та обробка даними стали однією з найважливіших вимог для сучасних авіаційних додатків. Зростання обсягу даних, що генеруються та оброблюються у цій галузі, створює ряд викликів та необхідних рішень:

Забезпечення миттєвого обміну даними між різними компонентами системи стало невід'ємною частиною авіаційних додатків. Наприклад, навігаційні системи повинні отримувати оновлені дані про маршрути, погоду та повітряний рух у режимі реального часу. Повідомлення про стан літака, які надходять до керівництва та інших служб, повинні бути миттєвими та достовірними.

Обмін даними у великому масштабі може призвести до значного навантаження на мережу та інфраструктуру. Оптимізація пропускної здатності є важливою для забезпечення ефективного обслуговування. Це може включати в себе використання технологій стиснення даних, кешування та розподіленого обчислення.

Під час швидкого обміну даними важливо забезпечити їхню безпеку та захист від кібератак. Зловмисники можуть намагатися завдати шкоду авіаційним системам, перешкодити обміну даними або використовувати дізнану інформацію для злочинних цілей. Тому кібербезпека та шифрування даних є важливою частиною інфраструктури обслуговування авіаційних додатків.

Для забезпечення якості та надійності обміну даними, важливо мати системи моніторингу та логування. Це дозволяє виявляти проблеми в обміні даними, вживати необхідні заходи та аналізувати події для подальшого вдосконалення.

Швидкий обмін даними є критично важливим для забезпечення безпеки, ефективності та надійності авіаційних додатків. З врахуванням динаміки цієї галузі, постійного росту обсягу даних та вимог до реального часу, цей аспект обслуговування має велике значення для безпеки та продуктивності авіаційної галузі.

Ці проблеми і виклики вимагають постійної уваги та розробки нових технологій та методів для забезпечення безпеки, надійності та ефективності авіаційних додатків. Розробка спеціалізованих систем обслуговування є важливим кроком у вирішенні цих проблем і забезпеченні безпеки та успішності авіаційних операцій [10].

1.4. Вимоги до системи обслуговування додатків

Розробка комп'ютерної десктопної системи для обслуговування авіаційних додатків вимагає чіткого визначення функціональних вимог, які забезпечать її ефективну та безпечну роботу. Враховуючи проблеми та виклики, що існують в авіаційній галузі, далі наведено вимоги до систем обслуговування додатків.

1.4.1. Моніторинг та аналіз стану додатків

Система моніторингу та аналізу стану авіаційних додатків в авіаційній галузі відіграє критичну роль у забезпеченні безпеки та надійності повітряного транспорту. Нижче розглянемо докладніше вимоги та функціональність цієї системи:

1. Система повинна надавати можливість в реальному часі відстежувати статус роботи авіаційних додатків.

2. Вона має автоматично перевіряти наявність та доступність додатків, запускати їх, а також відслідковувати їхню продуктивність.

3. Система повинна контролювати обсяг використаної пам'яті, завантаження центрального процесора (ЦП), об'єму дискового простору та мережевого трафіку кожного авіаційного додатку.

4. Важливо виявляти випадки перевищення ресурсів та низької продуктивності, що можуть призвести до недостатньої ефективності додатку.

5. Система повинна здійснювати збір та агрегацію даних щодо роботи додатків з різних джерел. Це може включати в себе логи подій, вимірювання ресурсів та мережевий трафік.

6. Зібрані дані допомагають створити повну картину стану кожного додатку.

7. Система повинна використовувати аналітичні методи для виявлення аномалій та незвичайних паттернів у роботі додатків.

8. Аналізуючи дані, система може реагувати на незвичайні ситуації, що допомагає попередити можливі неполадки.

9. У випадку виявлення проблем або аномалій, система повинна надсилати сповіщення операторам або адміністраторам, щоб вони могли вжити відповідних заходів.

10. Швидка реакція на потенційні загрози допомагає запобігти серйозним неполадкам.

11. Система має забезпечувати неперервну роботу та гарантувати, що моніторинг і аналіз стану додатків доступні цілодобово, 7 днів на тиждень.

Загальною метою цієї системи є підтримка надійності, безпеки та ефективності авіаційних додатків. Шляхом моніторингу та аналізу стану додатків в реальному часі, ця система допомагає забезпечити безпеку пасажирів та ефективність авіаперельотів, а також вчасно реагувати на потенційні загрози.

1.4.2. Надійність і доступність

Забезпечення надійності та доступності системи обслуговування авіаційних додатків є критично важливим для авіаційної галузі. Нижче розглянемо, як система вирішує ці аспекти:

1. Система повинна бути розроблена та налаштована таким чином, щоб гарантувати її функціонування цілодобово, 7 днів на тиждень.

2. Це важливо, оскільки авіаційні додатки не можуть допускати відмов у роботі навіть на короткий період часу, оскільки це може покласти під загрозу безпеку пасажирів та нормальний режим авіаперельотів.

3. Система повинна мати вбудовані механізми аварійного відновлення, які дозволяють відновлювати роботу після непередбачених відмов або збоїв.

4. Це включає в себе можливість швидко відновлювати працездатність системи та додатків після аварійних ситуацій.

5. Для забезпечення надійності та доступності важливих даних система повинна мати механізми резервного зберігання.

6. Це включає в себе регулярне резервне копіювання та зберігання даних в безпечних місцях, що гарантує їх доступність у випадку аварій або втрати даних.

7. Для забезпечення надійності, система повинна моніторити свій власний стан та роботу авіаційних додатків.

8. В разі виявлення потенційних проблем чи збоїв, система повинна вчасно надсилати попередження адміністраторам для подальших дій.

9. Для забезпечення надійності система повинна піддаватися регулярному тестуванню та валідації, включаючи тестування на відмовостійкість та тести в аварійних сценаріях.

Загальна мета - гарантувати, що система обслуговування авіаційних додатків завжди готова до роботи, навіть у випадку непередбачених обставин чи аварійних ситуацій. Це є критично важливим аспектом для безпеки та надійності авіаперельотів.

1.4.3. Виявлення та виправлення несправностей

Забезпечення безперебійної роботи авіаційних додатків вимагає ефективної системи виявлення та виправлення несправностей. Важливими аспектами цього завдання є:

1. Автоматичне виявлення несправностей: Система повинна мати вбудовані механізми для автоматичного виявлення несправностей та помилок в роботі авіаційних додатків.Це включає в себе моніторинг ключових метрик та параметрів додатків для виявлення аномалій.

2. Автоматичне виправлення простих помилок: Деякі типи несправностей можуть бути автоматично виправлені системою без втручання оператора. Наприклад, автоматичне перезавантаження додатку після краху або відновлення підключення до бази даних.

3. Надання рекомендацій операторам: У випадках, коли несправність не може бути автоматично виправлена, система може надати операторам рекомендації щодо кроків для виправлення помилки. Це може включати в себе вказівки щодо перевірки конфігурації, логів або інших діагностичних дій.

4. Журнал Подій та Аналіз: Система повинна зберігати журнал подій, який фіксує всі несправності, помилки та події, пов'язані з роботою додатків. Аналіз журналу подій допомагає виявляти кореляції між різними подіями та допомагає вдосконалювати систему для запобігання подібним проблемам у майбутньому.

5. Тривалість та Продуктивність Виправлень: Важливо забезпечити швидкість та ефективність виправлення несправностей. Це допомагає знизити вплив помилок на роботу авіаційних додатків та забезпечує надійність системи.

Забезпечення виявлення та виправлення несправностей в реальному часі гарантує, що авіаційні додатки залишаються стабільними та надійними під час використання.

1.4.5. Захист від кібератак та несанкціонованого доступу:

Забезпечення кібербезпеки є однією з найбільш критичних аспектів системи обслуговування авіаційних додатків. Основними вимогами та практиками для забезпечення захисту від кібератак та несанкціонованого доступу є:

1. Аутентифікація та авторизація: Система повинна вимагати сильну аутентифікацію користувачів та авіаційних додатків. Доступ до системи повинен бути обмежений за принципом "лише необхідний доступ", щоб запобігти несанкціонованому використанню ресурсів.

2. Захист даних в передачі: Всі дані, які передаються між системою обслуговування та авіаційними додатками, повинні бути зашифровані для запобігання перехопленню та зміні даних під час передачі.

3. Система виявлення та захисту від атак: Система повинна мати вбудовану систему виявлення атак (*IDS*) для виявлення незвичайної або підозрілої активності. Якщо виявлено атаку, система повинна мати засоби автоматичного блокування або відхилення таких атак.

4. Оновлення та патчі: Система повинна регулярно оновлювати своє програмне забезпечення та застосовувати патчі для виправлення відомих вразливостей. Перевірка на оновлення та автоматичне встановлення патчів допомагають уникнути використання вразливостей зловмисниками.

5. Моніторинг та журналювання: Система повинна регулярно моніторити свою активність та веде журнал подій. Це допомагає виявляти підозрілу активність та проводити дослідження інцидентів кібербезпеки.

6. Безпека внутрішнього забезпечення: Особлива увага повинна бути приділена безпеці внутрішнього забезпечення, так як загрози можуть виникнути як ззовні, так і зсередини організації.

Забезпечення високого рівня кібербезпеки є обов'язковим для забезпечення безпеки та надійності авіаційних додатків у сучасній галузі авіації.

Ці вимоги до системи обслуговування додатків в авіації спрямовані на забезпечення найвищого рівня безпеки, доступності та надійності авіаційних додатків, що використовуються в галузі авіації. Вони допоможуть уникнути аварій та забезпечити безпечну роботу повітряного транспорту.

Висновки до розділу 1

У рамках даного розділу було проведено детальний огляд існуючих систем обслуговування додатків в авіаційній галузі. Розглянуті різні аспекти та характеристики існуючих систем, спрямованих на полегшення організації та взаємодії з конструкторською документацією, а також на підвищення ефективності технічного обслуговування та безпеки польотів.

Огляд показав, що на ринку існує різноманіття систем обслуговування, від простих до комплексних. Кожна з них має свої переваги та недоліки, а також власні характеристики, спрямовані на різні потреби авіаційних підприємств.

Важливим аспектом є те, що більшість існуючих систем акцентує на питання безпеки польотів та технічного обслуговування, враховуючи величезне значення цих аспектів у сфері авіації.

Детально розглянуті функціональні можливості систем, такі як збереження та організація документації, можливості комунікації та обміну інформацією, відображення стану повітряних суден та впровадження штучного інтелекту для аналізу та виправлення помилок.

Велика увага приділяється можливостям інтеграції з іншими авіаційними системами, що сприяє створенню єдиної та злагодженої інфраструктури для оптимального управління та обслуговування авіаційним обладнанням.

Огляд підкреслив актуальність розробки нових систем, зокрема спеціалізованої десктопної системи для обслуговування авіаційних додатків. Це відкриває нові перспективи для покращення ефективності та безпеки в авіації.

У цілому, огляд існуючих систем обслуговування додатків в авіації є важливим етапом у розумінні та аналізі потреб галузі, а також визначенні перспектив розвитку для оптимізації авіаційних процесів.

РОЗДІЛ 2
ТЕХНОЛОГІЇ ДЛЯ РОЗРОБКИ ДЕСКТОПНИХ ДОДАТКІВ

Розробка десктопних додатків – це процес створення програмного забезпечення, яке призначене для використання на персональних комп'ютерах або ноутбуках з локальною операційною системою. Враховуючи широкий спектр функціональності та можливостей, десктопні додатки можуть включати в себе графічні інтерфейси, роботу з базами даних, мережеві можливості та інші комплексні функції. Розробка десктопних додатків є важливою складовою сучасної інформаційної технології. Вибір правильних технологій для реалізації проекту має вирішальне значення для успішної розробки та задоволення потреб користувачів. У цьому розділі дипломної роботи розглянемо різні мови програмування та фреймворки, які можуть бути використані для створення десктопних додатків.

2.1. *Java* та *JavaFX*

Java є високорівневою об'єктно-орієнтованою мовою програмування, розробленою компанією *Sun Microsystems* (тепер власність *Oracle*). Її крос-платформеність дозволяє використовувати один і той же вихідний код на різних операційних системах, що полегшує розробку та підтримку додатків. Велика активність спільноти та розгалужена історія розвитку забезпечують стабільність та надійність мови [11].

JavaFX - це фреймворк для створення графічних інтерфейсів у Java-додатках. З'єднуючи *Java* з *JavaFX*, розробники отримують комплексний інструментарій для створення десктопних додатків з привабливим та інтуїтивно зрозумілим інтерфейсом. Однією з ключових переваг *JavaFX* є його вбудована підтримка анімації та високоякісна графіка, що дозволяє створювати інтерактивні та естетично приємні додатки.

Основні переваги *Java* та *JavaFX*:

- Крос-платформеність: Можливість використовувати один код на різних операційних системах (*Windows, Linux, macOS*).

- Надійність та стабільність: *Java* славиться своєю стійкістю та безпекою, що робить її вибором для великих корпоративних проектів.

- Широка спільнота та екосистема: Велика кількість розробників та багатий вибір бібліотек та інструментів.

- Можливості *JavaFX* для *UI*: Інтуїтивний дизайн інтерфейсу, багатофункціональність та анімація для створення привабливих додатків.

- Відкритий код: *Java* є відкритою мовою програмування, що дозволяє розробникам вносити свої внески та активно співпрацювати у великій спільноті.

Java та *JavaFX* широко використовуються для розробки корпоративних додатків, систем управління, ігор та інтерактивних мультимедійних додатків. Їхні можливості та простота використання роблять їх ідеальним вибором для різних областей розробки програмного забезпечення [12].

Структура додатку на *JavaFX* включає ряд ключових елементів, що організовують код та забезпечують ефективну розробку. Основні компоненти структури додатку *JavaFX* включають:

1. Головний клас (*Main Class*):

Package: Додаток повинен бути організований в пакет (*package*) для логічної групування класів.

Main Method: Головний клас повинен містити метод main, який викликається при запуску додатку. Цей метод ініціалізує *JavaFX* та запускає основне вікно додатку.

2. Файли макетування (*FXML*):

JavaFX використовує *FXML* для визначення графічного інтерфейсу за допомогою *XML*-подібного синтаксису.

FXML-файли можуть містити візуальні елементи, властивості та контролери.

3. Контролер (*Controller*):

Контролер відповідає за логіку взаємодії між *FXML* та *Java*-кодом.

Методи контролера повинні відповідати подіям від елементів інтерфейсу.

4. Макет додатку (*Application Layout*):

Визначення макету додатку, який може включати головне вікно, меню, панелі інструментів тощо.

5. Ресурси та Стилі (*Resources & Styles*):

Використання зовнішніх ресурсів, таких як зображення, шрифти та файли стилів.

Визначення стилів для покращення вигляду інтерфейсу.

На рис 2.1. показано структуру типової програми *JavaFX FXML*.

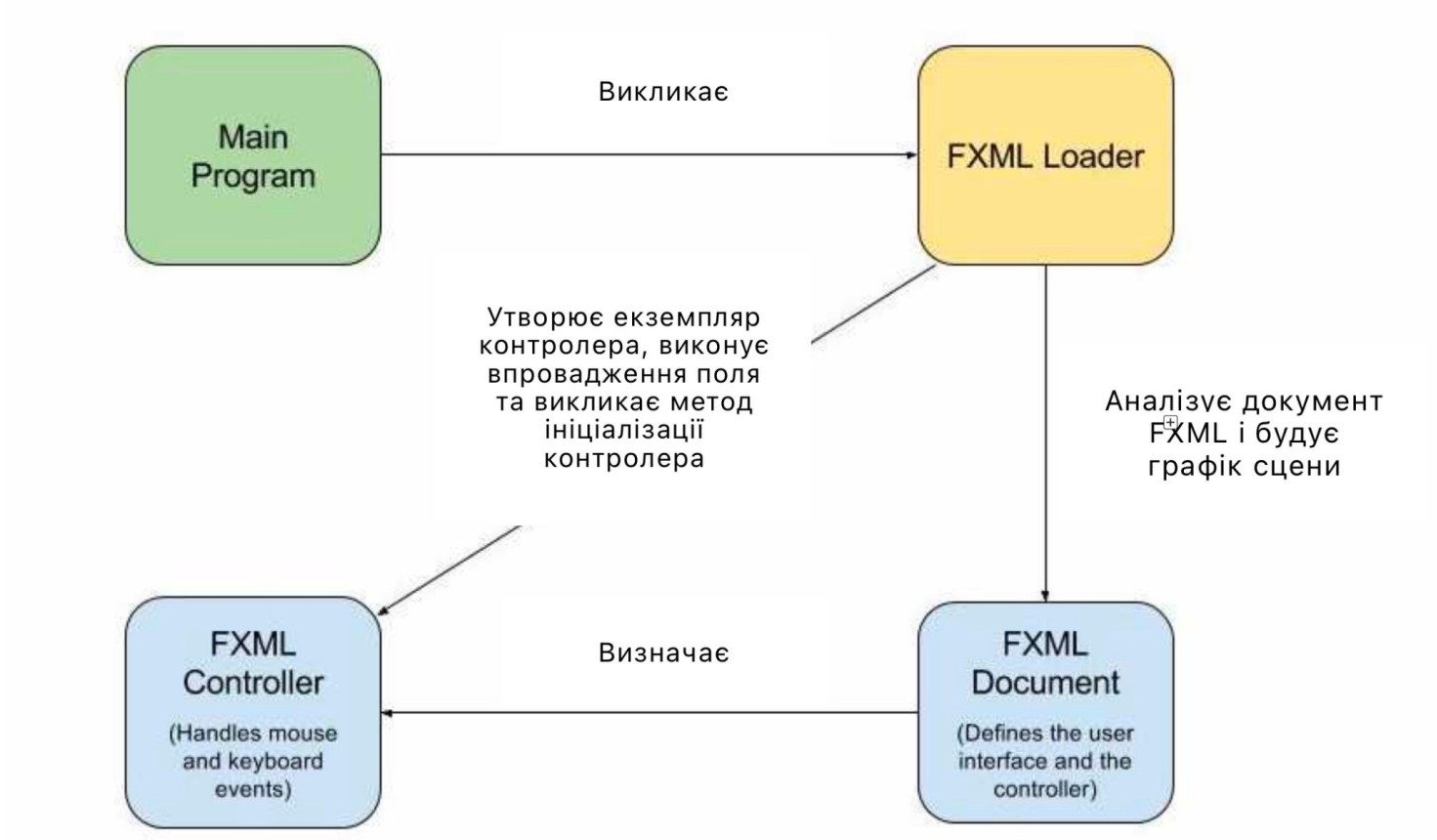


Рис. 2.1. Структуру типової програми *JavaFX FXML*

Структура додатку *JavaFX* може змінюватися в залежності від розміру та складності проекту, але основні компоненти завжди включають головний клас, *FXML*-файли, контролери та макет додатку [13].

2.2. *C++* та *Qt*

Мова *C++* є загальнодоступною мовою програмування, визначеною своєю високою продуктивністю та ефективністю. Вона дозволяє розробникам писати ефективний та масштабований код для різних застосунків. Заснована на мові *C*, *C++* включає об'єктно-орієнтовані можливості, що полегшують організацію та розширення коду.

Qt є крос-платформеним фреймворком для розробки програм, який надає набір інструментів для створення графічного інтерфейсу, роботи з мережею, базами даних, обробки подій та багато іншого. Розроблений компанією *Qt Company*, *Qt* визначається своєю крос-платформеністю, що дозволяє розробляти додатки, які працюють на різних операційних системах, таких як *Windows*, *Linux*, *macOS*.

Основні переваги *C++* та *Qt*:

- Крос-платформеність: Обидві технології (*C++* та *Qt*) надають можливість розробки додатків, що працюють на різних платформах, забезпечуючи високу переносимість коду.
- Ефективність та Продуктивність: *C++* дозволяє створювати ефективний та швидкий код, що робить його відмінним для розробки десктопних додатків з вимогливою продуктивністю.
- Об'єктно-орієнтованість: Використання об'єктно-орієнтованого підходу в C++ дозволяє створювати добре структуровані та легко розширювані додатки.
- Широкий Функціонал *Qt*: *Qt* надає готові рішення для створення інтерфейсів користувача, обробки подій, роботи з мережею, базами даних, що спрощує розробку та полегшує тестування.
- Спільнота та Документація: Як в *C++*, так і в *Qt* існує активна спільнота розробників та докладна документація, що полегшує вивчення та вирішення проблем.

Сполучення *C++* та *Qt* широко використовується для розробки десктопних додатків у різних областях, таких як програми для дизайну, графіки, ігри, системи керування базами даних, наукові та інженерні додатки. Крос-платформеність та

потужність цих технологій роблять їх відмінним вибором для великих та вимогливих проектів [14].

Структура та взаємодія між компонентами при роботі з *Qt* у *C++*:

1. Головний файл програми (*main.cpp):* Створення об'єкту *QApplication*. Створення та налаштування головного вікна (*QMainWindow* або власного класу). Запуск головного циклу за допомогою *QApplication::exec*().

2. Графічний інтерфейс (*mainwindow.ui, mainwindow.cpp, mainwindow.h*): Опис інтерфейсу в *Qt Designer* або в коді. Використання видів (*QWidgets*) для створення елементів інтерфейсу. Встановлення логіки та взаємодії через сигнали та слоти.

3. Бізнес-логіка (*mywidget.cpp, mywidget.h*): Створення власних класів, які виконують бізнес-логіку. Використання об'єктів *Qt* для обробки подій та взаємодії з інтерфейсом.

4. Система моделей та видів (*model.cpp, model.h, view.cpp, view.h*): Створення моделей для управління даними. Створення видів для відображення даних згідно із моделлю.

5. Сигнали та слоти (*QObject*): Використання механізму сигналів та слотів для взаємодії між об'єктами. Підписка на сигнали та підключення слотів для реалізації взаємодії.

6. Крос-платформовість та абстракції (*QPlatform, QAbstractClass*): Використання абстракцій та класів *Qt* для забезпечення крос-платформовості. Використання спеціалізованих класів, таких як *QIODevice* для роботи з файлами або *QTcpSocket* для мережевого з'єднання.

2.3. *Python* та *Tkinter*

Python – це високорівнева, інтерпретована, об'єктно-орієнтована мова програмування загального призначення. Вона визначається своєю простотою синтаксису та акцентом на читабельність коду, що робить її ідеальним вибором для

розробки широкого спектру застосунків, включаючи веб-розробку, наукові обчислення, штучний інтелект, автоматизацію завдань та багато іншого.

Python використовується як для розробки невеликих скриптів, так і для великих проектів, завдяки своїй гнучкості та великому екосистемі бібліотек і фреймворків, що розширюють його можливості.

Tkinter - це стандартна бібліотека для розробки графічного інтерфейсу користувача (*GUI*) в мові програмування *Python*. Назва "*Tkinter*" походить від інструменту "*Tk Toolkit*", який вона використовує для створення вікон та інших елементів інтерфейсу.

Python разом із бібліотекою *Tkinter* володіє привабливими особливостями, роблячи їх відмінним вибором для швидкого розроблення простих десктопних додатків:

1. Синтаксис та простота вивчення: *Python* відомий своїм читабельним синтаксисом, що робить його дружелюною мовою для початківців. Простота вивчення дозволяє новачкам швидко освоювати мову та розпочинати програмування.

2. Велика спільнота та багата екосистема: *Python* має активну та велику спільноту розробників. Це означає наявність безлічі бібліотек, модулів та фреймворків, які полегшують розробку різноманітних застосунків.

3. Крос-платформовість: Програми, написані на *Python*, легко переносяться між різними операційними системами без змін у вихідному коді. Це дозволяє створювати крос-платформові додатки.

4. Вбудована бібліотека для *GUI*: Tkinter є вбудованою бібліотекою *Python* для розробки графічного інтерфейсу користувача (*GUI*). Вона має простий та зрозумілий інтерфейс для створення вікон, кнопок, полів введення та інших елементів *GUI*.

5. Простота та легкість використання: *Tkinter* відрізняється легкістю використання та невеликим порогом входу. Створення простого вікна або діалогового вікна можливе всього кількома рядками коду.

6. Можливості кастомізації: Хоча *Tkinter* є простою бібліотекою, вона надає достатньо можливостей для кастомізації вигляду та поведінки елементів інтерфейсу.

7. Документація та підтримка: *Tkinter* має докладну документацію та велику кількість ресурсів для вивчення, що робить її привабливим вибором для новачків у галузі розробки *GUI* [15].

Загалом, *Python* та *Tkinter* становлять привабливий стек технологій для розробки швидких та ефективних десктопних додатків, зокрема для початківців та проектів з обмеженим бюджетом часу.

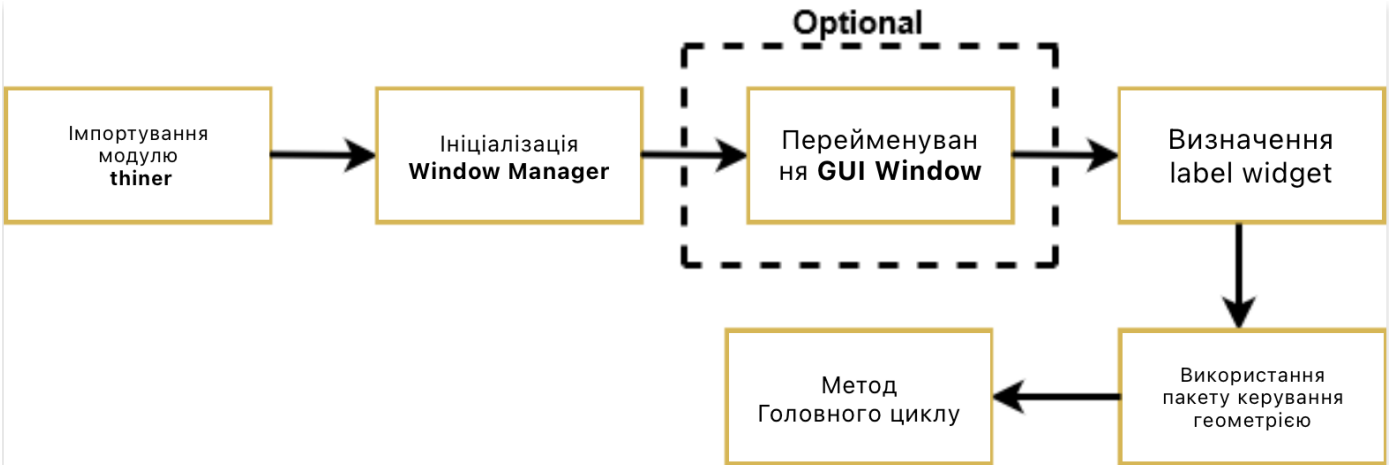


Рис. 2.2. Блок-схема графічного інтерфейсу за допомогою *Tkinter*.

Python є потужною та високорівневою мовою, яка відома своєю читабельністю та простотою використання. Це робить його чудовим вибором для розробки десктопних додатків, особливо з урахуванням широкого спектру бібліотек і фреймворків.

Tkinter, як стандартний пакет для розробки графічного інтерфейсу користувача (*GUI*) в *Python*, пропонує швидкий та досить простий спосіб створення вікон, кнопок, полів введення та інших елементів інтерфейсу (див. рис. 2.2). Враховуючи легкість використання та наявність документації, Tkinter стає зручним вибором для швидкого створення простих десктопних додатків.

Враховуючи зручність та широкі можливості мови програмування *Python* разом з *Tkinter*, можна зробити висновок, що цей стек технологій дозволить створити ефективний та легко збережений десктопний додаток. Оптимальний вибір залежить від конкретних потреб проекту та рівня досвіду команди розробників.

2.4. *Electron* (*JavaScript, HTML, CSS*)

Electron використовує веб-технології для розробки крос-платформених десктопних додатків. *JavaScript*, *HTML* та *CSS* дозволяють легко створювати додатки для різних операційних систем, але це може призвести до вищого споживання ресурсів.

JavaScript, *HTML*, *CSS*: *Electron* використовує три основні веб-технології: JavaScript для логіки додатку, *HTML* для розмітки та *CSS* для стилізації. *JavaScript* є мовою програмування, яка надає динамічність та інтерактивність, *HTML* використовується для створення структури сторінки, а *CSS* - для надання їй стилів та вигляду.

Electron - це фреймворк, який дозволяє використовувати веб-технології для розробки крос-платформених десктопних додатків. Він базується на двох основних складових: *Chromium* для відображення вмісту та *Node.js* для взаємодії з операційною системою та роботи з файловою системою [16].

Основні Переваги:

- Крос-платформеність: Завдяки використанню веб-технологій, додатки, створені на Electron, можуть працювати на різних операційних системах, включаючи *Windows*, *macOS* та *Linux*.

- Розширені можливості *JavaScript*: Використання *JavaScript* дозволяє легко реалізовувати динамічний та інтерактивний інтерфейс, що сприяє високій користувацькій привабливості.

- Спрощений доступ до системних ресурсів: *Electron* надає *API* для взаємодії з операційною системою, файловою системою та іншими системними ресурсами.

- Активна спільнота та розширення: Існує активна спільнота розробників, а також широкий вибір розширень та плагінів для *Electron*, що полегшує розширення функціоналу додатків.

Electron використовується для розробки різноманітних десктопних додатків, включаючи:

- Месенджери: Додатки для обміну повідомленнями, такі як *Slack* та *Discord*, використовують *Electron*.
- Редактори Коду: Популярні редактори коду, такі як *VSCode*, побудовані на *Electron*.
- Медіаплеєри: Деякі медіаплеєри та стрімінгові сервіси використовують Electron для створення нативних додатків.
- Нотатки та Органайзери: Додатки для ведення нотаток, організації завдань та календарі, які працюють на різних платформах.

З розробницької точки зору, додаток на основі *Electron* суттєво є додатком *Node.js*. *Electron* використовує основний файл, визначений у вашому файлі package.json, і виконує його. Цей основний файл створює вікна додатку, які містять відображені веб-сторінки та взаємодію зі стандартним графічним інтерфейсом вашої операційної системи.

При запуску додатку за допомогою *Electron* створюється основний процес. Цей основний процес відповідає за взаємодію із стандартним графічним інтерфейсом операційної системи. Він створює графічний інтерфейс вашого додатка.

Просте запускання основного процесу не створює для користувачів вашого додатку жодного вікна. Вони створюються основним процесом у основному файлі за допомогою модуля *BrowserWindow*. Кожне вікно браузера потім запускає свій власний процес відображення. Процес відображення бере *HTML*-файл, який посилається на звичайні файли *CSS, JavaScript*, зображення і т.д., та відображає його у вікні [17].

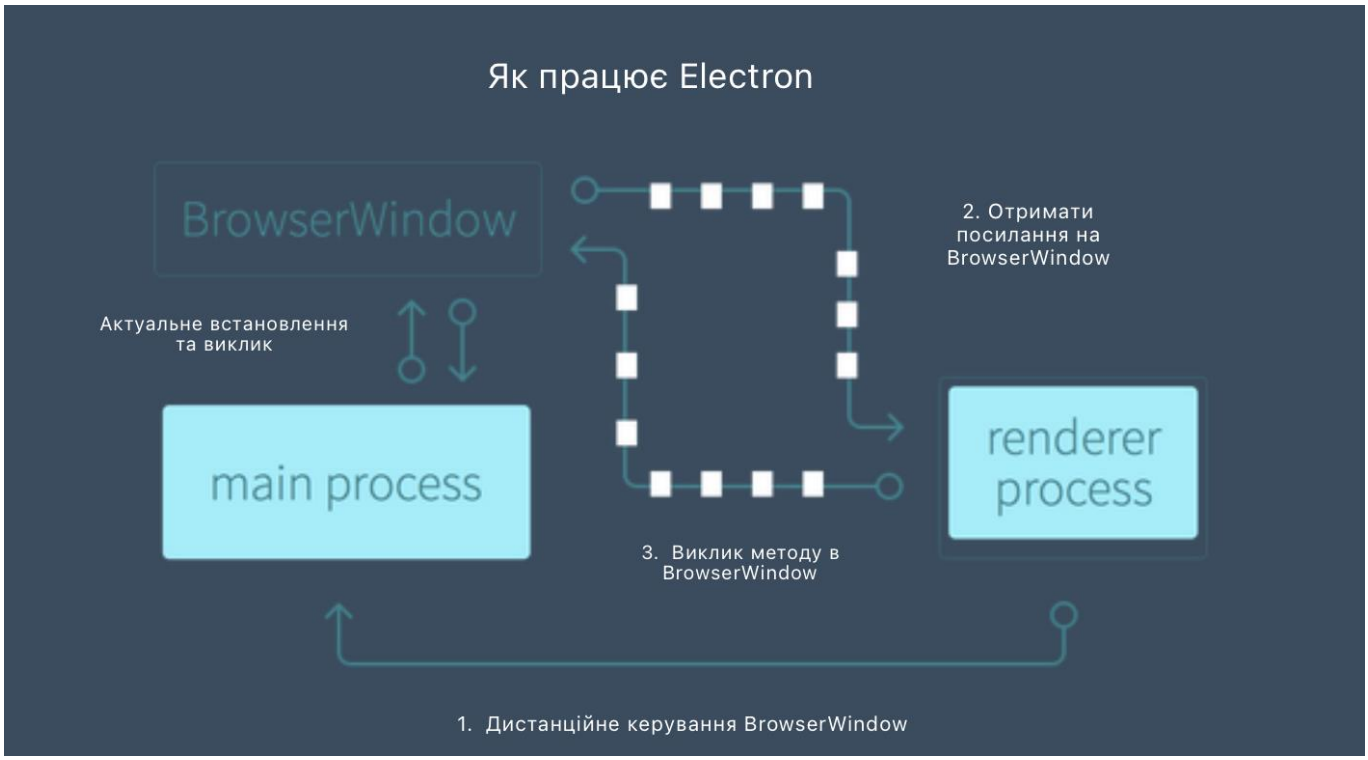
Основний процес може отримати доступ до стандартного графічного інтерфейсу за допомогою модулів, доступних безпосередньо в *Electron*. Десктопний додаток може отримувати доступ до всіх модулів *Node*, таких як модуль файлової

системи для роботи з файлами, запит для виконання *HTTP*-викликів і т.д. На рисунку 2.3 наведена схема роботи додатків на *Electron*.

Рис. 3.3. Схема роботи *Electron*

Далі описані принципи роботи *Electron.js*:

1. Архітектура та компоненти:



Main Process: Основний процес, що відповідає за вікно додатку та взаємодію з операційною системою.

Renderer Process: Відповідає за відображення інтерфейсу користувача та виконання веб-контенту в окремих вкладках.

2. Використання *Web*-Технологій:

HTML, CSS, JavaScript: Всі елементи інтерфейсу та логіка додатку базуються на веб-технологіях, що дозволяє використовувати звичайні веб-ресурси.

3. Відображення контенту:

Chromium Engine: Використовує движок *Chromium* для відображення веб-сторінок та взаємодії з *DOM*.

4. Взаємодія з операційною системою:

Node.js Integration: Інтеграція з *Node.js* дозволяє використовувати системні ресурси та модулі.

5. Мінімізація використання ресурсів:

Одноразові процеси: Використання окремих процесів для кожної вкладки дозволяє мінімізувати вплив на продуктивність.

6. Забезпечення безпеки:

Sandboxing: Використання механізму пісочниці для ізоляції процесів та запобігання вразливостям.

7. Системні *API*:

API для взаємодії з ОС: Electron надає *API* для взаємодії з операційною системою, такими як доступ до файлової системи, взаємодія з принтером, тощо.

8. Збереження даних:

Local Storage: Можливість використання локального сховища для збереження даних між сеансами.

9. Можливості розширення:

Плагіни та розширення: *Electron* дозволяє використовувати розширення для розширення функціоналу додатку.

10. Пакування та розгортання:

Розгортання у вигляді прикладного програмного забезпечення: *Electron* дозволяє пакувати додаток для різних операційних систем використовуючи засоби пакування.

11. Використання *Electron API*:

IPC (*Inter-Process Communication*): Взаємодія між основним та відображувальними процесами за допомогою *IPC*.

12. Розробка та налаштування:

Electron CLI та *DevTools*: Використання інструментів командного рядка та DevTools для розробки та налагодження додатків.

Electron надає зручний спосіб для розробників об'єднувати силу веб-технологій з можливостями крос-платформеного десктопного програмування. Підсумовуючи огляд принципів роботи Electron.js, можна зробити висновок, що цей фреймворк є

потужним інструментом для розробки десктопних додатків, особливо тих, які використовують веб-технології. Використання *HTML*, *CSS* та *JavaScript* дозволяє розробникам створювати додатки, які легко взаємодіють з веб-ресурсами та виглядають схоже на веб-сторінки. *Electron* дозволяє розробникам швидко створювати десктопні додатки, використовуючи відомі веб-інструменти та ресурси. Підтримка плагінів та розширень дозволяє розробникам розширювати функціонал своїх додатків та адаптувати їх до конкретних потреб. Використання механізму пісочниці та ізоляція процесів додає додатковий рівень безпеки та захисту від вразливостей. Завдяки засобам пакування, додатки, створені з використанням Electron, легко розгортаються на різних операційних системах, що робить їх доступними для широкого кола користувачів. Інтеграція з *Node.js* дозволяє взаємодіяти з системними ресурсами та модулями. Використання *Electron CLI* та *DevTools* забезпечує зручні інструменти для розробки та налагодження додатків.

Загалом, *Electron* відкриває нові можливості для створення сучасних та ефективних десктопних додатків, що забезпечують високий рівень користувацького досвіду та функціональності.

2.5. *Swift* та *macOS/iOS SDK*

Swift є мовою, створеною спеціально для розробки додатків для платформ *Apple*. Вона визначається своєю об'єктно-орієнтованою природою та вбудованими засобами безпеки, що робить її ефективною та надійною.

Swift взаємодіє з розширеним фреймворком розробки, який відомий як *macOS/iOS SD*K. Цей фреймворк забезпечує розробникам повний спектр інструментів для створення не лише привабливого графічного інтерфейсу, але й обробки подій та інших функціональних аспектів. Це стає можливим завдяки використанню Xcode, потужного інтегрованого середовища розробки, яке надає розширені засоби для творчого процесу програмістів [18].

Розглядаючи аспекти розробки додатків для пристроїв *Apple*, слід визначити основні переваги використання *Swift* та *macOS/iOS SDK*. Це включає нативний вигляд

та відчуття додатків, великий вибір готових рішень для розробки та активну спільноту розробників, що сприяє обміну досвідом та розвитку.

Застосування відповідної технічної структури – це не лише створення програмного продукту, а й інтеграція з великою аудиторією користувачів. Це стає можливим завдяки популярності продукції *Apple* та відмінній роботі *Swift* разом із *macOS/iOS SDK*. Такий підхід стає ключем до успіху в розробці великих та високотехнологічних проектів, які відповідають сучасним вимогам та стандартам.

Swift має простий синтаксис, що полегшує вивчення, та високу продуктивність, що робить її зручною для розробки десктопних та мобільних додатків. Як офіційна мова для розробки додатків для *macOS* та *iOS, Swift* забезпечує глибоку інтеграцію з іншими технологіями *Apple*.

Основні переваги:

- Нативний вигляд та відчуття: Розробка на *Swift* та з використанням *macOS/iOS SDK* дозволяє створювати додатки, які інтегруються зі стандартним дизайном та функціональністю *Apple*.
- Велика кількість готових рішень: Існує багато готових компонентів та бібліотек, що полегшують розробку та розширення функціоналу додатків.
- Широка аудиторія користувачів: З огляду на популярність продукції *Apple*, додатки, розроблені на *Swift*, можуть отримати доступ до широкої аудиторії користувачів.

Застосування:

- Десктопні додатки для *macOS*: Розробка професійних програм, графічних редакторів, інструментів продуктивності.
- Мобільні додатки для *iOS*: Створення ігор, соціальних мереж, бізнес-додатків та інших мобільних рішень [19].

Основною концепцією *Clean Swift* є *ViewController/Interactor /Presenter (VIP)*. Його робочий процес виглядає так (рис. 2.4): *ViewController* надсилає запит *Interactor* для управління бізнес-логікою. Після цього *Interactor* відправляє відповідь Presenter для управління логікою відображення з метою підготовки даних до використання. Потім ці дані передаються *ViewController* для відображення.

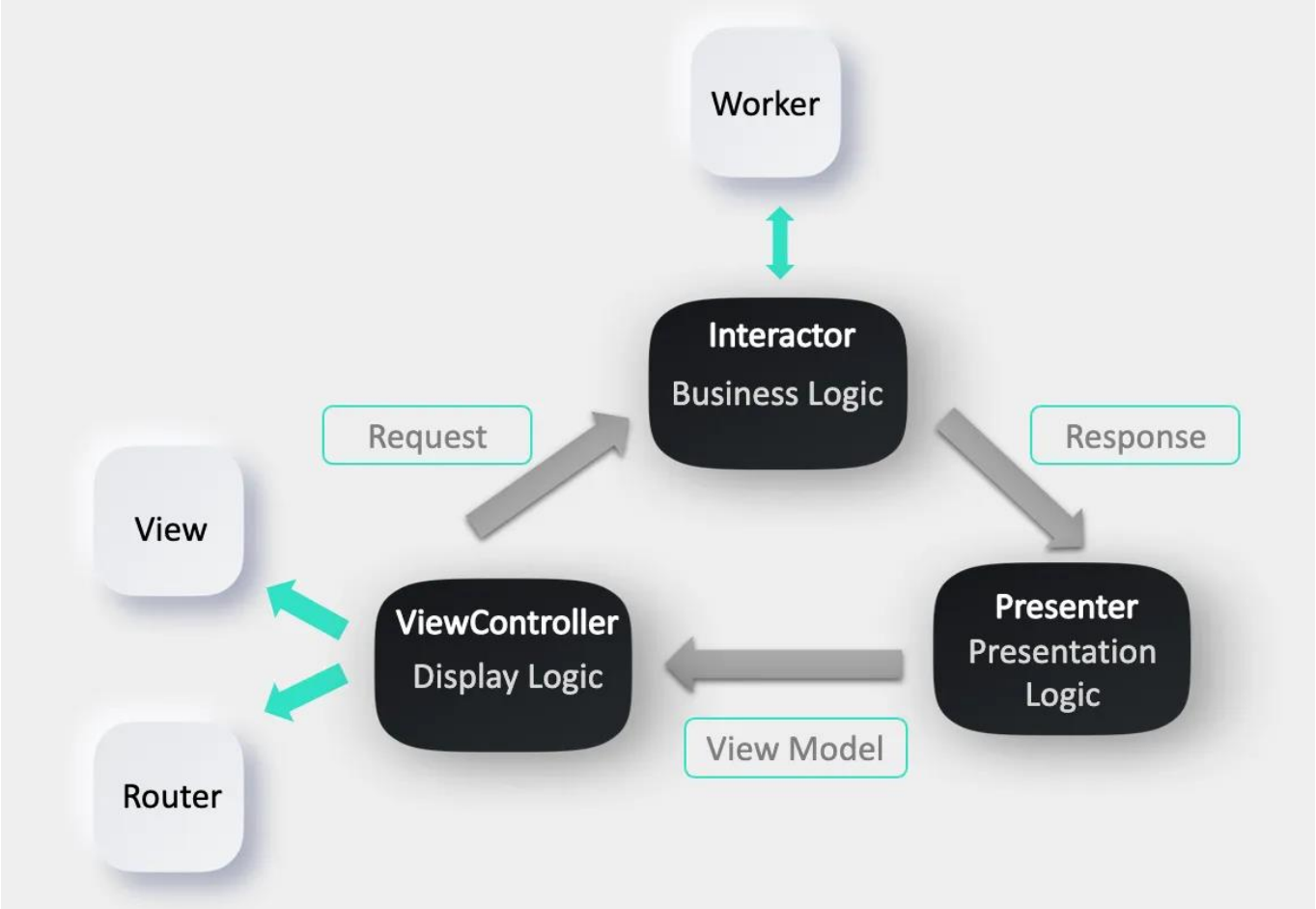


Рис. 2.4. Робочий процес *Swift*

Висновок до розділу 2

У цьому розділі проведено аналіз та порівняння різних технологій для розробки десктопних додатків, таких як *JavaFX*, *Qt*, *Tkinter*, *Electron* та *macOS/iOS SDK*. *Java* та *JavaFX* визначаються високою кросплатформенністю, що надає можливість створення зручних інтерфейсів. Проте, вимагають значної кількості коду. *Qt* славиться своєю потужністю та ефективністю, але може бути складним для новачків. *Tkinter* є простим та легким інструментом для десктопних додатків, але менш потужним порівняно з іншими технологіями. *Electron* вирізняється великою кросплатформенністю, активною спільнотою та зручністю використання веб-технологій, що робить його привабливим для розробників. *Swift* використовується для розробки додатків для *macOS* та *iOS*, ідеально підходить для екосистеми *Apple*.

Вибір технології для розробки десктопного додатку здійснено на користь *Electron* через його високу кросплатформенність, швидкість розробки та зручність використання веб-технологій. Ця технологія відповідає вимогам проекту та забезпечує необхідну гнучкість у розробці додатків для різних операційних систем.

РОЗДІЛ 3
АНАЛІЗ ПОТРЕБ АВІАЦІЙНОЇ ГАЛУЗІ

3.1. Огляд поточних вимог та проблем авіаційних підприємств

Авіаційні підприємства стикаються з рядом вимог і проблем, які визначають їхню діяльність та розвиток. Розглянемо ключові аспекти цього питання:

1. Однією з головних вимог в авіації є забезпечення безпеки. Підприємства повинні дотримуватися високих стандартів безпеки польотів, технічного обслуговування літаків та інших аспектів авіаційної діяльності.

2. В умовах зростання обсягів авіаперевезень важливо оптимізувати процеси та ресурси, забезпечуючи ефективне використання літаків, пального та інших ресурсів.

3. Авіаційні підприємства мають справу з великим обсягом технічної документації. Системи збереження та управління цією документацією повинні бути стандартизовані та ефективні.

4. Сучасні стандарти екології вимагають від авіапідприємств дотримання екологічних норм та впровадження ефективних практик для зменшення впливу авіації на довкілля.

5. Використання сучасних цифрових технологій, таких як Інтернет речей (*IoT*), штучний інтелект (*AI*) та аналітика даних, може покращити моніторинг, обслуговування та безпеку польотів.

6. Покращення взаємодії між різними частинами авіаційної системи, такими як аеропорти, авіаперевізники та контрольно-пропускні пункти, може сприяти ефективнішому функціонуванню системи в цілому.

7. Авіаційні підприємства повинні бути гнучкими та адаптабельними до змін в економічному, політичному та технічному середовищі.

8. Забезпечення належного навчання та кваліфікації персоналу є важливим завданням для підтримання високих стандартів безпеки та ефективності.

Загальною метою авіаційних підприємств є постійне вдосконалення та відповідь на сучасні вимоги, щоб забезпечити безпеку, ефективність та стійкість авіаційної діяльності в умовах постійних змін.

Авіаційна галузь є високотехнологічною і вимагає величезного обсягу технічної документації для конструювання, виробництва та експлуатації літаків і пов'язаних систем. У сфері авіації використовуються різні типи документів, включаючи: креслення, схеми, технічні специфікації, інструкції.

Виготовлення літаків – це складний процес, що вимагає тисячі годин розробки та величезний обсяг технічної документації. Кожен літак складається з тисяч деталей, кожна з яких має відповідні технічні характеристики.

Збереження та управління цією великою кількістю документів - це завдання високого рівня складності. Важливо мати ефективні системи архівування, каталогізації та доступу до інформації.

Технічна документація повинна бути точною та актуальною. Навіть найдрібніші помилки чи застарілі дані можуть призвести до серйозних проблем під час виробництва чи експлуатації.

Авіаційна галузь характеризується величезним обсягом технічної документації. Це створює виклик у збереженні, управлінні та знаходженні необхідної інформації великою кількістю документів.

Комунікація в авіаційній галузі також відіграє ключову роль у розробці, виробництві та експлуатації літаків. Наочно це виявляється у великій кількості інженерів, проектних груп та інших учасників, які взаємодіють під час різних етапів робочого процесу.

Проектування літака включає в себе велику кількість інженерів, які співпрацюють для створення детальних креслень, розробки схем та визначення технічних специфікацій. Інженери повинні чітко спілкуватися щодо технічних вимог, зокрема вказівок щодо матеріалів, аеродинамічних характеристик та функціональності.

Великі виробництва включають в себе декілька департаментів, таких як виробництво, якість та логістика, повинні ефективно співпрацювати для забезпечення гладкого процесу збирання.

Ефективна комунікація в авіаційній галузі є критично важливою для успішної розробки, виробництва та експлуатації літаків. Брак зручних інструментів та систем для обміну думками та обговорення документації може призвести до серйозних проблем, вплинути на якість та безпеку літаків, а також призвести до затримок у виробництві та експлуатації. Ефективна комунікація в авіаційній галузі має стратегічне значення для запобігання проблемам, забезпечення високої якості проектів та збереження безпеки в усіх етапах виробництва та експлуатації літаків.

Проекти в авіаційній галузі часто виконуються командами інженерів, що вимагає ефективного засобу спільного редагування та обміну інформацією між учасниками. Шлях до ефективного спільного редагування та обміну інформацією в проектах авіаційної галузі полягає в інтеграції сучасних інструментів, які сприяють колективній роботі, структурованій комунікації та оптимізації процесів розробки.

Тобто в цьому пункті було визначені три проблеми які потребують удосконалення: об'єм та складність документації, необхідність комунікації та колективна робота. Послідовне вирішення цих проблем допоможе створити ефективну та спрощену виробничу екосистему в авіаційній галузі. Застосування інноваційних інструментів та підходів до організації виробництва дозволить досягти високих стандартів якості, безпеки та ефективності в розробці та експлуатації літаків.

3.2. Аналіз можливостей удосконалення обслуговування додатків

Провівши ретельний аналіз потреб авіаційної галузі, наша команда визначила низку важливих проблем, які впливають на ефективність та оперативність цього сектору. Серед цих викликів особливо виділяються об'єм та складність документації, а також необхідність оптимізованої комунікації та ефективної колективної роботи. Ці аспекти є важливими факторами для успішного функціонування авіаційного сектору, і їх вирішення може значно поліпшити процеси у цій галузі.

У даному розділі обгрунтовано та розглянуто стратегічний підхід до вирішення проблеми значного обсягу технічної документації в контексті розробки та впровадження ефективної системи зберігання. Запропоновано комплексний план дій, спрямований на створення структурованого та ефективного інформаційного середовища для забезпечення оптимального доступу та управління різноманітною технічною документацією.

Першочерговим етапом вирішення проблеми є розробка системи категоризації та структуризації документів. Ця система буде базуватися на логічних категоріях та підкатегоріях, що сприятиме більш ефективній організації та навігації в базі даних. Додатково, використання метаданих та тегів забезпечить точну ідентифікацію та класифікацію документів, що є важливим елементом оптимізації пошуку.

Інтерфейс користувача розроблятиметься таким чином, щоб забезпечити інтуїтивну зрозумілість та легкість орієнтації у структурі системи зберігання. Додатково буде введено можливість створення персоналізованих списків, закладок та шляхів, що підвищить швидкість навігації між ключовими документами.

З метою оптимізації пошуку, будуть розроблені та впроваджені алгоритми, спроможні забезпечити швидке та точне визначення потрібних документів. Використання передових технологій штучного інтелекту та машинного навчання дозволить автоматизувати процес пошуку та підвищити релевантність результатів.

Окрема увага буде приділена системі автоматичного оновлення для забезпечення постійної актуальності документації. Регулярне архівування старих версій документів буде впроваджено з метою підтримки порядку та легкості в системі зберігання.

У підсумку, наведений стратегічний план вирішення проблеми обсягу технічної документації розглядається як ефективний та науково обґрунтований метод для підвищення ефективності управління інформаційним ресурсом в контексті авіаційної галузі.

З метою подолання необхідності комунікації та підвищення колективної ефективності в авіаційній галузі, пропонується впровадження системи коментарів та обговорень, спеціально адаптованої для конструкторської документації.

1. Інтерактивні коментарі та система обговорень: Розробка інтерактивної системи коментарів, що дозволяє інженерам долучати коментарі безпосередньо до конкретних частин конструкторської документації. Забезпечення можливості взаємодії між учасниками через відповіді на коментарі, створюючи таким чином ефективний діалог між членами команди.

2. Інструменти для колективної роботи: Розробка інструментів, що дозволяють кільком інженерам одночасно працювати над тим самим документом, забезпечуючи синхронізацію та відслідковування змін. Введення можливості створення спільних проектів та груп, що сприятиме ефективній колективній роботі та обміну ідеями між учасниками.

Розробка ефективних інструментів для зберігання та обговорення конструкторської документації представляє собою стратегічний компонент, спрямований на підвищення продуктивності та оптимізацію робочих процесів інженерів в авіаційній галузі. Це інноваційне рішення, яке висувається як ключовий крок у забезпеченні ефективної комунікації та спільної роботи на етапі розробки та експлуатації повітряних систем [20].

Активна комунікація та колективна робота визначаються як невід'ємні елементи стратегії зменшення ризиків та уникнення можливих затримок у виробництві та експлуатації. Швидке виявлення проблем та оперативне їх вирішення стає досяжним завдяки використанню спеціалізованої системи обговорень, яка створює надійний канал комунікації між учасниками процесу та сприяє розвитку ефективного взаємодії.

Удосконалення якості та забезпечення безпеки в авіаційній галузі є метою зручного обміну інформацією через систему коментарів та обговорень. Цей інструмент виступає як механізм для об'єктивного обговорення ідей та розгляду різних альтернативних рішень, сприяючи тим самим підвищенню якості виробництва та забезпеченню безпеки в період експлуатації.

Впровадження інструментів колективної роботи відкриває нові перспективи для створення сприятливого середовища для інновацій та творчості. Це дозволить об'єднати зусилля різних фахівців для досягнення спільних цілей, сприяючи тим самим розробці новаторських рішень у конструкторських та експлуатаційних процесах. Останній імпульс, що вони вносять у сектор, є важливим кроком у напрямку оптимізації робочих процесів та підвищення загальної продуктивності у галузі авіації.

3.3. Визначення функціональних вимог до десктопної системи

У даному розділі проведено детальний аналіз ключових функціональних вимог до розробленої десктопної системи, спрямованої на збереження та обробку конструкторської документації, а також на сприяння коментуванню та колективному редагуванню. Запропоновані можливості цього додатку визначені з урахуванням конкретних проблем, що існують в авіаційній галузі, та спрямовані на активне поліпшення робочого процесу інженерів.

Однією з основних функцій системи є створення зручного та структурованого сховища для зберігання конструкторської документації. Розроблена система має забезпечувати логічну категоризацію та підкатегоризацію документів, спрощуючи процес організації та навігації великим обсягом інформації.

Для полегшення пошуку та доступу до конкретних документів в системі використовуються метадані та теги, що дозволяють точно ідентифікувати та класифікувати документи. Введені засоби навігації та потужна система пошуку сприяють швидкому та ефективному локалізуванню необхідної інформації серед обширного обсягу документації.

Одним із важливих аспектів є розробка інтуїтивно зрозумілих інтерфейсів для користувачів, спрямованих на швидку орієнтацію в структурі системи зберігання. Також впроваджено можливість створення персоналізованих списків, закладок та шляхів, що сприяє прискоренню навігації між часто використовуваними документами.

Алгоритми ефективного пошуку, розроблені в рамках цього додатку, забезпечують швидке та точне визначення необхідних документів. Використання технологій штучного інтелекту та машинного навчання дозволяє автоматизувати процес пошуку та підвищити релевантність результатів.

Впроваджена система автоматичного оновлення забезпечує актуальність документації, а регулярне архівування старих версій документів сприяє збереженню чистоти та порядку в системі.

У результаті використання цих функціональностей, система зберігання конструкторської документації відображається не лише як обліковий центр, але і як інструмент, що істотно полегшує роботу персоналу, скорочує час пошуку та сприяє оптимальному використанню інформації.

В додаток будуть внесені наступні ключові можливості, спрямовані на вирішення вищезазначених проблем в авіаційній галузі щодо конструкторської документації:

1. Система коментарів та відгуків: Забезпечення можливості користувачам залишати коментарі, лайки та дізлайки під кожним документом. Створення простого та доступного інструменту для обговорення та обміну думками щодо різних аспектів документації. Надання можливості вираження власної думки через лайки та дизлайки для оцінки правильності вказаної інформації.

2. Спільне редагування документів: Реалізація можливості колективного редагування, що дозволяє інженерам спільно працювати над проектами та вносити зміни в конструкторську документацію. Забезпечення ефективного механізму синхронізації та контролю версій для уникнення конфліктів при одночасному редагуванні.

3. В рамках інтеграції з штучним інтелектом (*AI*) передбачено впровадження системи, яка автоматично оброблятиме користувацькі запити. Це означає використання передових алгоритмів та технологій штучного інтелекту для ефективного розпізнавання, інтерпретації та відповіді на запитання чи команди від користувачів. Інтеграція з AI в обробку користувацьких запитів дозволить створити інтелектуальну та відзивчиву систему, яка не лише забезпечить швидкі та точні

відповіді на запитання, але й може підлаштуватися під унікальні потреби кожного користувача.

Систематичний аналіз коментарів та використання рекомендацій для оптимізації взаємодії з документацією та забезпечення швидкого доступу до необхідних даних.

Ці можливості впроваджують інноваційний підхід до взаємодії з конструкторською документацією в авіаційній галузі, надаючи користувачам зручні та ефективні інструменти для спільної роботи, обговорень та автоматизованого аналізу. Це сприятиме покращенню комунікації, точності і актуальності інформації, а також оптимізації робочих процесів інженерів в авіаційній сфері.

Тому, оскільки у авіаційній галузі постійно зростає потреба у зручних та продуктивних інструментах для обробки конструкторської документації, майбутній додаток розроблено відповідно до цієї наростаючої необхідності. Важливою особливістю є збереження коментарів та можливість їхньої інтерактивності через використання лайків та дизлайків, що значно поліпшує ефективність роботи інженерів та сприяє підвищенню якості конструкторської документації.

Інтеграція з штучним інтелектом робить можливим застосування передових технологій для аналізу та удосконалення робочих процесів. Застосування *AI* дозволяє автоматизувати аналіз великого обсягу інформації, швидко виявляти та виправляти можливі недоліки та покращувати загальний стан документації. Таке рішення демонструє інноваційний підхід, впроваджуючи сучасні технології для оптимізації роботи в інженерній сфері. Даний додаток визначається своєю актуальністю та високою відповідністю потребам сучасної авіаційної галузі, пропонуючи інтегроване та інтелектуальне середовище для роботи з конструкторською документацією.

Висновок до розділу 3

Результати аналізу потреб авіаційної галузі дозволили нам глибше зрозуміти ключові виклики та завдання, з якими стикається цей сектор. Основні проблеми

ідентифіковані у вигляді об'єму та складності документації, необхідності комунікації та колективної роботи.

Перший виявлений аспект — величезний обсяг технічної документації, що вимагає вдосконалення системи зберігання. Пропоновані рішення включають розробку структурованої системи зберігання, розподіл документації за категоріями та використання метаданих для полегшення навігації. Важливим є також впровадження потужної системи пошуку та інтуїтивно зрозумілих інтерфейсів.

Другий аспект — необхідність комунікації та колективної роботи. Запропоновані рішення включають введення системи коментарів та обговорень, що дозволить інженерам взаємодіяти під час перегляду документації. Також важливим є можливість спільного редагування документів та формування спільних проектів.

Обґрунтованість і необхідність впровадження інтеграції з штучним інтелектом (*AI*) визначено для оптимізації обробки користувацьких запитів, аналізу коментарів та виправлення помилок. Введення *AI* в документаційний процес може значно підвищити ефективність та точність взаємодії з інформацією.

Узагальнюючи, надані рішення висвітлюють комплексний підхід до вирішення проблем авіаційної галузі, забезпечуючи не лише підвищення продуктивності та ефективності, але й сприяючи покращенню якості робочих процесів та безпеки в даному секторі. Запропоновані рішення не лише відповідають потребам авіаційної галузі, але також відкривають шлях для інновацій та подальшого розвитку в цьому стратегічно важливому секторі.

РОЗДІЛ 4
РОЗРОБКА ДЕСКТОПНОЇ СИСТЕМИ

4.1. Вибір платформи та технологій розробки

Electron веб-фреймворк, який дозволяє створювати крос-платформні додатки з використанням *JavaScript*, *HTML* та *CSS*. Це робиться за допомогою багатого набору *JS API*, які керують особливостями взаємодії з різними операційними системами.

Додатки, створені за допомогою *Electron*, працюють як веб-додатки, з єдиною відмінністю - вони можуть інтерпретувати та зберігати інформацію в файлі системи обробки даних. Electron став частиною інструментарію розробників завдяки своїй здатності прискорювати швидкість розробки.

Electron заповнив пропасть між потребою в сучасних десктопних додатках та доступним інструментарієм для їх створення. Ця веб-технологія підняла розробки програмного забезпечення для десктопу на новий рівень.

Electron використовує логіку домену, дизайн та загальну архітектуру веб-додатків. Результатом є зменшення вартості розробки крос-платформних додатків. *Electron* працює на основі двигуна *Chromium*, що дозволяє реалізовувати найфантастичніші можливості *Chrome*. Двигун може перезавантажувати себе, що означає, що вам не потрібно перезапускати програмне забезпечення кожного разу, коли ви змінюєте код. Це призводить до миттєвого перезавантаження та оптимізації загального враження. Також всі помилки та втрати пам'яті виявляються самими вбудованими інструментами *Chrome*. Таким чином, вам не потрібно залучати сторонній відлагоджувач.

Розглянемо чому додатки на базі *Electron* є гідною альтернативою інших технологій для створення десктопних застосунків.

Якщо звичайну програму потрібно перетворити на крос-платформну за допомогою Electron, вся інформація зберігається в системі на локальному рівні. Таким чином, безпека даних гарантована. З іншого боку, якщо користувач хоче

зберігати інформацію в хмарі, програміст повинен переконатися, що ця хмара має високий рівень безпеки.

JavaScript/Plugin пропонує розробникам програмного забезпечення безкоштовне керування та дає можливість використовувати *API* апаратного рівня. Фреймворк забезпечує гнучкість щодо функцій, дозволяючи розробникам вибирати необхідні функції для своїх настільних проектів. Крім того, перехід на обговорювану технологію відбувається плавно.

У сучасному інформаційному вимірі, де технологічний прогрес розгортається з неабиякою енергією, розгляд програмної трансформації від звичайного до крос-платформенного застосування, з використанням *Electron*, викликає особливий інтерес. Цей перехід дозволяє об'єднати два світи — локального зберігання та хмарної інфраструктури, що в поєднанні охоплюють безпеку даних.

Високу ефективність платформи *Electron* можна пояснити декількома важливими аспектами, які сприяють успіху цього інструменту для розробки десктопних додатків.

По-перше, слід відзначити, що *Electron* використовує високопродуктивний движок *Chromium*, який є основою популярного браузера *Google Chrome*. Це рішення відкриває доступ до сучасних технологій та функціоналу, що забезпечує швидкий та ефективний рендеринг веб-сторінок. Використання такого потужного движка не лише підвищує продуктивність додатків, але і робить їх більш відгукливимий.

По-друге, *Electron* вражає своєю унікальною можливістю перезавантаження без повного перезапуску програми. Це особливо важливо для розробників, оскільки зміни в коді відображаються миттєво, що економить час і прискорює процес розробки. Такий підхід дозволяє швидко вносити зміни та експериментувати з функціоналом, не затримуючи весь цикл розробки.

Крім того, використання мови програмування JavaScript та потужного фреймворка *Node.js* в *Electron* надає розробникам доступ до швидких і продуктивних інструментів. Це стимулює ефективність розробки та сприяє оптимізації роботи додатків, забезпечуючи їх високу швидкість та надійність.

Не менш важливою є крос-платформеність *Electron*, що дозволяє додаткам працювати на різних операційних системах, включаючи *Windows*, *macOS* та *Linux*. Це розширює географію користувачів та робить додаток доступним для широкого кола аудиторії.

Завершуючи, широкі можливості *API Electron* грають ключову роль у взаємодії з операційною системою та різними аспектами додатка. Це робить інтеграцію різноманітних функцій легкішою та сприяє оптимізації робочого процесу додатка.

Оскільки *Electron* підтримується єдиною кодовою базою, розробники програмного забезпечення можуть використовувати цю перевагу для створення як веб-програм, так і програм для настільних ПК. Крім того, базовий код можна повторно використовувати сам по собі, оскільки розробники використовують єдиний код для однієї програми, але поширюють його на всі платформи.

Для забезпечення клієнт-серверної взаємодії для побудови сервісних запитів вирішено використовувати *Express.js*.

Express – це фреймворк для *Node.js*, який використовується для швидкої та ефективної розробки веб-застосунків та *API*. Він є легким, мінімалістичним фреймворком, який надає розробникам інструменти для створення різноманітних застосунків та обслуговування *HTTP*-запитів. Він дозволяє швидко налаштовувати маршрути, обробляти *HTTP*-запити та відправляти відповіді.

Express дозволяє легко визначати маршрути, обробляти запити та відправляти відповіді. Це сприяє швидкій розробці веб-застосунків. Він ідеально підходить для створення *API*, оскільки він надає розширюваний механізм обробки маршрутів та підтримку різних типів відповідей (*JSON*, *XML*, тощо). *Express* легко визначати маршрути та їх обробники, що робить навігацію та обслуговування *HTTP*-запитів ефективним. Просто обслуговує статичні файли, такі як зображення або *CSS*.

Express має лаконічний синтаксис та мінімалістичну структуру, що робить його легким для вивчення та використання. В ньому присутня проста та потужна система маршрутизації дозволяє ефективно обробляти *HTTP*-запити, а концепція *middleware* дозволяє виконувати дії перед обробкою запиту або після неї, розширюючи функціональність додатку. Вибір *middleware* та модульної структури дозволяє легко

додавати функціональність. Завдяки активній спільноті та популярності, *Express* став стандартом в області *Node.js* розробки. Він надає підтримку різних типів даних, різних баз даних та можливість інтеграції з іншими інструментами.

Використання *Express* полегшує розробку застосунків та *API*, надаючи потужні інструменти для створення сучасних та ефективних рішень. Тому він ідеально підходить для розробки сервісної частини застосунку. Модуль що виконується всередині основного процесу знаходиться в Додатку 1.

Ви можете запустити процес електронного рендерингу звідси та спілкуватися з іншими процесами через IPC. Під час запуску `npm run build` або `npm run build:main` цей файл компілюється у `./src/main.js` за допомогою webpack. Це дає нам певні переваги в продуктивності.

Для збереження даних додатку обрано базу даних *MongoDB*. *MongoDB* – це нереляційна, документ-орієнтована база даних, яка зберігає дані у форматі *BSON* (*Binary JSON*). Розроблена компанією *MongoDB* Inc., ця база даних є однією з найпопулярніших в сфері *NoSQL*-рішень.

MongoDB, як сучасна система управління базами даних (СУБД), впроваджує документ-орієнтовану структуру, в якій дані зберігаються у вигляді документів, що представляють собою *JSON*-подібні об'єкти. Ця концепція дозволяє кожному документу мати свій унікальний набір полів та значень, надаючи гнучкість у представленні різноманітних типів даних.

Важливо відзначити гнучкість схеми *MongoDB*, яка не вимагає фіксованої структури даних для документів у колекціях. Це дозволяє розробникам без зайвих обмежень вносити зміни до структури даних, спрощуючи процес розширення та модифікації існуючої інформації.

Система індексації в *MongoDB* підтримується для ефективного доступу до даних та оптимізації запитів, забезпечуючи швидкий пошук та фільтрацію інформації у великих обсягах даних.

MongoDB володіє масштабованістю горизонтального типу, що означає можливість легкого розширення бази даних додаванням нових серверів. Це

забезпечує ефективне розподілення навантаження та забезпечує сталу продуктивність при збільшенні обсягу даних.

Підтримка операцій *ACID* (A*tomicity, Consistency, Isolation, Durability*) в *MongoDB* гарантує надійність даних, забезпечуючи їхню цілісність та стабільність в умовах різних операцій та транзакцій.

Гнучкість у роботі з мовами програмування визначається наявністю офіційних драйверів *MongoDB* для різних мов, таких як *Python, JavaScript, Java*. Завдяки цьому, розробники можуть використовувати мову, яка найкраще відповідає їхнім потребам, забезпечуючи при цьому зручність та ефективність взаємодії із базою даних.

MongoDB використовується в різних областях, таких як веб-розробка, аналітика даних, інтернет речей (*IoT*) та інші, завдяки своїй гнучкості, швидкості та здатності ефективно робити збереження та опрацювання великої кількості неструктурованих даних.

Щоб реалізувати зручну систему колективного редагування та залишення коментарів доцільно використовувати *WebSocket*.

WebSocket – це протокол зв'язку, який забезпечує двосторонню комунікацію між клієнтом та сервером через одне постійне з'єднання. У порівнянні з традиційним HTTP, який використовує запити та відповіді, *WebSocket* дозволяє взаємодіяти з сервером в режимі реального часу.

WebSocket створює постійне двостороннє з'єднання між клієнтом та сервером, що дозволяє обмінюватися даними в режимі реального часу без потреби постійно ініціювати нові запити.

У порівнянні з *HTTP, WebSocket* має менше накладних витрат на з'єднання та обробку даних, що робить його ефективним для реалізації реального часу. Клієнт та сервер можуть взаємодіяти, надсилати та отримувати дані одночасно, що важливо для сценаріїв реального часу, таких як чати, спільне редагування тощо. *WebSocket* дозволяє взаємодіяти з сервером, навіть якщо він розташований на іншому домені, що полегшує реалізацію крос-доменної комунікації.

Для збірки (бандлінгу) і оптимізації ресурсів в створюваній системі використовуємо *Webpack*. Використання *Webpack* при створенні додатку на Electron

має ключове значення через ряд причин, пов'язаних із зручністю розробки, ефективністю та оптимізацією.

Webpack та *Electron* утворюють потужний тандем для розробки модульних та високоефективних додатків. Їхні переваги охоплюють кілька ключових аспектів:

Webpack володіє вражаючою здатністю керувати модулями та збіркою ресурсів. Використання модульної структури *React* та *Electron* робить організацію коду зручною та легкою. *Webpack* автоматично пакує та оптимізує різні ресурси, включаючи *CSS, JavaScript*, зображення та інші, що спрощує розробку.

Webpack пропонує інструмент гарячої заміни модулів (*HMR*), що забезпечує швидкий та інтерактивний процес розробки. Здатність вносити зміни у коді і перезавантажувати тільки змінені частини спрощує відладку та поліпшує продуктивність розробників.

Webpack дозволяє здійснювати мінімізацію та оптимізацію коду перед розгортанням, що призводить до створення компактних та ефективних версій додатків для виробничого середовища.

Webpack славиться своєю гнучкістю у налаштуванні, завдяки конфігураційним файлам. Це дозволяє налаштовувати його для великих та складних проектів, регулюючи параметри для різних режимів роботи (розробка, виробництво).

Webpack ідеально взаємодіє з іншими інструментами розробки, такими як *Babel* та *ESLint*, що дозволяє використовувати сучасний *JavaScript* та виявляти помилки та недоліки в коді.

Оскільки *Electron* часто використовується для створення клієнтських додатків, *Webpack* стає важливим інструментом для організації коду, який виконується на стороні клієнта. Це забезпечує зручність та ефективність в розробці клієнтської частини додатка.

Загалом, використання *Webpack* при розробці додатків на *Electron* з *React* дозволяє створювати продуктивні та оптимізовані за розміром додатки, які легко підтримувати та розвивати.

Об'єднання *Electron, Ract, Express, MongoDB, WebSocket* та *Webpack* в єдиному стеку технологій відкриває широкі можливості для розробки десктопних додатків, які поєднують в собі реактивний інтерфейс та ефективне управління даними.

На першому етапі *Electron* використовується для створення клієнтської частини додатка, яка операційно працює в локальному середовищі користувача. Його можливості дозволяють створювати десктопні додатки з усією функціональністю, доступною на пристрої користувача. Серверна частина, відповідальна за обробку *HTTP*-запитів та взаємодію з базою даних *MongoDB*, реалізована за допомогою *Express*. Це створює зручний місток для об'єднання клієнтської та серверної логіки.

React, у свою чергу, відповідає за реактивний інтерфейс, де компоненти можуть взаємодіяти та реагувати на зміни стану. *WebSocket* забезпечує можливість отримання реального часу для оновлення інтерфейсу при зміні даних на сервері.

MongoDB виступає в ролі бази даних для збереження документів, які використовуються як джерело даних для додатка. *Express* надає *API* для зручної взаємодії з базою даних та оптимальної обробки запитів.

Webpack бере на себе відповідальність за ефективне управління модулями та пакування ресурсів, таких як *CSS, JavaScript*, зображення. Це зменшує обсяг додатка та поліпшує його продуктивність.

Загалом, цей стек технологій створює сучасну інфраструктуру для розробки десктопних додатків з реактивним інтерфейсом, ефективним управлінням даними та високою продуктивністю.

4.2. Проектування інтерфейсу користувача

Проектування інтерфейсу користувача (*UI*) для додатків, які розробляються з використанням *Electron*, є ключовим етапом роботи, який включає кілька важливих аспектів. Оскільки *Electron* використовує веб-технології для створення десктопних додатків, розробка *UI* базується на стандартних веб-технологіях, таких як *HTML, CSS* та *JavaScript*.

Процес проектування інтерфейсу користувача для десктопних додатків через фреймворк Electron визначається основним пріоритетом - забезпечення зручності та високої функціональності взаємодії користувача з програмним продуктом. Дизайн інтерфейсу повинен враховувати технічні аспекти програмування, а також візуально вирішувати завдання з метою створення ефективного та привабливого інтерфейсу.

На першому етапі, розробники повинні визначити логіку взаємодії та функціональні можливості додатка. Це включає в себе аналіз завдань, які додаток повинен вирішувати, та розробку інтерактивних елементів для їх виконання.

Далі важливо врахувати дизайн-принципи, що забезпечать зручність користування та інтуїтивно зрозумілу навігацію. Графічне вирішення інтерфейсу, використовуючи *CSS* та візуальні ефекти, грає важливу роль у створенні привабливого та сучасного обличчя додатка.

Ключовим елементом є також врахування різних розділів користувачів та їхніх потреб. Розробка адаптивного дизайну та підтримка різних роздільних здатностей екранів дозволяє створювати універсальний інтерфейс для різних пристроїв.

В цілому, проектування інтерфейсу користувача для *Electron*-додатків вимагає гармонійного поєднання технічної ефективності та візуальної привабливості з метою забезпечення задоволення та комфорту від використання програмного продукту.

Проектування ІК за допомогою *Electron* включає в себе аналіз потреб цільової аудиторії, розробку ефективної структури та оформлення відповідно до сучасних трендів. Важливим етапом є використання елементів графічного дизайну, анімації та інтерактивності для досягнення найвищого рівня користувацької залученості.

React, як важлива бібліотека для створення інтерфейсів додатків, призначена для полегшення процесу розробки ефективних та динамічних інтерфейсів у веб-додатках, зокрема в односторінкових додатках, де сторінка не перезавантажується під час взаємодії з користувачем.

Однією з ключових переваг *React* є його компонентний підхід, що базується на розбитті інтерфейсу на невеликі та повторно використовувані компоненти. Це забезпечує легку розширюваність та обслуговуваність коду, роблячи розробку ефективнішою.

Використання віртуального *DOM* в *React* сприяє оптимізації оновлення інтерфейсу та підвищує продуктивність. Це дозволяє ефективно взаємодіяти з інтерфейсом та забезпечує високу швидкодію додатка.

JSX (JavaScript XML) в *React* дозволяє писати код, схожий на *XML/HTML*, спрощуючи читання та розуміння структури інтерфейсу. Це робить код більш декларативним та ефективним у виразі.

Модульність та перевикористовування компонентів є ще однією вагомою перевагою *React*. Це забезпечує легку перевикористовуваність компонентів, сприяє структурованості коду та полегшує утримання додатка.

Крім того, *React* дозволяє автоматично оновлювати інтерфейс при зміні стану, роблячи додатки більш реактивними та надаючи користувачеві миттєве відображення змін. Це сприяє вдосконаленню користувацького досвіду та забезпечує плавну та ефективну взаємодію з додатком.

Далі наведені десктопні додатки, які використовують *React* та *Electron*:

- *Visual Studio Code* - це потужний текстовий редактор від *Microsoft*, який використовується для розробки різних типів додатків.
- *Slack* - це комунікаційна платформа для командної роботи, яка надає інструменти для обміну повідомленнями та спільної роботи.
- *WhatsApp Desktop* - це десктопна версія популярного месенджера, що дозволяє користувачам обмінюватися повідомленнями через комп'ютер.
- *Postman* - це інструмент для розробки та тестування *API*, який дозволяє створювати та надсилати *HTTP*-запити.
- *Skype* - це платформа для аудіо- та відеозв'язку, яка надає можливість спілкування за допомогою текстових, голосових та відеовикликів.
- *Discord* - це платформа для групового спілкування, яка часто використовується гравцями та спільнотами.

Ці додатки, побудовані на основі *Electron* та *React*, вирізняються своєю зручністю та функціональністю, а створення їх дизайну значно полегшено завдяки використанню *Styled Components*.

Для стилізації додатку використовується *Styled Components*, бібліотека для *React*, яка дозволяє писати *CSS* в стилі *JavaScript*, який легко інтегрується з *React*-компонентами. Використання *Styled Components* дуже корисне при створенні десктопного додатку на *React* та *Electron*. Перевагами використання *Styled Components* є модульність, динамічність, гнучкість, зменшення кількості колізій.

Однією з вагомих переваг *Styled Components* є їхній модульний підхід. Вони дозволяють писати стилі в межах самого компонента, забезпечуючи велику модульність. Це особливо корисно при роботі зі складними інтерфейсами та зменшенні кількості колізій у глобальному просторі стилів.

Styled Components також вражають своєю динамічністю та гнучкістю. З їхньою допомогою можна використовувати JavaScript для динамічного визначення стилів на основі умов, пропсів чи стану компонента. Крім того, вони дозволяють використовувати всі функції *CSS*, включаючи анімації, псевдокласи, псевдоелементи та інше.

Унікальні ідентифікатори, які використовує *Styled Components*, допомагають уникнути колізій у глобальному просторі стилів, роблячи керування стилями в десктопному додатку читабельним та легким. Такий підхід враховує не лише естетичні аспекти, але і полегшує роботу розробників, що сприяє ефективній і продуктивній розробці.

Styled Components дозволяють писати стилі в межах самого компонента, що робить їх модульними. За допомогою *Styled Components* можна використовувати JavaScript для динамічного визначення стилів на основі умов, пропсів чи стану компонента, а також дозволяє використовувати всі функції *CSS*, включаючи анімації, псевдокласи, псевдоелементи та інше. Завдяки унікальним ідентифікаторам, *Styled Components* допомагають уникнути колізій у глобальному просторі стилів.

Через це використання *Styled Components* у десктопному додатку спрощує роботу зі стилями, роблячи їх читабельними та легко керованими. Використання *Styled Components* та *React* в десктопних застосунках на платформі *Electron* представляє собою потужну комбінацію для створення зручних та функціональних інтерфейсів користувача. *Styled Components*, як бібліотека для стилізації компонентів,

разом із React, який відповідає за структуру та логіку додатку, надають численні переваги в контексті десктопної розробки на *Electron*.

Однією з ключових переваг є модульність та гнучкість *Styled Components*. Їхній підхід до стилізації, де стилі визначаються прямо в межах компонентів, сприяє підтримці чистоти коду та полегшує уникання конфліктів стилів. Це особливо важливо в середовищі, де декілька компонентів можуть спільно використовувати однакові назви класів або ідентифікаторів.

Використання *Styled Components* також сприяє динамічності дизайну, оскільки дозволяє застосовувати стилі в залежності від стану компонента чи його властивостей. Це робить інтерфейс більш адаптивним та реактивним до змін, що може бути важливим для десктопних застосунків, які працюють в різних режимах.

Комбінація *Styled Component*s та *React* в *Electron* дозволяє розробникам зосередитися на легкості розширення та обслуговування коду, що є ключовими аспектами при створенні десктопних додатків. Цей стек технологій допомагає досягти ефективності, модульності та високого рівня користувацької зручності в десктопних застосунках, що базуються на *Electron*.

4.3. Розробка основного функціоналу системи

В цьому розділі детально розглядається процес розробки ключового функціоналу десктопної системи, спрямованої на зберігання та управління конструкторською документацією в авіаційній галузі. Основна мета розробки функціоналу - це створення потужного та ефективного інструменту, який відповідає потребам та вимогам користувачів, активно взаємодіє з конструкторською документацією та полегшує її управління.

Один із ключових аспектів функціоналу - це система зберігання, яка надає надійний та безпечний механізм для збереження конструкторської документації. Враховуючи специфіку авіаційної галузі, детально розробляється структура бази даних, забезпечуючи ефективність та швидкий доступ до необхідної інформації.

Далі, акцент робиться на інтерфейсі системи, орієнтованому на максимальну зручність та інтуїтивність використання. Розробка інтерфейсу включає в себе створення візуальних елементів, які легко сприймаються користувачем, а також можливість швидкого та легкого взаємодії з конструкторською документацією.

Загальною метою є створення системи, яка відповідає високим стандартам авіаційної галузі та надає користувачам інструмент для ефективного та надійного управління конструкторською документацією.

У впровадженому додатку відзначається висока ефективність системи зберігання конструкторської документації, яка не лише надає надійне сховище для важливих даних, але й забезпечує зручний та легкий доступ до цієї інформації.

Система використовує структурований підхід до зберігання документів, ретельно класифікуючи їх за різними параметрами, такими як тип літака, виробник чи дата створення. Це дозволяє користувачам швидко та ефективно знаходити необхідні документи, скорочуючи час на пошук і полегшуючи навігацію в середовищі додатку. Висока рівень уваги до деталей в процесі розробки навігаційної системи сприяє створенню інтуїтивного та легкого у використанні інтерфейсу, що позитивно позначається на загальному досвіді користувача та його продуктивності при роботі з додатком.

При входженні в систему відображається головний екран (рис.4.1), який інтуїтивно пропонує користувачам вибрати компанію, документація якої їх цікавить. Далі, за допомогою послідовних виборів типу літака та його модифікації, користувачі отримують доступ до конкретної документації.

Однією з ключових ідей розробки додатку є структуроване зберігання інформації, що включає в себе не лише основні параметри, але й докладні фільтри, такі як номер деталі, частина літака чи навіть сторона, з якої потрібний компонент розташований. Це робить систему дуже гнучкою та придатною для використання в умовах складних конструкторських завдань.

Зокрема, велика увага приділяється зручності навігації в середовищі додатку, щоб користувачі могли швидко та ефективно отримувати необхідну інформацію. Це

важливо, оскільки розгалужена структура документації вимагає пристосованої системи навігації для оптимального використання всіх можливостей додатку.

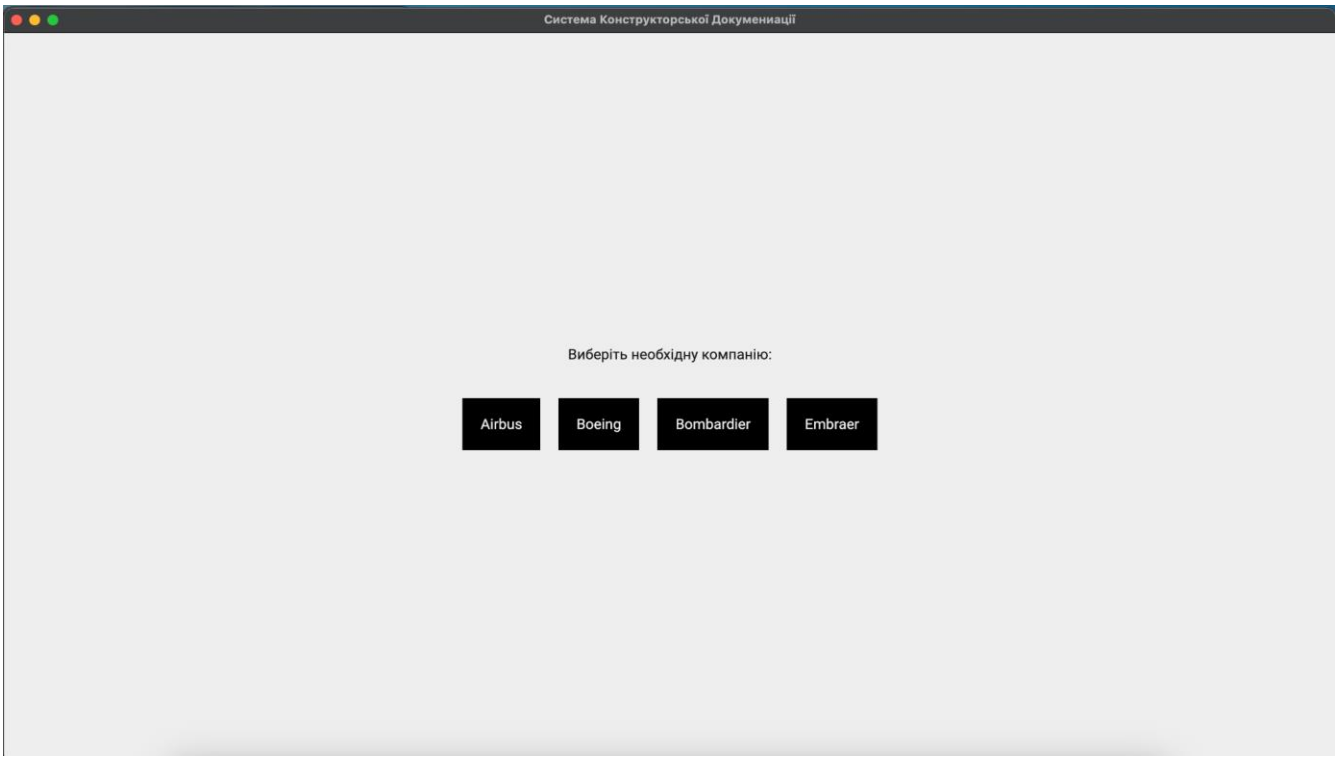


Рис.4.1. Головний екран десктопного додатку

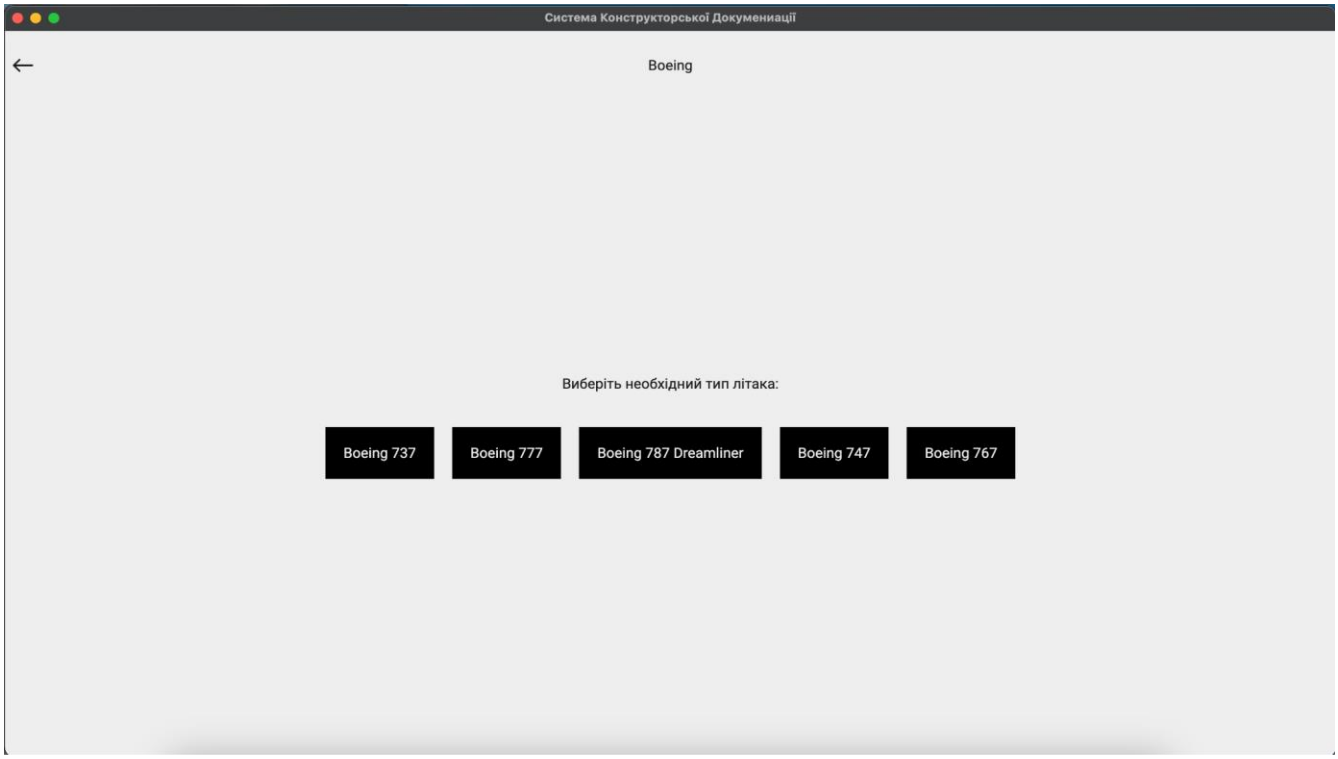


Рис. 4.2. Вибір типу літака

Важливим етапом є вибір типу літака (див рис. 4.2). Враховуючи специфіку додатку, в шапці застосунку розміщено інформацію про попередньо обраний статус та доступ до кнопки навігації, яка дозволяє зручно повертатися на крок назад.

При виборі типу літака користувачеві надається можливість визначити конкретний напрямок пошуку документації. Зручна навігація у вигляді кнопки "Back" спрощує користування додатком, дозволяючи швидко переглядати попередні обрані параметри.

Ця система навігації покликана забезпечити користувачеві зрозуміле та послідовне рухання в середовищі додатку, де кожен вибір інтуїтивно зв'язаний з попереднім, сприяючи легкому та ефективному пошуку конструкторської документації.

Вибір типу літака відкриває можливість обрати модифікацію обраного літака (рис. 4.3). Цей етап надає користувачеві додатковий рівень деталізації та уточнення у виборі конструкторської документації.

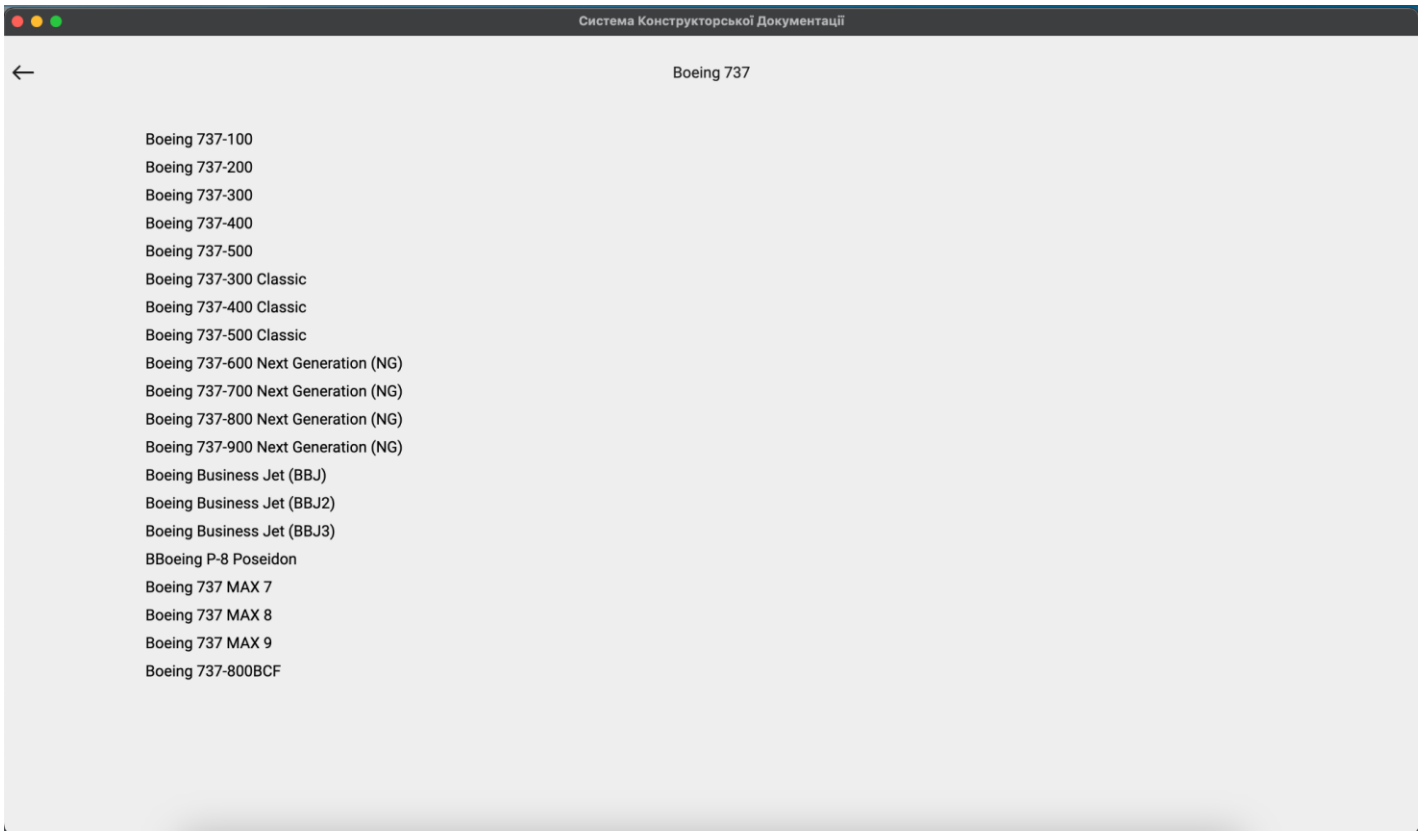


Рис. 4.3. Вибір модифікації літака

За допомогою цього функціоналу система спрощує пошук інформації, дозволяючи користувачеві точно визначити конкретну модифікацію літака, для якої він бажає переглядати або редагувати документацію. Це важливий крок у напрямку забезпечення користувача необхідними інструментами для успішної роботи з системою управління конструкторською документацією в авіаційній галузі.

Обираючи модифікацію літака, система дозволяє користувачеві вибрати конкретний номер або діапазон номерів літака. Такий вибір є важливим, оскільки літаки різних років виготовлення чи партій можуть відрізнятися за типами деталей через зміни в конструкції, виправлення попередніх недоліків або інші фактори.

Цей етап дозволяє точно визначити необхідний діапазон літаків для подальшого отримання відповідної конструкторської документації. Після обрання діапазону номерів літака система пропонує вибрати необхідний тип документації для вибраного діапазону. Цей підхід дозволяє користувачам отримати точно той набір документів, який відповідає їхнім потребам та завданням в роботі з конструкторською документацією (рис. 4.4).

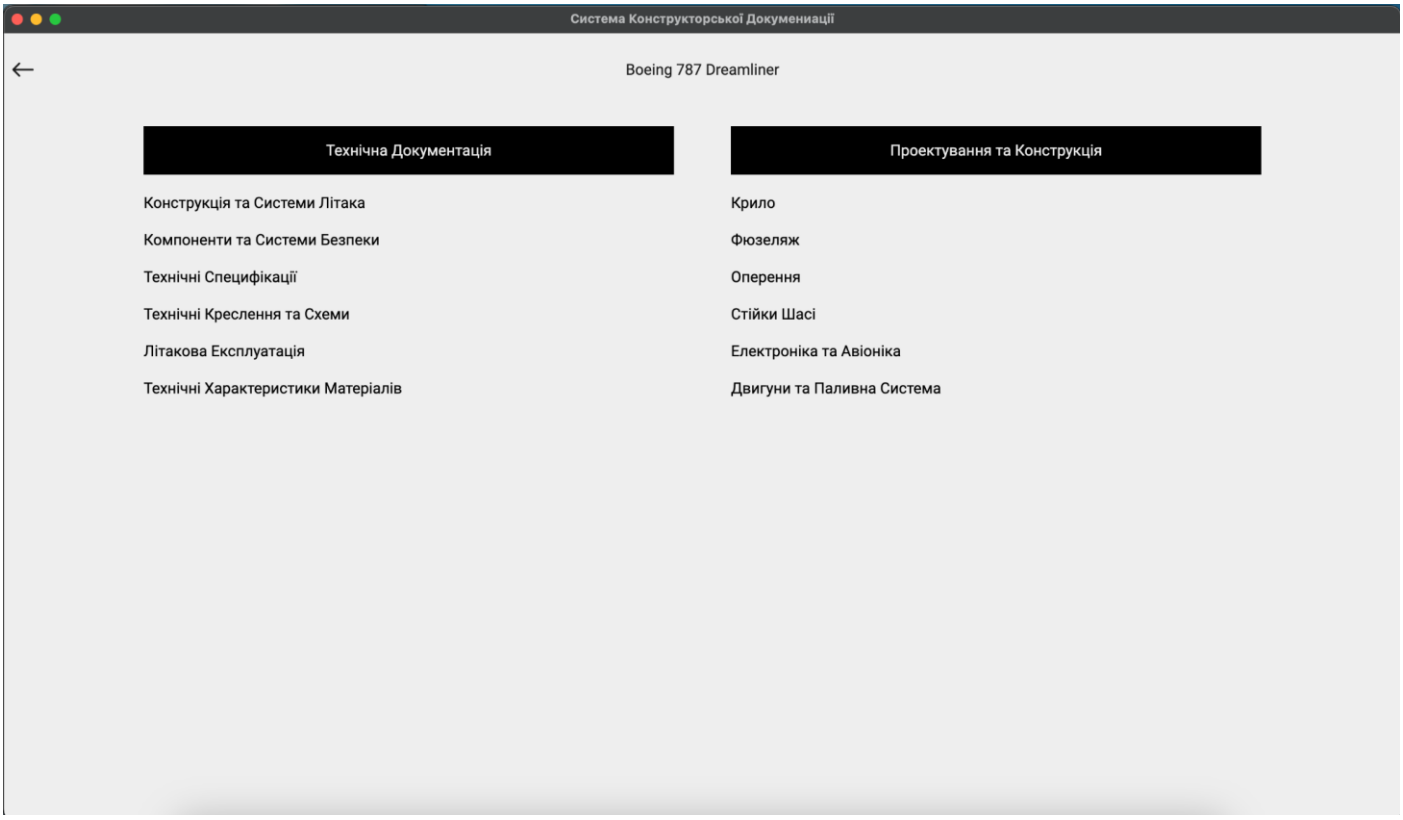


Рис. 4.4. Вибір необхідної документації

Після вибору типу документації користувач переходить до перегляду креслень, які можуть включати креслення конструкцій, елементів, частин елементів та деталей різних типів. Цей етап є важливим для користувача, оскільки він отримує можливість детально розглянути будову літака та окремі його компоненти.

Обираючи конкретне креслення, користувач переходить на сторінку, де впроваджена система коментарів. Це робить процес взаємодії та обговорення деталей більш зручним і ефективним. Користувач може висловлювати свої думки, зауваження чи запитання, а також слідкувати за коментарями інших учасників.

Система коментарів дозволяє створювати відкрите та спільне середовище для обговорення конструкторської документації, обміну думками та швидкої взаємодії між учасниками. Цей функціонал покращує комунікацію в команді, забезпечуючи обгрунтування та обговорення будь-яких аспектів робочої документації для поліпшення результатів роботи.

Система коментарів забезпечує можливості залишати коментарі під кожним документом для обговорення та оцінки правильності вказаної інформації. Користувачі можуть залишати докладні коментарі, обговорюючи різні аспекти документації. Корисувачам надається можливості вираження власної думки, що сприяє оцінці правильності вказаної інформації.

Профілі користувачів та система коментарів в системі передбачають попередню реєстрацію користувача. Для зберігання та управління даними профілів та коментарів використовується база даних *MongoDB*, що забезпечує ефективне зберігання і доступ до інформації.

Серверна частина, яка реалізована на Express.js, відіграє ключову роль у взаємодії з базою даних та наданні *API* для функцій системи, таких як створення користувачів та коментарів. За допомогою *Express.js* створено *API*, яке дозволяє користувачам реєструватися, а також залишати та переглядати коментарі.

В результаті такого підходу в системі існує механізм аутентифікації та авторизації, що дозволяє кожному користувачеві мати власний профіль та здійснювати взаємодію з коментарями від свого імені. Це забезпечує

персоналізований досвід використання системи та дозволяє ефективно керувати обговоренням конструкторської документації.

WebSocket використовується для реалізації миттєвого оновлення коментарів в системі. Процес починається з встановлення постійного *WebSocket* – з'єднання між клієнтом та сервером. Це з'єднання залишається активним протягом тривалості сесії користувача.

WebSocket покращує взаємодію користувача з системою, забезпечуючи швидке та ефективне сповіщення про події. Такий підхід до реалізації коментарів зробив систему більш динамічною та інтерактивною, полегшуючи обговорення конструкторської документації серед користувачів.

Коли користувач додає новий коментар або оновлює існуючий, ця подія обробляється клієнтом. Клієнт використовує *WebSocket*, щоб надіслати коментар чи повідомлення на сервер через встановлене з'єднання. Сервер отримує коментар та використовує *WebSocket*, щоб розповсюджувати оновлення всім підключеним клієнтам.

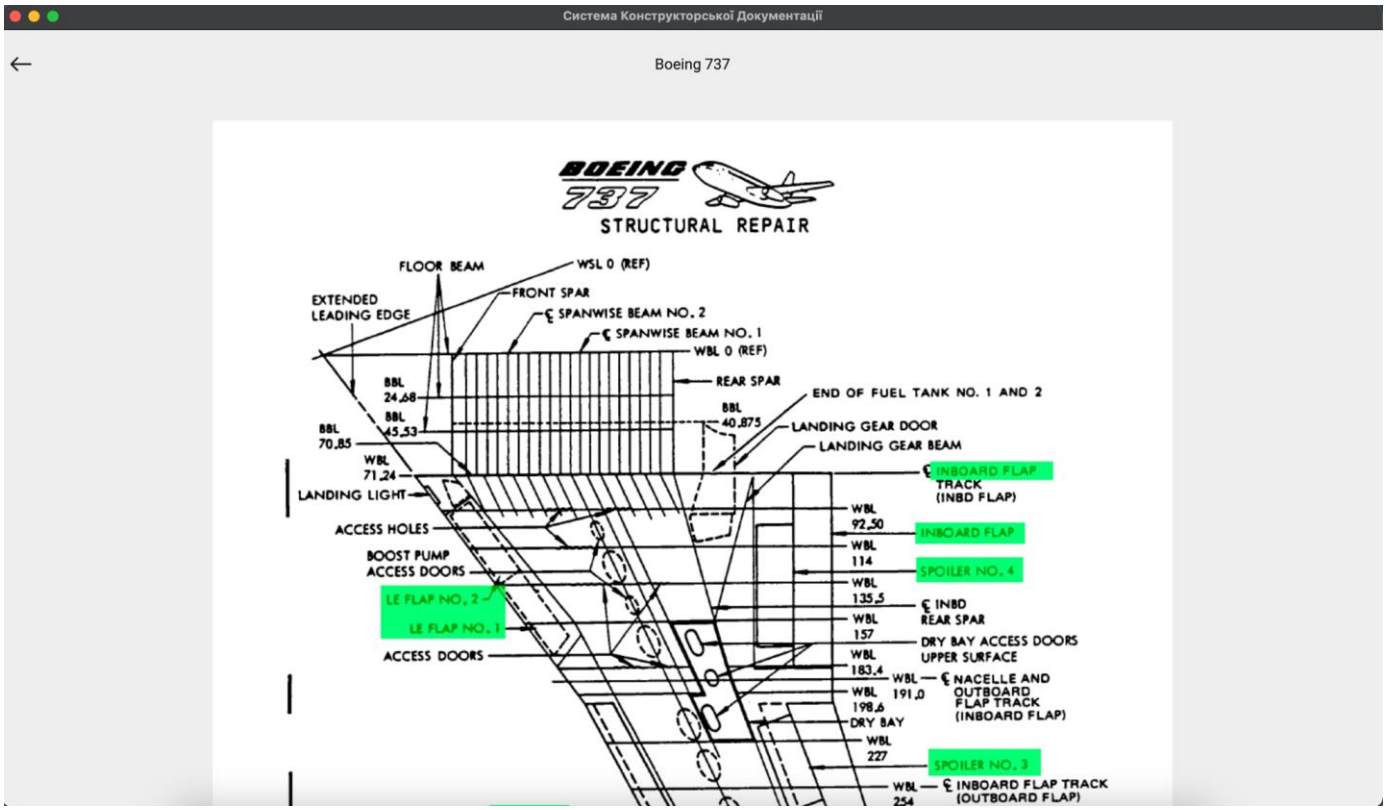


Рис. 4.5. Відображення креслення в додатку

Це дозволяє реалізувати миттєву доставку коментарів в реальному часу і сприяє спільній взаємодії та обміну даними між користувачами та сервером.

Після вибору конкретного документу, додаток відображає креслення чи інше обране зображення. Цей етап є ключовим для користувача, який бажає отримати детальну інформацію (див рис. 4.5).

Далі якщо прогорнути сторінку, знаходиться секція з коментарями користувачів (рис. 4.6).

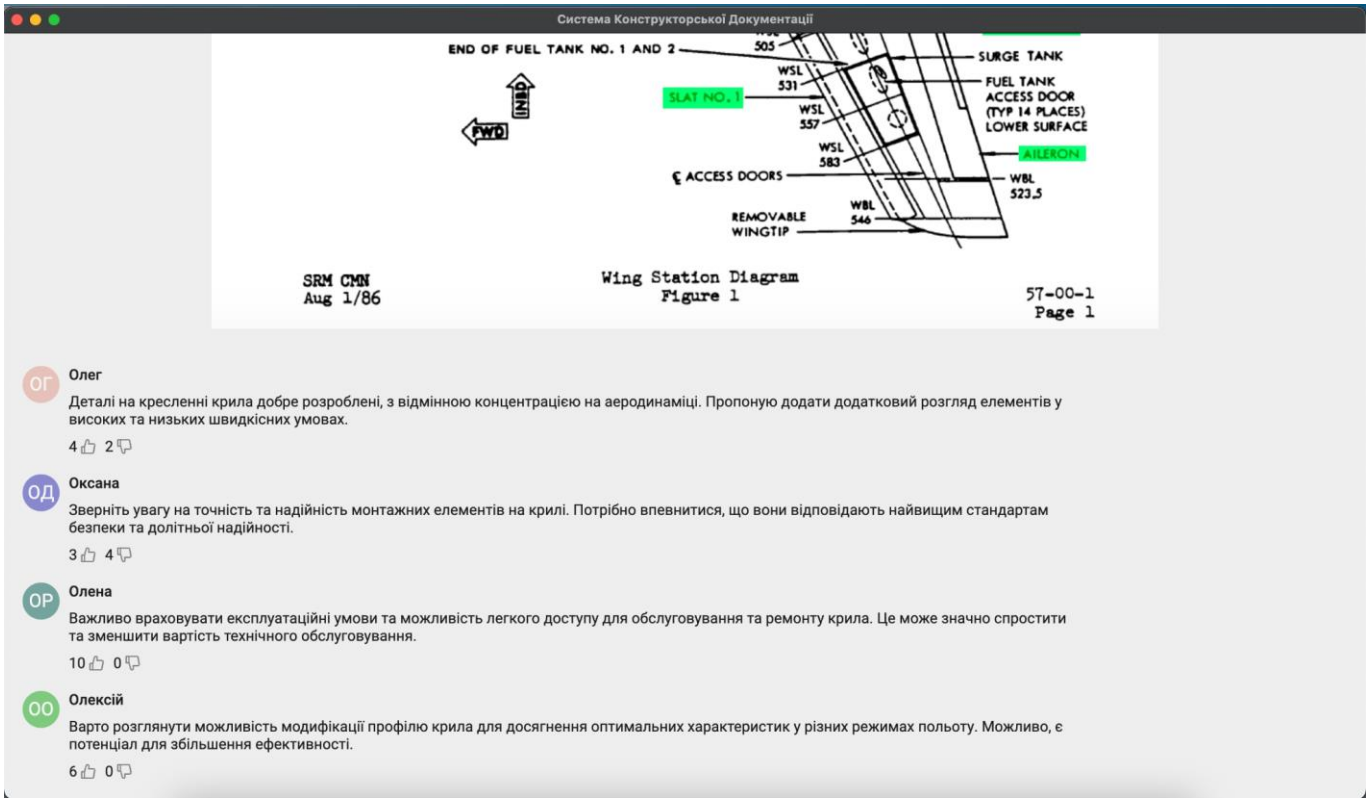


Рис.4.6. Секція з коментарями користувачів

Користувачі мають можливість взаємодіяти з коментарями інших користувачів методом залишення реакцій (рис. 4.7.). Можливість залишення реакцій на коментарі робить взаємодію користувачів більш динамічною та виразною. Основні аспекти цієї функціональності можуть включати. Користувачі можуть використовувати реакції як фільтр для відсортування коментарів за рівнем популярності чи типом відгуку.Відображення кількості реакцій на кожний коментар може слугувати індикатором популярності чи важливості деяких висловлень.

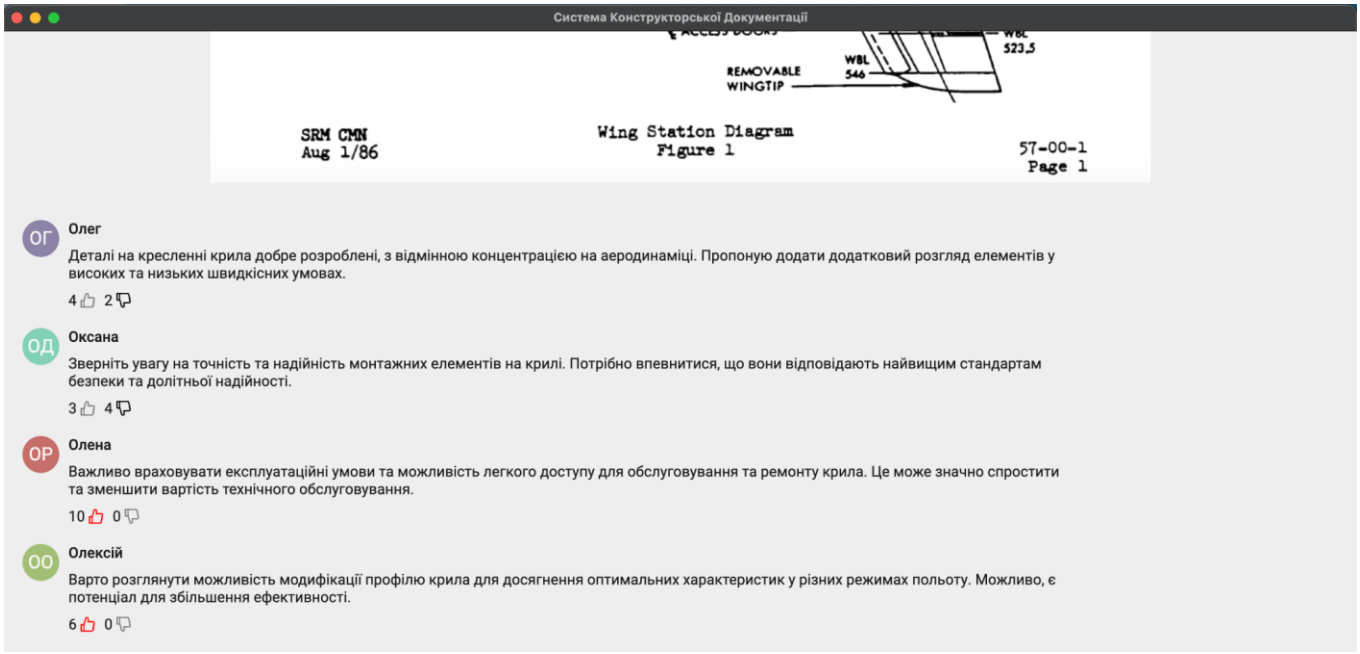


Рис.4.7. Реакції під коментарями

Також дана система надає можливість залишати коментарі під документацією, поле для коментаря зображене на рисунку 4.8.

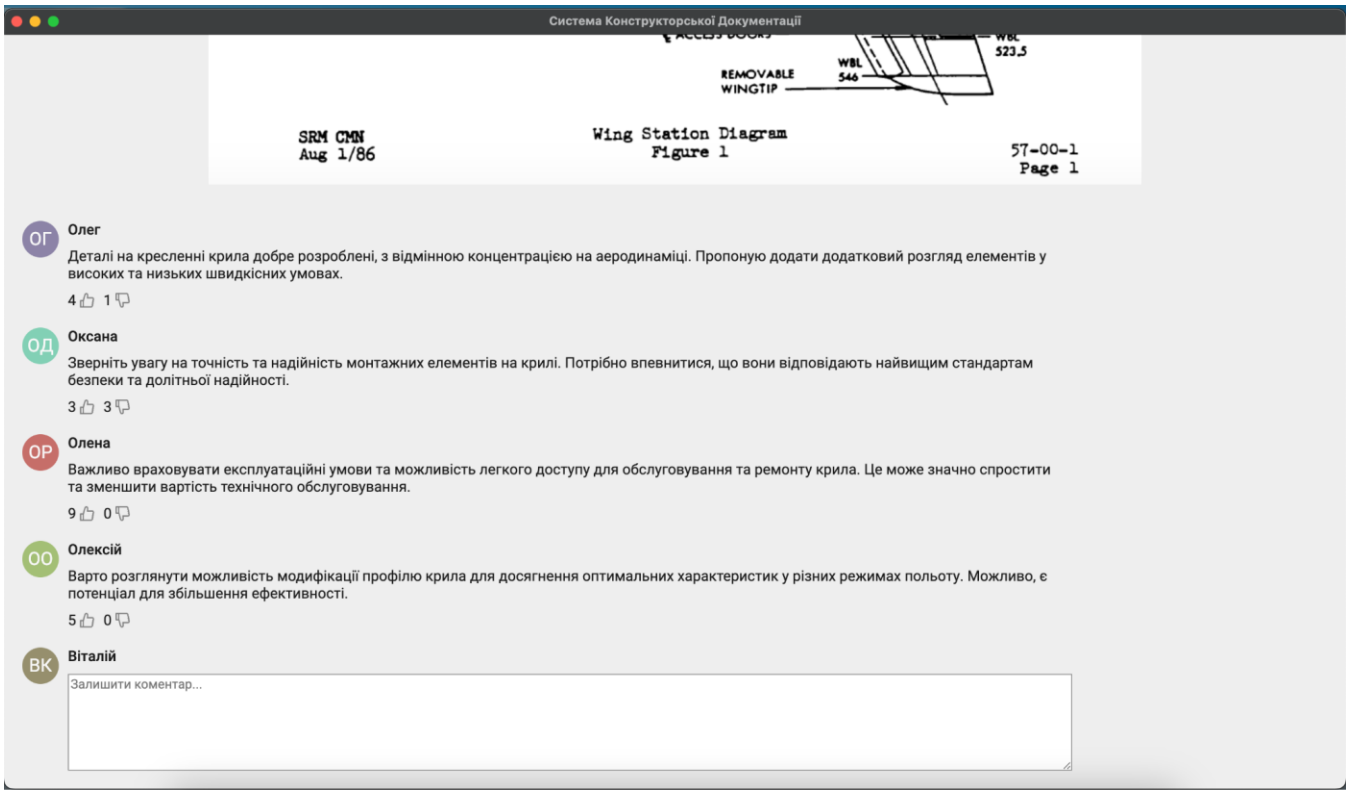


Рис.4.8. Поле для коментарів

У контексті додатку, впровадження штучного інтелекту (ШІ) відкриває широкі можливості для автоматизації та поліпшення взаємодії з користувачем. Найважливіші аспекти використання ШІ включають обробку природної мови (*Natural Language Processing*, *NLP*), машинне навчання та автоматичне виявлення інтенцій користувача.

При використанні ШІ у додатку, система стає здатною розуміти та аналізувати користувацькі запитання в реальному часі, враховуючи їхні інтенції та контекст. Машинне навчання використовується для постійного вдосконалення відповідей на користувацькі запитання, забезпечуючи оптимальну реакцію на змінюючийся контекст.

Система може надавати персоналізовані рекомендації та ефективно взаємодіяти з користувачем за допомогою розширених моделей мовлення, таких як GPT. Реалізація автоматичного виявлення інтенцій дозволяє точно та конкретно реагувати на потреби користувачів.

Важливим аспектом є також врахування персональних даних користувачів та забезпечення високого рівня безпеки та конфіденційності. ШІ може інтегруватися з іншими компонентами системи, такими як база даних чи серверна частина, для отримання більш повного контексту та точних відповідей.

Загальне впровадження ШІ сприяє автоматизації та покращує користувацький досвід, забезпечуючи високу швидкодію та точність відповідей на запитання користувачів в реальному часі.

4.4. Реалізація заходів забезпечення безпеки

У розробці додатку для зберігання та управління конструкторською документацією в авіаційній галузі, я приділив максимальну увагу забезпеченню високого рівня безпеки та конфіденційності користувальницької інформації. Давайте розглянемо кілька ключових аспектів забезпечення безпеки в контексті мого додатку. Всі дані, які зберігаються в базі даних *MongoDB*, шифруються для захисту конфіденційності. Я використовуюте сучасні шифрувальні алгоритми, такі як *AES (Advanced Encryption Standard)*, та засоби шифрування вбудовані в самі бази даних.

Для ц реалізації шифрування ього я застосовую бібліотеки crypto у *Node.js*.

Кожен користувач повинен пройти процес аутентифікації перед доступом до документації. Додаток реалізує різні ролі користувачів, які надають різні рівні доступу, враховуючи принципи авторизації.

Також відбувається забезпечення безпеки шляхом регулярного оновлення додатку та виправлення виявлених вразливостей. Використання сервісів автоматичного впровадження оновлень (*CI/CD*) для швидкого розгортання патчів [21].

В додатку застосовуються токени та протоколи безпеки (*JWT*) для захисту сесій користувачів від перехоплення.

Ці технологічні рішення в сукупності гарантують високий рівень безпеки та захисту користувальницької інформації в моєму додатку.

4.5. Тестування та валідація системи

Тестування – це процес визначення якості чи відповідності продукту визначеним вимогам. У програмуванні та розробці програмного забезпечення тестування використовується для виявлення помилок, перевірки правильності роботи програми та впевненняться, що вона задовольняє визначеним критеріям якості. Розглянемо підходи та стратегії, які застосовуються для забезпечення надійності, ефективності та відповідності функціональності нашої системи для зберігання та управління конструкторською документацією в авіаційній галузі.

1. *Unit Testing*: Кожен модуль та компонент системи піддається одиницьному тестуванню для перевірки його коректності та відповідності визначеним специфікаціям. Використовуючи фреймворки, такі як *Jest* чи *Mocha*, ми можемо автоматизувати процес тестування.

2. Інтеграційне тестування: Після успішного *Unit Testing* модулі об'єднуються для проведення інтеграційних тестів. Мета - перевірити правильність взаємодії між компонентами системи. Застосовуючи фреймворки, такі як *Supertest* для тестування *HTTP*-запитів, ми переконуємося у стабільності взаємодії.

3. Функціональне та регресійне тестування: Проводимо тестування функціональності для перевірки того, чи відповідає система визначеним вимогам. Регресійні тести використовуються для забезпечення того, що нові зміни або додатки не порушать існуючу функціональність.

4. Валідація введених даних: Здійснюється перевірка правильності та валідності даних, які вводяться в систему. Використання регулярних виразів та механізмів валідації допомагає уникнути некоректних даних.

5. Навантажувальне тестування: Визначаємо максимальне навантаження, яке система може витримати, використовуючи інструменти, такі як *Apache JMeter* чи *Locust*. Це дозволяє забезпечити ефективність та стійкість системи при великому обсязі даних чи користувачів.

6. Тестування безпеки: Використовуючи інструменти для тестування на проникнення, ми перевіряємо систему на наявність слабких місць та вразливостей, забезпечуючи високий рівень безпеки.

7. Валідація документації: Перевіряє, чи відповідає документація системи її актуальному стану та функціональним можливостям. Це дозволяє уникнути невідповідностей між описом системи та реальним її станом.

8. Тестування відновлення: Впровадження тестів на відновлення системи після аварійних ситуацій. Використовуючи механізми аварійного відновлення, ми переконуємося в швидкості та ефективності відновлення після можливих збоїв.

9. Автоматизовані тести: Автоматизовані тести використовуються для систематичного виконання тестових сценаріїв під час розробки та регулярних оновлень. Це допомагає швидко виявляти та усувати помилки.

10. Тестування в реальному середовищі: Тестування системи в реальному середовищі, для виявлення певних аспекти при реальному використанні системи користувачами [22].

Ці види тестування можуть використовуватися окремо чи у поєднанні для забезпечення високої якості програмного продукту. Тестування є важливою частиною життєвого циклу розробки програмного забезпечення та сприяє вдосконаленню продукту перед його випуском на ринок.

В процесі розробки десктопного додатку великий акцент приділяється тестуванню та валідації системи, щоб забезпечити високу надійність та якість продукту. Використовуються різні види тестування, які охоплюють різні аспекти функціональності, продуктивності та безпеки.

Підхід *Unit Testing* використовуємо для кожної функція. Для проведення тестів викристовується *Jest*.

Jest – це популярний фреймворк для тестування *JavaScript*, розроблений *Facebook*. Він широко використовується для тестування коду *React*-додатків, а також інших проектів на основі *JavaScript* і *Node.js*. *Jest* підтримує різні види тестів, включаючи Unit Testing, тестування інтеграцій та функціональне тестування.

Взаємодія між різними частинами системи перевіряється через тести *Cypress*.

Cypress – це фреймворк для автоматизованого тестування веб-додатків. Він призначений для створення, запуску та відлагодження тестів, що спрощує процес забезпечення якості веб-проектів. Основні особливості *Cypress* включають:

Оцінка продуктивності та швидкодії системи під різними умовами завантаження тестуємо використовуючи інструменти, такі як *LoadRunner*.

LoadRunner - це програмний продукт для тестування продуктивності та витривалості програмного забезпечення. Він розроблений компанією *Micro Focus* і використовується для виконання тестів завантаження на додатки, веб-сайти та інші сервіси для визначення їхньої продуктивності під навантаженням.

Кожен з цих видів тестування виконує свою роль у забезпеченні стабільності та високої якості розробленого додатку. Тестування проводиться на різних етапах розробки, починаючи від ранніх стадій та закінчуючи фінальними тестами перед випуском продукту.

В десктопному додатку проводиться тестування, що забезпечує надійність створеної системи.

Висновок до розділу 4

У ході розробки десктопної системи для зберігання та управління конструкторською документацією в авіаційній галузі було проведено комплексні дослідження та реалізовано різні аспекти створення програмного продукту. Реалізовані технологічні рішення та функціонал системи виправдали своє призначення, забезпечуючи ефективне управління та надійність в роботі. Розглянемо кожний з етапів розробки відокремлено.

1. Вибір платформи та технологій розробки: На цьому етапі були обрані оптимальні платформи та технології для створення десктопної системи. Використання *Electron, React, Express*, та інших технологій у єдиному стеку дозволило створити продукт, який поєднує ефективність використання ресурсів та зручний інтерфейс користувача.

2. Проектування інтерфейсу користувача: Основним завданням цього етапу було створення зручного та функціонального інтерфейсу для користувачів. Використання бібліотеки React сприяло розробці інтерактивних та естетичних компонентів, а використання *Styled Components* дозволило ефективно управляти стилями, забезпечуючи модульність та гнучкість.

3. Розробка основного функціоналу системи: На цьому етапі був впроваджений зручний механізм зберігання конструкторської документації, який дозволяє користувачам швидко та ефективно отримувати необхідну інформацію. Використання *MongoDB* для зберігання даних, *Express* для реалізації серверної частини, та *WebSocket* для надання можливості реального часу сприяли створенню потужного функціоналу.

4. Реалізація заходів забезпечення безпеки: Безпека системи була високопріоритетною задачею, і на цьому етапі були впроваджені відповідні заходи. Використання *WebSocket* для обміну даними та *HTTPS* для шифрування комунікації забезпечує захищеність інформації, а система автентифікації користувачів здійснюється за допомогою бази даних *MongoDB*.

5. Тестування та валідація системи: Для забезпечення надійності та ефективності системи були використані різні засоби тестування, включаючи *Jest* для модульного тестування компонентів *React* та *Cypress* для виконання енд-тестів. Крім

того, використання *LoadRunner* дозволило провести тести на продуктивність та витривалість, щоб гарантувати оптимальну роботу системи під навантаженням.Основними аспекти, які роблять цю систему вдалим інструментом в авіаційній сфері, наведені нижче.

Використання стеку технологій, таких як *Electron, React, Express, MongoDB, WebSocket* та інші, створило збалансований та ефективний фундамент для десктопної системи. Ці технології не лише забезпечують швидку та ефективну роботу, а й гармонійно співіснують, утворюючи потужний інструмент.

Проектування інтерфейсу користувача зосереджено на створенні зручної та легкозасвоюваної системи. Використання *React* дозволило створити інтерактивні та естетичні компоненти, а *Styled Components* забезпечили модульність та легке управління стилями.

Впроваджені заходи забезпечення безпеки, такі як використання *WebSocket* для шифрування комунікації та автентифікація користувачів через базу даних *MongoDB*, гарантують, що система залишається надійною та захищеною від несанкціонованого доступу.

Механізм зберігання та організації конструкторської документації надає користувачам зручну та швидку навігацію. Використання бази даних *MongoDB* для структурованого зберігання документів забезпечує легкий доступ та сортування за різними параметрами.

Впровадження *WebSocket* для системи коментарів дозволило реалізувати миттєву доставку та обмін коментарів між користувачами в реальному часі. Це сприяє спільній взаємодії та обміну даними, підвищуючи ефективність колективної роботи.

Використання різноманітних інструментів тестування, таких як *Jest, Cypress* та *LoadRunner*, гарантує високу якість та відповідність системи. Модульні та енд-тести сприяють виявленню та виправленню можливих проблем на ранніх етапах.

В цілому, розробка та впровадження десктопної системи є успішним процесом, що підкреслює здатність системи ефективно вирішувати завдання з управління

конструкторською документацією в авіаційній галузі, забезпечуючи зручність та безпеку користування.

ВИСНОВКИ

Дипломна робота присвячена розробці та реалізації десктопної системи обслуговування додатків в авіаційній галузі. Проект враховує поточні технології та вимоги галузі, а також використовує сучасні підходи до розробки та забезпечення безпеки.

В першому розділі диплому проведений огляд існуючих систем обслуговування додатків в авіаційній галузі.У рамках даного розділу було проведено докладний аналіз існуючих систем обслуговування додатків у сфері авіаційної індустрії. Були розглянуті різноманітні аспекти та характеристики існуючих систем, спрямованих на полегшення організації та взаємодії з конструкторською документацією, а також на підвищення ефективності технічного обслуговування та безпеки польотів.

Огляд виявив, що ринок насичений різноманітними системами обслуговування, від простих до комплексних. Кожна з них має свої переваги та недоліки, враховуючи власні характеристики, спрямовані на різні потреби авіаційних підприємств.

Важливою відзначено тим, що більшість існуючих систем акцентує на питання безпеки польотів та технічного обслуговування, враховуючи величезне значення цих аспектів у сфері авіації.

Проведений детальний аналіз функціональних можливостей систем, таких як збереження та організація документації, можливості комунікації та обміну інформацією, відображення стану повітряних суден та використання штучного інтелекту для аналізу та виправлення помилок.

Особлива увага приділяється можливостям інтеграції з іншими авіаційними системами, що сприяє створенню єдиної та злагодженої інфраструктури для оптимального управління та обслуговування авіаційного обладнання.

Огляд виокремив актуальність розробки нових систем, зокрема спеціалізованої десктопної системи для обслуговування авіаційних додатків, що відкриває нові перспективи для покращення ефективності та безпеки в авіаційній галузі.

У цілому, огляд існуючих систем обслуговування додатків в авіації є важливим етапом у розумінні та аналізі потреб галузі, а також визначенні перспектив розвитку для оптимізації авіаційних процесів. Виявивши ключові виклики та проблеми. Це дало змогу визначити напрямки для подальшого дослідження та розробки нової системи.

Розділ 2 диплому розглядав різні технології для розробки десктопних додатків, такі як *Java, C++, Python, Electron, ma Swift*. Було проведено ретельний аналіз і порівняння різних технологій для розробки десктопних додатків, таких як *JavaFX, Qt, Tkinter, Electron ma macOS/iOS SDK*. *Java* та *JavaFX* відрізняються високою кросплатформенністю, що дозволяє створювати зручні інтерфейси, але вимагають значної кількості коду. *Qt* славиться своєю потужністю та ефективністю, хоча може бути вважається складним для новачків. *Tkinter* є простим та легким інструментом для десктопних додатків, але менше потужним у порівнянні з іншими технологіями. *Electron* відзначається великою кросплатформенністю, активною спільнотою та зручністю використання веб-технологій, що робить його привабливим для розробників. Swift використовується для розробки додатків для *macOS* та *iOS,* ідеально вписуючись у екосистему *Apple*.

Вибір технологій пов'язаний з особливостями авіаційної галузі та потребами проекту. Вибір технології для розробки десктопного додатку було здійснено на користь *Electron* через його високу кросплатформенність, швидкість розробки та зручність використання веб-технологій. Ця технологія відповідає вимогам проекту та забезпечує необхідну гнучкість у розробці додатків для різних операційних систем.

У розділі 3 було проведено аналіз потреб авіаційної галузі, поточних вимог та проблем авіаційних підприємств, визначивши можливості для вдосконалення системи обслуговування додатків. Зазначено функціональні вимоги до десктопної системи. Результати аналізу вимог авіаційної галузі дозволили глибше розібратися з ключовими викликами та завданнями, що постають перед цим сектором. Зокрема, виявлені проблеми виражені у великому обсязі та складності технічної документації, а також у необхідності поліпшення комунікації та колективної роботи.

Однією з ключових проблем є значний обсяг технічної документації, що вимагає оптимізації системи зберігання. Запропоновані рішення включають створення структурованої системи зберігання, категоризацію документації та використання метаданих для полегшення навігації. Також важливо впровадити потужну систему пошуку та інтуїтивно зрозумілі інтерфейси.

Другою важливою аспект є необхідність поліпшення комунікації та колективної роботи. Серед пропонованих рішень — введення системи коментарів та обговорень, що дозволить інженерам взаємодіяти при перегляді документації. Також важлива можливість спільного редагування документів та формування спільних проектів.

Обґрунтовано важливість впровадження інтеграції з штучним інтелектом (*AI*) для оптимізації обробки користувацьких запитів, аналізу коментарів та виправлення помилок. Використання *AI* в документаційному процесі може значно підвищити ефективність та точність взаємодії з інформацією.

Узагальнюючи, надані рішення відображають комплексний підхід до вирішення проблем авіаційної галузі, сприяючи не лише підвищенню продуктивності та ефективності, але й поліпшенню якості робочих процесів та безпеки в цьому секторі. Запропоновані рішення відповідають потребам авіаційної галузі та відкривають перспективи для інновацій та подальшого розвитку в цьому стратегічно важливому секторі.

У розділі 4 детально описаний вибір платформи та технологій, проектування інтерфейсу, розробку функціоналу, заходи забезпечення безпеки, та тестування системи. Застосування різноманітних технік та інструментів забезпечило стабільність, ефективність та безпеку додатку.

Розробка десктопної системи для зберігання та управління конструкторською документацією в авіаційній галузі є вкрай актуальною та важливою. Комплексні дослідження та ефективні технологічні рішення, реалізовані на різних етапах розробки, говорять про високий рівень професіоналізму та відповідність результатів потребам галузі.

Обрані платформи та технології, такі як *Electron, React, Express, MongoDB* та *WebSocket*, утворили збалансований стек інструментів, що забезпечив високу ефективність та зручний інтерфейс користувача. Ця технологічна гармонія підкреслює спритність системи в оптимізації ресурсів та забезпеченні надійності роботи.

Проектування інтерфейсу користувача на базі *React* та *Styled Components* дозволило створити не лише зручний, але й естетичний інтерфейс, що сприяє легкому сприйняттю користувачем інформації.

Важливим аспектом є також успішна реалізація механізму зберігання та організації конструкторської документації, який дозволяє користувачам швидко та ефективно отримувати необхідну інформацію. Використання бази даних *MongoDB* та *WebSocket* для реального часу сприяє зручній навігації та взаємодії.

Особлива увага приділялася заходам забезпечення безпеки, що включає шифрування комунікації та автентифікацію користувачів через *MongoDB*. Це гарантує, що система залишається відомою своїм користувачам захищеною та надійною.

Застосування різноманітних інструментів тестування, таких як *Jest, Cypress* та *LoadRunner*, підтверджує високу якість та надійність системи. Це важливо для забезпечення оптимальної роботи системи під різними умовами навантаження.

Розробка та успішне впровадження десктопної системи вказує на значущість досягнень у сфері управління конструкторською документацією в авіаційній галузі. Це не просто технічний прогрес, а важливий крок у вирішенні завдань, що стоять перед цим сектором. Високий рівень фаховості та відповідність розробленої системи потребам галузі підкреслюють її релевантність та здатність ефективно впливати на сучасні виклики авіаційного сектора.

Результати роботи виявилися високопрофесійними та адаптованими до потреб авіаційної галузі. Успішна реалізація системи не тільки віддзеркалює технічний прогрес, але й надає можливість значно покращити ефективність та безпеку операцій в авіаційному секторі.

Розроблена система не лише обґрунтовано відповідає викликам галузі, але й визначається як перспективний інструмент, який відкриває нові горизонти для розвитку та інновацій в стратегічно важливому секторі авіаційної індустрії. Такий підхід підсилює роль розробки в контексті досягнення ключових цілей та вирішення актуальних проблем в авіаційній галузі.
Розроблена десктопна система успішно впоралася із завданнями, що стосуються управління конструкторською документацією в авіаційній галузі. Вона надає користувачам ряд ключових функціональностей, спрямованих на оптимізацію робочих процесів та підвищення комунікаційної ефективності:

1. Створення та зберігання документів у структурованій формі: Користувачі можуть легко створювати та зберігати документи у відповідно структурованому форматі, що полегшує їхнє подальше використання.

2. Організація документів за категоріями: Документи систематизуються за категоріями, що дозволяє користувачам легко знаходити необхідну інформацію.

3. Коментування та обговорення документації: Користувачі можуть залишати коментарі та вести обговорення, що сприяє колективній роботі та обміну ідеями.

4. Швидкий та синхронізований обмін змінами: Механізм обміну дозволяє користувачам швидко та ефективно вносити зміни в документацію, забезпечуючи синхронізацію інформації.

5. Інтеграція штучного інтелекту забезпечує моментальні відповіді на користувацькі запити, тепер не потрібно мати кілька джерел інформації, а можна скористатися пошуком відповідей на важливі питання в самому додатку.

Ці функції допомагають підвищити продуктивність та забезпечити зручний та безпечний доступ до конструкторської документації. Розроблений інструмент може стати невід'ємною частиною робочого процесу авіаційних підприємств, сприяючи покращенню ефективності та забезпеченню високого стандарту безпеки в галузі.

Дипломна робота успішно реалізувала поставлені завдання та врахувала вимоги авіаційної галузі. Розроблена десктопна система відповідає сучасним стандартам, а забезпечення безпеки та тестування гарантують надійність та ефективність використання. Проект може стати цінним інструментом для авіаційних підприємств,

що мають потребу у зручному та надійному інструменті для обслуговування своїх додатків.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. *Moisejenko K.* П’ять найкращих технологічних тенденцій в авіаційній промисловості [Електронний ресурс] / *Karina Moisejenko*. – 2023. – Режим доступу до ресурсу: *https://baatraining.com/blog/5-top-technology-trends-in-aviation-industry/*.

2. *Altintas N.* Роль програмного забезпечення в авіаційній галузі [Електронний ресурс] / *Nazif Altintas*. – 2023. – Режим доступу до ресурсу: *https://medium.com/@aviator2012/the-importance-of-software-and-technology-in-the-aviation-industry-8ab84c866c4d*.

3. *DR. OMAR MEMON*. Різні типи інженерів в авіаційній промисловості [Електронний ресурс] / *DR. OMAR MEMON*. – 2023. – Режим доступу до ресурсу: *https://simpleflying.com/aviation-industry-engineer-types-guide/*.

4. Застосування технологій надійності в цивільній авіації: отримані уроки та перспективи [Електронний ресурс] / Enrico ZIO, Mengfei FAN, Zhiguo ZENG, Rui KANG. – 2019. – Режим доступу до ресурсу: https://www.sciencedirect.com/science/article/pii/S1000936118301948.

5. Методи та етапи дослідження авіаційної галузі [Електронний ресурс] / *Dennis Keiser, Lars Henrik Schnoor, Birte Pupkes, ichael Freitag*. – 2023. – Режим доступу до ресурсу: *https://www.sciencedirect.com/science/article/abs/pii/S0969699723000613*.

6. Системи навігації в авіаційній галузі [Електронний ресурс] // havkar – Режим доступу до ресурсу: https://havkar.com/en/blog/view/aircraft-navigation-systems/121.

7. Система моніторингу та діагностики [Електронний ресурс] – Режим доступу до ресурсу: https://aero-sight.com/flight-data-monitoring.

8. Системи управління бортовою авіонікою [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Aircraft_flight_control_system.

9. Огляд існуючих систем обслуговування [Електронний ресурс] // OAG. – 2023. – Режим доступу до ресурсу: https://www.oag.com/blog/system-transition-airline-industry.

10. Ludwig S. Проблеми та виклики в авіації [Електронний ресурс] / Sean Ludwig. – 2023. – Режим доступу до ресурсу: https://uschamber.com/infrastructure/aviation/the-most-important-issues-facing-the-aviation-industry.

11. *Bloch J. Effective Java / Joshua Bloch., 2018. – (ISBN-13: 978-013468599).*

12. *Kathy Sierra. Head First Java / Kathy Sierra, Bert Bates.. – (ISBN-13: 978-0596009205; 2005).*

13. Hendrik Ebbers. Mastering JavaFX 8 Controls / Hendrik Ebbers., 2014. – (Oracle Press). – (ISBN-13: 978-0071833770).

14. *Lee Zhi Eng. Qt5 C++ GUI Programming Cookbook / Lee Zhi Eng., 2016. – (ISBN-13: 978-1783280271).*

15. *Bhaskar Chaudhary. Tkinter GUI Application Development Blueprints / Bhaskar Chaudhary., 2015. – (ISBN-13: 978-1785889738).*

16. Steve Kinney. Electron in Action / Steve Kinney., 2018. – (ISBN-13: 978-1617294143).

17. Clayton Powell. Mastering Electron / Clayton Powell., 2018. – (ISBN-13: 978-1788299328).

18. Paul Hudson. Pro SwiftUI: Develop for iOS with SwiftUI, Combine, and Xcode / Paul Hudson., 2019. – (ISBN-13: 978-1484255159).

19. Apple Inc. App Development with Swift / Apple Inc., 2021. – (ISBN-13: 978-1714363158).

20. Klaus Pohl. Requirements Engineering: Fundamentals, Principles, and Techniques, Springer / Klaus Pohl, Chris Rupp., 2011. – (ISBN-13: 978-3642145737).

21. Ross J. Anderson. Security Engineering: A Guide to Building Dependable Distributed Systems / Ross J. Anderson., 2020. – (ISBN-13: 978-1119642816).

22. *Kanglin Li. Effective Software Test Automation / Kanglin Li., 2020. – (ISBN-13: 978-1484254794).*

ДОДАТОК 1

ЛІСТИНГ ГОЛОВНОГО ФАЙЛУ *ELECTRON*

```
import path from 'path';
import { app, BrowserWindow, shell, ipcMain } from 'electron';
import { autoUpdater } from 'electron-updater';
import log from 'electron-log';
import MenuBuilder from './menu';
import { resolveHtmlPath } from './util';

class AppUpdater {
 constructor() {
   log.transports.file.level = 'info';
   autoUpdater.logger = log;
   autoUpdater.checkForUpdatesAndNotify();
 }
}

let mainWindow: BrowserWindow | null = null;

ipcMain.on('ipc-example', async (event, arg) => {
 const msgTemplate = (pingPong: string) => `IPC test: ${pingPong}`;
 console.log(msgTemplate(arg));
 event.reply('ipc-example', msgTemplate('pong'));
});

if (process.env.NODE_ENV === 'production') {
 const sourceMapSupport = require('source-map-support');
 sourceMapSupport.install();
}

const isDebug =
 process.env.NODE_ENV === 'development' || process.env.DEBUG_PROD === 'true';

if (isDebug) {
 require('electron-debug')();
}

const installExtensions = async () => {
 const installer = require('electron-devtools-installer');
 const forceDownload = !!process.env.UPGRADE_EXTENSIONS;
 const extensions = ['REACT_DEVELOPER_TOOLS'];

 return installer
   .default(
     extensions.map((name) => installer[name]),
```

```
      forceDownload,
    )
    .catch(console.log);
};

const createWindow = async () => {
 if (isDebug) {
   await installExtensions();
 }

 const RESOURCES_PATH = app.isPackaged
   ? path.join(process.resourcesPath, 'assets')
   : path.join(__dirname, '../../assets');

 const getAssetPath = (...paths: string[]): string => {
   return path.join(RESOURCES_PATH, ...paths);
 };

 mainWindow = new BrowserWindow({
   show: false,
   width: 1024,
   height: 728,
   icon: getAssetPath('icon.png'),
   webPreferences: {
     preload: app.isPackaged
       ? path.join(__dirname, 'preload.js')
       : path.join(__dirname, '../../.erb/dll/preload.js'),
   },
 });

 mainWindow.loadURL(resolveHtmlPath('index.html'));

 mainWindow.on('ready-to-show', () => {
   if (!mainWindow) {
     throw new Error('"mainWindow" is not defined');
   }
   if (process.env.START_MINIMIZED) {
     mainWindow.minimize();
   } else {
     mainWindow.show();
   }
 });

 mainWindow.on('closed', () => {
   mainWindow = null;
 });
```

```
 const menuBuilder = new MenuBuilder(mainWindow);
 menuBuilder.buildMenu();

 // Open urls in the user's browser
 mainWindow.webContents.setWindowOpenHandler((edata) => {
   shell.openExternal(edata.url);
   return { action: 'deny' };
 });

 new AppUpdater();
};

app.on('window-all-closed', () => {
 if (process.platform !== 'darwin') {
   app.quit();
 }
});

app
 .whenReady()
 .then(() => {
   createWindow();
   app.on('activate', () => {
     if (mainWindow === null) createWindow();
   });
 })
 .catch(console.log);
```