**ДЕРЖАВНИЙ УНІВЕРСИТЕТ**
**ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ**
**НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ**
**ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**
**КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

# КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка програмного забезпечення для відео-трансляції Міжнародної Космічної Станції (МКС), в момент, коли вона пролітає максимально близько над користувачем з використанням Golang, гаверсинуса, та орбітальних ефемеридних даних NASA»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

*Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело*

_____ Роман БЄЛИЙ
*(підпис)*

Виконав: здобувач вищої освіти групи ППЗ-51

Роман БЄЛИЙ

Керівник: Віталій ЗАЛИВА
*доктор філософії*
*PhD.*

Рецензент: _____

**Київ 2024**

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ**
**ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ**

**Навчально-науковий інститут інформаційних технологій**

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

**ЗАТВЕРДЖУЮ**
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
«_____» _____ 2024 р.

# ЗАВДАННЯ
# НА КВАЛІФІКАЦІЙНУ РОБОТУ

Бєлому Роману Денисовичу

1. Тема кваліфікаційної роботи: «Розробка програмного забезпечення для відео-трансляції Міжнародної Космічної Станції (МКС), в момент, коли вона пролітає максимально близько над користувачем з використанням Golang, гаверсинуса, та орбітальних ефемеридних даних NASA»
керівник кваліфікаційної роботи доктор філософії., старший викладач кафедри ІПЗ Віталій ЗАЛИВА,
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» лютого 2024 р. №145.

2. Строк подання кваліфікаційної роботи «28» травня 2024 р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Аналіз предметної галузі.
2. Проектування програмного застосунку.
3. Розробка програмного застосунку.
4. Тестування програмного застосунку.

5. Перелік графічного матеріалу: *презентація*
   1. Діаграма класів - 1.
   2. Діаграма класів - 2.
   3. Діаграма послідовності.

6. Дата видачі завдання«28» лютого 2024 р.

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів кваліфікаційної роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Підбір та аналіз науково-технічної літератури | 28.02-06.03.2024 | |
| 2 | Аналіз та дослідження існуючих аналогів | 07.03-13.03.2024 | |
| 3 | Вивчення матеріалів для аналізу існуючих рішень самостійної роботи здобувача | 15.03-31.03.2024 | |
| 4 | Аналіз переваг та недоліків існуючих програмних засобів | 01.04-10.04.24 | |
| 5 | Дослідження функціональних та нефункціональних вимог до застосунку | 11.04-04.05.24 | |
| 6 | Проєктування інтерфейсу користувача | 05.05-10.05.24 | |
| 7 | Оформлення роботи: вступ, висновки, реферат | 29.04-05.05.24 | |
| 8 | Розробка демонстраційних матеріалів | 06.05-12.05.24 | |
| 9 | Попередній захист роботи | 13.05-31.05.2024 | |

Здобувач вищої освіти _____ Роман БЄЛИЙ
*(підпис)*

Керівник
кваліфікаційної роботи _____ Віталій ЗАЛИВА
*(підпис)*

# РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 60 стор., 3 табл., 15 рис., 30 джерел.

*Мета роботи* – спрощення процесу відео-трансляції руху Міжнародної Космічної Станції за допомогою програмного забезпечення, розробленого з використанням мови програмування Golang.

*Об'єкт дослідження* – процес відео-трансляції руху Міжнародної Космічної Станції.

*Предмет дослідження* – програмне забезпечення для відео-трансляції руху Міжнародної Космічної Станції.

*Короткий зміст роботи:* В роботі проведено дослідження галузі відео-трансляції МКС, розглянуті основні методи обчислення відстані між двома геопозиціяними точками на сфері. Проаналізовано методи визначення геопозиції МКС у режимі реального часу, досліджено основні формати орбітальних даних та програмні засоби для відстеження та трансляції МКС:Heavens-Above, Sky-View, NASA's Spot the Station.

Розроблено архітектуру десктопного програмного забезпечення та реалізовано ключові функціональні можливості: автоматичне визначення розташування користувача, прив'язка моменту запуску відеотрансляції МКС до геолокації користувача та сповіщення за допомогою email та нативно через ОС. Проведено функціональне тестування програмного забезпечення. Сферою використання програмного забезпечення є спрощення відеотрансляції руху МКС з прив'язкою до геопозиції користувача.

КЛЮЧОВІ СЛОВА: МІЖНАРОДНА КОСМІЧНА, GOLANG, ВІДЕО-ТРАНСЛЯЦІЯ, ГЕОПОЗИЦІЯ, АВТОМАТИЗАЦІЯ, ДЕСКТОП.

## ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

| | | |
|---|---|---|
| БД | - | База Даних |
| AMQP | - | Advanced Message Queuing Protocol |
| REST | - | Representational State Transfer |
| UI | - | Інтерфейс користувача |
| NUI | - | Природний інтерфейс користувача |
| GUI | - | Графічний інтерфейс користувача |
| VDI | - | Мережеве налаштування |
| TUI | - | Відчутний інтерфейс користувача |

# ВСТУП

У сучасному світі технологій і космосу, постійно вдосконалюється та розширюється використання космічних досліджень для повсякденного життя людини. Міжнародна Космічна Станція (МКС), яка є символом міжнародної співпраці та технологічного прогресу, надає унікальні можливості для наукових досліджень і розвитку новітніх технологій. Одним із важливих аспектів використання МКС є можливість відео-трансляції її польоту, що дає змогу людям по всьому світу спостерігати за роботою станції та красою земної орбіти.

Бакалаврська робота зосереджена на розробці програмного забезпечення, яке дозволить користувачам в реальному часі отримувати відео-трансляції проліту МКС максимально близько над ними. Особливістю проекту є використання мови програмування Golang, яка відома своєю високою продуктивністю та ефективністю в обробці конкурентних даних, що є критично важливим для реалізації такого типу програмного забезпечення.

Для точного визначення часу та положення МКС над користувачем, використовується гаверсинус формула — метод, який дозволяє обчислити найкоротші відстані між двома точками на сфері на основі їхніх географічних координат. Також критичним аспектом роботи є інтеграція орбітальних ефемеридних даних від NASA, які забезпечують актуальну інформацію про траєкторію польоту станції.

Розробка такого програмного забезпечення ставить перед дослідником низку викликів, зокрема, забезпечення високої точності визначення місцезнаходження МКС та ефективної обробки відео-даних в режимі реального часу. Однак, успішне вирішення цих завдань не тільки сприятиме науковому співтовариству та широкому загалу в отриманні нових знань про

космічний простір, але й відкриває двері до нових технологічних можливостей у сфері космічних технологій.

Актуальність теми дослідження полягає у необхідності розробки програмного забезпечення, яке легко використовувати для відеотрансляції МКС у режимі реального часу з автоматічною прив’язкою до розташування користувача.

В результаті роботи створено функціональне та зручне десктопне програмне забезпечення, яке дозволить підвищити ефективність спостереження руху МКС в режимі онлайн.

# 1 АНАЛІЗ ГАЛУЗІ ВІДЕОСТРАНСЛЯЦІЇ МКС ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## 1.1 Огляд галузі відеотрансляції

Відео-трансляція з МКС є унікальною областю, що поєднує космічні технології та медійні комунікації. За останні роки інтерес до космосу знову зростає, що сприяло збільшенню попиту на безпосереднє спостереження за космічними місіями. МКС, будучи найбільшою космічною платформою для наукових досліджень та міжнародної співпраці, має різноманітне обладнання для відеозйомки, від простих камер всередині станції до складних зовнішніх систем, що дозволяють вести трансляції неймовірної якості.

## 1.2 Технологічні аспекти

Обладнання для відеозйомки: Включає в себе внутрішні та зовнішні камери, які здатні працювати в екстремальних умовах космічного простору. Особливу увагу варто звернути на стійкість цих камер до космічних випромінювань та екстремальних температур.

Системи передачі даних: Висока залежність відеотрансляцій від стабільності та швидкості передачі даних. Використання супутникових систем зв'язку дозволяє передавати відео в реальному часі, але потребує високого рівня координації та технічного обслуговування.

Програмне забезпечення для обробки відео: Розвиток програмного забезпечення, яке дозволяє оптимізувати і покращувати якість відео в умовах обмеженої пропускної спроможності, є ключовим для забезпечення якісних трансляцій.

## 1.3 Виклики і можливості

Логістичні та технічні виклики: Необхідність впоратися з затримками у передачі даних, обмеженою пропускною спроможністю, а також постійними випробуваннями обладнання в умовах космосу.

Безпека даних: Забезпечення безпеки відеоданих від несанкціонованого доступу та можливих кібератак, враховуючи що дані передаються через відкритий космічний простір.

Нові технологічні розробки: Можливість впровадження нових технологій, таких як штучний інтелект та машинне навчання, для автоматизації процесів обробки відео та покращення якості трансляцій.

## 1.4 Соціальні та культурні аспекти

Едукативна цінність: Віде-отрансляції з МКС надають унікальну можливість для освітніх інституцій використовувати ці матеріали для навчання студентів астрономії, фізики та інженерії.

Залучення громадськості: Трансляції можуть залучити ширшу аудиторію до космічних досліджень, підвищуючи інтерес і підтримку космічних програм.

Розглядаючи всі ці аспекти, можна зрозуміти, як технічні інновації та виклики формують поточний стан та майбутнє відеотрансляцій з МКС, а також які стратегії можуть бути використані для подальшого розвитку цієї важливої галузі.

## 1.5 Аналіз програмного забезпечення для відео-трансляції МКС

### 1.5.1 Heavens-Above

Heavens-Above є популярним веб-сайтом, який надає інформацію про небесні явища, включаючи видимість супутників, Міжнародної Космічної Станції (МКС), планет, комет та інших астрономічних об'єктів. Цей ресурс використовує

ефемеридні дані для розрахунку орбіт і видимості об'єктів в конкретному місці на Землі. Основна мета Heavens-Above — зробити астрономію доступною і зрозумілою для широкої публіки без необхідності використання спеціалізованого обладнання.

Переваги

Точність і надійність: Heavens-Above відомий своїм точним прогнозуванням видимості космічних об'єктів. Використання актуальних орбітальних даних забезпечує користувачам достовірну інформацію.

Доступність та освітній потенціал: веб-сайт має простий та інтуїтивний інтерфейс, що робить астрономію доступною для широкої публіки, включаючи школярів та аматорів астрономії.

Безкоштовний доступ: Heavens-Above надає всю свою функціональність безкоштовно, що є великою перевагою для користувачів, які не хочуть витрачати гроші на додаткові ресурси.

Міжнародне покриття: сайт підтримує різні мови та дозволяє вибрати будь-яке місцезнаходження по всьому світу, що робить його корисним для міжнародної аудиторії. На рис. 1.1 представлено веб-інтерфейс Heavens-Above.

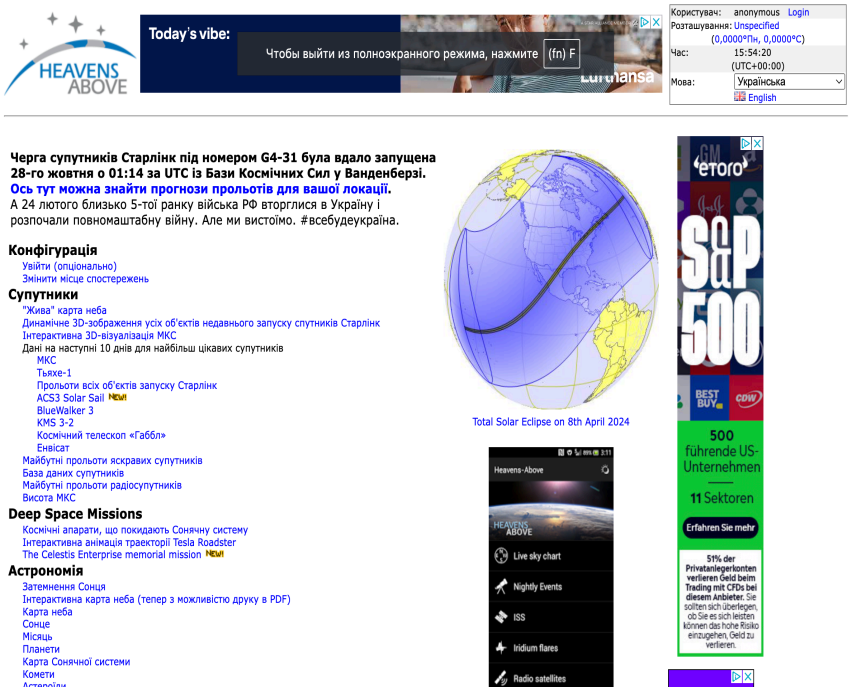



Рис. 1.1 – Веб-інтерфейс Heavens-Above

Недоліки

Обмеженість у функціоналі: на відміну від деяких інших астрономічних платформ, Heavens-Above може бути обмеженим у плані інтерактивності та додаткових освітніх ресурсів.

Відсутність мобільної адаптації: наразі не існує офіційного мобільного додатку, що може ускладнити використання сервісу на мобільних пристроях.

Дизайн та візуальні аспекти: якщо функціональність сайту на високому рівні, дизайн може здатися застарілим порівняно з сучасними веб-додатками, що може відштовхувати нових користувачів.

Технічна підтримка: Heavens-Above може мати обмежені можливості технічної підтримки, що може створювати проблеми при вирішенні складних запитань користувачів.

### 1.5.2 SkyView

SkyView — популярний астрономічний додаток, який дозволяє користувачам використовувати доповнену реальність для ідентифікації зірок, планет, сузір'їв та інших космічних об'єктів у небі. Додаток надає детальну інформацію про кожен об'єкт і покращує освітній досвід, використовуючи візуальні технології для залучення користувачів. На рис. 1.2 представлено Інтерфейс додатку SkyView.

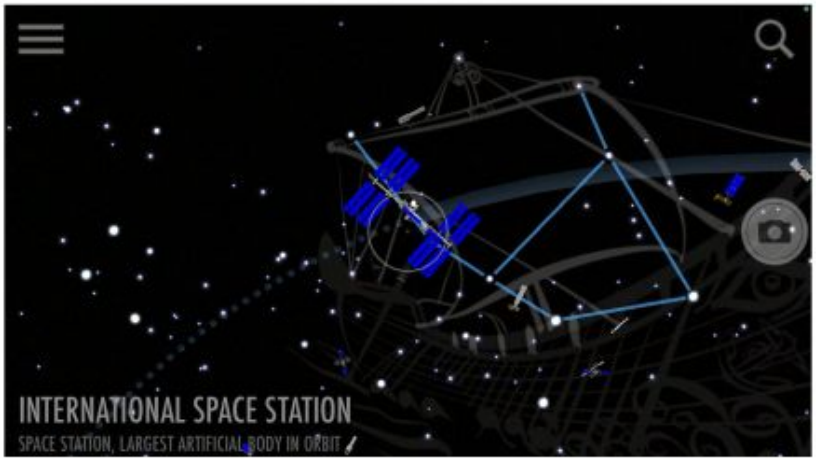



Рис. 1.2 – Інтерфейс додатку SkyView

Недоліки

Залежність від технічних параметрів пристрою: для коректної роботи додаток потребує смартфонів із підтримкою AR та високою продуктивністю.

Обмеження в залученні: хоча додаток і популярний, його використання може бути обмежене серед людей, які не використовують смартфони.

Потреба в постійному оновленні: додаток вимагає регулярних оновлень для забезпечення актуальності і точності даних.

Переваги

Зручність і доступність: мобільний додаток забезпечує легкий доступ до інформації про космос, де б ви не знаходились.

Освітній вплив: SkyView надає освітні матеріали, які допомагають користувачам дізнаватися більше про космос і астрономію.

Технологія доповненої реальності: інноваційне використання AR технології робить досвід використання додатку інтерактивним і захоплюючим.

### 1.5.3 Spot The Station

Spot The Station — це сервіс від NASA, який надає можливість користувачам отримувати повідомлення про часи, коли Міжнародна Космічна Станція (МКС) буде видима з їхньої локації. Сервіс дозволяє користувачам вибрати своє місцезнаходження і налаштувати сповіщення по електронній пошті чи SMS, інформуючи про найкращий час для спостереження МКС.

На рис. 1.3 представлено веб-інтерфейс NASA Spot The Station.

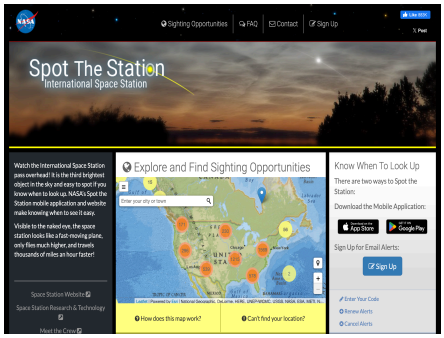



Рис 1.3 Веб-інтерфейс NASA’s Spot The Station

Переваги

Доступність: легкість отримання сповіщень робить SpotTheStation зручним для широкого кола користувачів.

Освітній вплив: сервіс сприяє підвищенню обізнаності про МКС і космічні місії.

Підтримка від NASA: спираючись на репутацію і ресурси NASA, сервіс має високий рівень довіри та надійності.

В таблиці 1 наведено аналіз порівняння аналогів.

Таблиця 1

Аналіз порівняння аналогів

| **Назва** | **Heavens-Above** | **SkyView** | **NASA's Spot the Station** | **OrbitEYE** |
|---|---|---|---|---|
| Відкритий програмний код | - | - | - | + |
| Десктопне кросплатформне ПЗ | - | - | - | + |
| Прив'язка відео-трансляції до геолокації користувача | - | - | - | + |
| Автоматичне визначення розташування користувача | - | - | - | + |
| Сповіщення по Email | - | - | + | + |

Недоліки

Обмежені функціональні можливості: порівняно з іншими астрономічними додатками, SpotTheStation може здаватися обмеженим лише сповіщеннями про видимість МКС.

Відсутність візуальної інтерактивності: не надає візуальної інформації або карт, які можуть зробити досвід більш захоплюючим і освітнім.

# 2 ОГЛЯД ЗАСОБІВ РЕАЛІЗАЦІЇ ДЕСКТОПНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ВІДЕО-ТРАНСЛЯЦІЇ МКС

## 2.1 Засоби реалізації

Розглянемо засоби розробки програмного забезпечення. Інструменти розробки програмного забезпечення – це програми та платформи, які допомагають створювати, тестувати, налагоджувати та підтримувати програмне забезпечення. Ці інструменти охоплюють широкий діапазон, включаючи інтегровані середовища розробки (IDE), редактори коду, компілятори, налагоджувачі, системи контролю версій і тестові рамки. Вони підтримують різноманітні завдання в життєвому циклі розробки програмного забезпечення, надаючи функціональні можливості, призначені для ефективного програмування та управління проектами.

Переваги використання засобів розробки програмного забезпечення включають підвищення ефективності та продуктивності процесу розробки програмного забезпечення. Вони допомагають покращити якість коду, зменшити кількість помилок і оптимізувати співпрацю між командами розробників. Ці інструменти автоматизують повторювані завдання та створюють структуроване середовище для керування складними програмними проектами. Вони мають вирішальне значення для дотримання графіків проекту, сприяння ефективній комунікації та забезпечення відповідності програмного забезпечення необхідним стандартам і функціям.

Наевдемо 10 найкращих інструментів розробки програмного забезпечення.

Visor — це інструмент керування проектами, який пропонує двонаправлену інтеграцію з Jira, що дозволяє імпортувати понад 75 типів полів Jira для комплексного керування даними та візуалізації. Завдяки цій можливості Visor є

найкращим вибором для користувачів, які прагнуть створювати власні перегляди та ефективно керувати проектами в екосистемі Jira.

Visor має унікальну здатність надавати спеціальні перегляди та надійну інтеграцію з Jira. Також Visor здатний підключатися до різних програм SaaS, створюючи уніфікований робочий простір, який задовольняє різноманітні потреби управління проектами.

Visor є найкращим інструментом для інтеграції Jira та користувальницьких переглядів, оскільки він не лише гарантує, що дані залишаються синхронізованими з вкладеною інформацією Jira, але також підтримує двонаправлену інтеграцію, дозволяючи командам керувати та ділитися звітами, які ефективно відображають складні ієрархії Jira.

Видатні функції та інтеграції візора. Функції включають API, інформаційні панелі, експорт/імпорт даних, візуалізацію даних, зовнішню інтеграцію, діаграми Ганта, багатокористувацьку роботу, керування проектами, планування та відстеження завдань, керування ресурсами та плагіни/додатки сторонніх виробників. Інтеграції включають Jira, Salesforce і HubSpot.

GitHub — це служба розміщення репозиторіїв Git, яка надає веб-кодування та можливості співпраці, які розробники використовують для створення, тестування та розгортання програм.

GitHub, якому довіряють мільйони користувачів у всьому світі, є видатним інструментом, створеним для співпраці. GitHub підвищує продуктивність і пропонує функції для розробки високоякісного програмного забезпечення. Необмежені репозиторії GitHub, спільнота з відкритим кодом, можливості автоматизації та система контролю версій підвищують швидкість розробника та допомагають вашій команді ефективно працювати разом.

Вбудовані функції безпеки GitHub виділяють цей інструмент. Платформа дає вам змогу захистити свій код за лічені хвилини та підтримувати відповідність нормам. За допомогою GitHub Actions ви можете автоматизувати процеси збірки, тестування та розгортання за допомогою суперзахищеного CI/CD. Він має рішення GitHub Copilot на основі ШІ, яке пришвидшує розробку на 55%.

Відмінні функції та інтеграції GitHub.

Функції включають можливості співпраці, вбудовану безпеку, повне середовище розробника, сканування коду, автоматизацію робочого процесу, необмежену кількість сховищ і контроль версій.

Інтеграції включають Slack, GitKraken, Atom, Disbug, Jira, Azure Boards, Codetree, Nessus і GitHub Scanner.

Bitbucket Cloud, оптимізований для розробників, які використовують Jira, є рідним інструментом Git у рішенні Open DevOps від Atlassian, який пропонує чудову інтеграцію з Jira та вбудований CI/CD.

Мільйони розробників розробляють Bitbucket завдяки його видатним можливостям, включаючи легкість співпраці в кількох командах. Це потужне рішення Git дозволяє легко керувати проблемами Jira за допомогою вбудованого інтерфейсу користувача Jira. Як результат, це допомагає мінімізувати перемикання контексту.

Bitbucket має функції перевірки коду, які допомагають вам знаходити та вирішувати проблеми перед розгортанням програмного забезпечення. Завдяки вбудованим конвеєрам CI/CD і Bitbucket Pipelines ви можете збільшити швидкість розробника шляхом автоматизації робочих процесів.

Видатні функції та інтеграції Bitbucket  Функції включають вбудований інтерфейс Jira, можливості перегляду коду, автоматичне розгортання, автоматичне сканування безпеки, автоматичне відстеження проблем і можливості співпраці.  Інтеграції включають Jira, IronGit, Makerflow, Mergetastic, Fleep, Atlas Bot, Bucket Board, CircleCI, CloudCannon і Code Dog.

Linx — це платформа розробки з низьким кодом, яка використовується командами програмного забезпечення для створення та розгортання API, автоматизації та інших програмних рішень.

Linx — це найкраще середовище розробки без коду, яке дозволяє створювати програмні рішення без коду, фреймворків чи інфраструктури. Платформа має вбудовані інструменти для керування та моніторингу того, що ви створюєте.

Linx дозволяє розробляти складні програми за допомогою розумних редакторів, попередньо створених шаблонів і знайомих концепцій кодування.

Він інтегрується з будь-яким стеком. Отже, ви можете збільшити швидкість розробки, імпортуючи ресурси з ваших баз даних. Крім того, Linx має видатні та прості процеси налагодження та широкий спектр плагінів.

У результаті розробники легко позбавляються проблем і прискорюють розробку за допомогою попередньо створених компонентів.

Видатні функції та інтеграції Linx. Функції включають редактор низького коду, автоматизацію робочого процесу, інтеграцію, вбудоване керування та моніторинг, API та мікросервіси, попередньо створені шаблони та налагодження коду. Інтеграція включає Google Drive, Microsoft Excel, Gmail, GitHub, MySQL, Microsoft Azure, Xero, MongoDB, Adobe Analytics і Amazon EC2.

Apache NetBeans надає кодове середовище та інтелектуальні інструменти для створення програм багатьма мовами. Apache NetBeans має видатні функції, які дозволяють легко реалізувати рефакторинг коду та виділити вихідний код синтаксично та семантично.

Apache NetBeans пропонує майстри, шаблони та редактори, які допомагають пришвидшити розробку та уможливити використання кількох мов. Apache NetBeans можна запускати в ОС, які підтримують Java - це крос-платформа.

Apache NetBeans можна встановити на такі популярні операційні системи, як BSD, Mac OSX, Windows і Linux. Крім того, концепція Write Once, Run Anywhere застосовується до середовища розробників Apache.

Видатні функції та інтеграції Apache NetBeans. Функції включають кросплатформні інструменти, рефакторинг коду, редактори, майстри та шаблони. Інтеграції включають DataMelt, CSS, JRebel, Zulu Enterprise, Java, PHP, Payara Enterprise, JavaScript, C++ і XRebel.

New Relic надає інженерам можливості, необхідні для моніторингу, налагодження та вдосконалення свого програмного забезпечення або всього стеку New Relic універсальний інструмент поєднує понад 30 можливостей, щоб ви

могли контролювати свій стек. New Relic розширює свої функції за допомогою 530+ інтеграцій.

Платформа керується даними та надає рішення, необхідні вашій команді для моніторингу та вирішення проблем продуктивності програмного забезпечення. New Relic має гнучку модель ціноутворення, яка дозволяє вам платити лише за те, що ви використовуєте.

Він надає інформацію, необхідну для планування, розробки та розгортання.

З кількома інструментами, вбудованими в одну платформу, New Relic дозволяє швидше виправляти речі за допомогою єдиного підключеного досвіду.

New Relic New RelicStandout функції та інтеграції. Функції включають моніторинг серверної частини, моніторинг Kubernetes, мобільний моніторинг, моніторинг продуктивності моделі, моніторинг інфраструктури, керування журналами, відстеження помилок, моніторинг мережі, керування вразливістю та моніторинг браузера.

Інтеграція включає кероване встановлення, Java, WordPress, MySQL, Tomcat, PHP, Django, .NET, Kafka та ActiveRecord. New Relic пропонує модель ціноутворення «платіть лише за те, що використовуєте», і дозволяє користувачам почати роботу безкоштовно.

AWS Cloud9, найкраще інтегроване середовище розробки (IDE), надає всі інструменти, необхідні для написання, запуску та налагодження коду за допомогою браузера. AWS Cloud9 пропонує повне середовище для розробки, включаючи редактор коду, налагоджувач і термінал.

Cloud9 підтримує популярні мови кодування, що дозволяє користувачам легко запускати проекти без необхідності інсталювати файли чи запускати додаткові конфігурації. AWS Cloud9 базується на хмарі.

Немає необхідності встановлювати локальну IDE. Це дозволяє членам вашої команди працювати з будь-якого місця. Cloud9 покращує співпрацю, оскільки команди можуть працювати разом у режимі реального часу.

Кількома клацаннями миші члени команди можуть ділитися середовищем розробників і залишатися на зв’язку в режимі реального часу за допомогою функції обміну повідомленнями.

Прискорити розробку можливо за допомогою деяких функцій, що економлять час, наприклад підказки коду, покрокове налагодження та завершення коду. Видатні функції та інтеграції AWS Cloud9.

Функції включають редактор коду, відладчик, підтримку кількох мов програмування та співпрацю в режимі реального часу. Інтеграція включає DronaHQ, SaaS customer intelligence 360, NewGenEducationApp, Ataccama ONE, Privitar, VidPowr, TouchBase, AWS, Python і CSS.

App Builder — це платформа з низьким кодом, яка допомагає командам програмістів швидше створювати корпоративні програми за допомогою попередньо створених шаблонів програм і функцій перетягування. App Builder позбавляє від складності, пов’язаної з дизайном інтерфейсу користувача і спрощує весь процес створення програмного забезпечення.

App Builder робить можливим швидку розробку програм за допомогою високоякісних інструментів, включаючи понад 60 елементів керування інтерфейсом користувача. App Builder дає змогу на 80% швидше завантажувати програму за допомогою настроюваних шаблонів і адаптивних макетів екрана.

Платформа сумісна з такими популярними інструментами дизайну, як Sketch, Figma та Adobe XD. Ви можете імпортувати свої проекти та перетворити їх на HTML-код кількома клацаннями миші. Видатні функції та інтеграції App Builder. Функції включають можливості перетягування WYSIWYG, попередньо створені шаблони та початкові макети, понад 60 елементів керування інтерфейсом користувача, високошвидкісну розробку RAD, інтеграцію та попередній перегляд вихідного коду в реальному часі. Серед інтеграцій Fitter, Stacker, GoodBarber, AppSheet, Interfaces by Zapier, Softr, Tadabase, Typebot, Amazon Honeycode і Kaspr.

Visual Studio є комплексною платформою розробки, яка пропонує широкий спектр інструментів для кодування, налагодження та розгортання програм. Visual

Studio одна з найповніших IDE із трьома основними рішеннями: Visual Studio для Windows, Visual Studio для Mac і Visual Studio Code.

Visual Studio Code — це редактор вихідного коду, сумісний із Windows, macOS і Linux. Visual Studio Code ідеально підходить для JS і веб-розробників і поставляється з широким набором розширень для підтримки інших мов і середовищ виконання.

Microsoft Visual Studio надає видатні можливості, такі як компілятори, функції завершення коду та графічні дизайнери. Це дуже швидка IDE, яка дозволяє кодувати швидше на будь-якій платформі/пристрої. Існують інструменти для співпраці, які дозволяють членам вашої команди працювати разом у режимі реального часу.

Завдяки Visual Studio можливо визначити та вирішити проблеми до того, як вони виникнуть. Видатні функції та інтеграції Microsoft Visual Studio. Функції включають IDE для Windows, IDE для Mac, інструменти завершення коду, компілятори, розширення, відстеження проблем і графічні дизайнери. Інтеграції включають Microsoft Teams, TeamSupport, Targetprocess, Time Doctor, Condenvy, Flock, LambdaTest, TMetric, ClicData та BrowserStack.

Dreamweaver, продукт від Adobe, надає інструменти веб-дизайну, які дозволяють розробникам створювати веб-сайти для будь-якого браузера чи пристрою. Dreamweaver програмне забезпечення підтримує кілька мов (HTML, CSS, JavaScript тощо) і дозволяє створювати веб-сервіси для будь-якого браузера чи пристрою.

Dreamweaver має хороший механізм кодування, який забезпечує швидке та гнучке кодування динамічних сторінок. За допомогою функції підказок коду та візуальної допомоги ви можете значно зменшити кількість помилок і прискорити процес розробки.

Існують настроювані стартові шаблони, які можна використовувати для швидкого створення веб-сторінок, наприклад, сторінок електронної комерції, електронних листів HTML, сторінок із інформацією та блогів. Крім того, цей інструмент є частиною Creative Cloud.

У результаті можливо імпортувати ресурси зі своїх бібліотек Creative Cloud і Adobe Stock. Видатні функції та інтеграції Dreamweaver. Функції включають кілька мов програмування, гнучке та швидке кодування, спрощену систему кодування, налагодження коду, настроювані шаблони, підтримку кількох моніторів для Windows, інтеграції та навчальні посібники. Інтеграція включає Adobe Creative Cloud, Adobe Photoshop, Arlo for Training Providers, SalesCart, CSS, Diib, POS2Net, Adobe Comp, Adobe Stock і OpenText xPression.

Інтерфейс користувача. Кожен інструмент із цього списку має зручний, простий та інтуїтивно зрозумілий інтерфейс користувача. Інструменти мають добре розроблені інтерфейси, завдяки чому новим користувачам дуже легко орієнтуватися.

Кнопки навігації дуже чуйні та стратегічно розташовані, щоб полегшити використання. Крім того, інтерфейси естетично привабливі. Юзабіліті Незалежно від того, наскільки потужним є інструмент, він не буде дуже корисним, якщо користувачам важко застосувати функції.

Я вибрав інструменти програмування, які не є складними або занадто технічними для використання. Розробникам потрібні інструменти, які спрощують кодування та роблять розробку менш громіздкою. Як наслідок, усі інструменти зі списку прості у використанні. Щоб почати, вам не потрібні тижні навчальних посібників. Легко зрозуміти, як працюють інструменти, і почати роботу над проектами.

Інтеграція програмного забезпечення Розробка програмного забезпечення величезна і вимагає використання багатьох інструментів і фреймворків. Однією з проблем, з якою стикаються розробники, є використання багатьох інструментів одночасно. Перемикання між інструментами вимагає розумових зусиль і забирає багато часу.

Рішенням цієї проблеми є інтеграція. Інтеграція інших інструментів у вашу основну програму розробки програмного забезпечення дає вам доступ до набору інструментів з однієї платформи. Це позбавляє від необхідності переходити від одного інструменту до іншого під час створення програмного забезпечення.

Отже, я шукав інструменти, які можна інтегрувати з широким спектром програм розробки програмного забезпечення та інструментів продуктивності.

Якщо ви прагнете створити успішне, високопродуктивне програмне забезпечення, ваш вибір інструментів розробки програмного забезпечення відіграє життєво важливу роль. Інструмент, який ви використовуєте, значною мірою визначає якість програмного забезпечення, швидкість його доставки та вартість розробки.

Задовольнитесь інструментом розробки програмного забезпечення, який має всі можливості, необхідні для вашого проекту. Хороший інструмент має комплексне середовище для розробників, технології, що відповідають майбутньому, сумісність із кількома ОС, інтеграцію тощо.

Проте уважно вивчіть наведені вище інструменти, щоб знайти той, який відповідає вашим потребам і бюджету.

Програмне забезпечення для відео-трансляції буде розроблятись у вигляді кросплатформного десктопного програмного забезпечення з графічним інтерфейсом користувача (GUI), щоб забезпечити простий та легкий доступ програми на основних десктопних ОС.

Головними завданнями програмного забезпечення є відео-трансляція руху МКС з прив’язкою до геолокації (розташування) користувача використовуючи офіційні орбітальні дані від NASA.

Десктопне програмне забезпечення

Десктопне програмне забезпечення — це категорія програм, які призначені для використання на персональних комп'ютерах, таких як настільні (десктопи) та портативні комп'ютери (лептопи). Це програми, які встановлюються локально на комп'ютері користувача та використовують його обчислювальні ресурси для виконання своїх функцій.

Основні характеристики десктопного програмного забезпечення:

Локальна установка: Відрізняється від веб-додатків, програмне забезпечення встановлюється безпосередньо на операційну систему користувача і виконується на його апаратному забезпеченні.

Використання системних ресурсів: Програма має доступ до системних ресурсів, таких як ЦПУ, пам'ять, дисковий простір і периферійні пристрої, що може забезпечити високу швидкість обробки та ефективність.

Функціональність без інтернету: Багато десктопних програм можуть працювати без безпосереднього підключення до Інтернету, надаючи користувачам можливість працювати офлайн.

Безпека: Оскільки дані зберігаються локально, зазвичай вважається, що десктопні програми можуть бути безпечнішими від веб-додатків з погляду контролю доступу до даних.

Інтерфейс користувача: Десктопні додатки часто надають більш розширені та гнучкі інтерфейси, які можуть бути краще адаптовані під потреби користувача.

Приклади десктопного програмного забезпечення:

Офісні програми: Microsoft Office, LibreOffice тощо.

Графічні редактори: Adobe Photoshop, GIMP.

Розробка програмного забезпечення: Visual Studio, Eclipse.

Медіаплеєри: VLC Media Player, Winamp.

Десктопне програмне забезпечення є фундаментальною частиною багатьох бізнес-процесів і особистого користування, надаючи потужні та налаштовувані інструменти для різноманітних завдань.

Вимоги до десктопного програмного забезпечення

Функціональні вимоги:

Автоматичне визначення розташування користувача.

Прив’язка моменту запуску відеотрансляції МКС до геолокації користувача.

Сповіщення за допомогою Email та нативно через ОС.

Нефункціональні вимоги:

Забезпечити захист від несанкціонованого доступу до веб-додатку.

Можливість працювати з різних операційних систем та браузерів.

Орбітальні ефемеридні дані.

Orbital Ephemeris Message (OEM) J2000 – це формат повідомлень, який використовується для опису орбітальних даних космічних апаратів.

Цей формат часто застосовується у аерокосмічній індустрії для обміну точними ефемеридними даними, які містять інформацію про положення і швидкість космічних об'єктів у відношенні до стандартної астрономічної системи координат, зокрема J2000.

Основні аспекти OEM J2000

Система координат J2000.

J2000 є відомою небесною системою координат, що базується на положенні небесних тіл станом на 1 січня 2000 року.

Система використовується як стандартна точка відліку для астрономічних і космічних досліджень.

Формат даних.

Орбітальні ефемериди в OEM J2000 зазвичай містять вектори стану (положення і швидкість), які описують траєкторію космічного апарата.

Дані можуть бути представлені в картеційській системі координат у вигляді компонент x, y, z для положення та компонент vx, vy, vz для швидкості.

Використання:

Ефемеридні дані використовуються для навігації космічних апаратів, здійснення корекцій їхніх траєкторій, а також для планування місій та здійснення наукових досліджень.

Вони надзвичайно важливі для забезпечення точності та надійності космічних операцій.

Інструменти та застосування:

Розробники місій та оператори космічних апаратів використовують спеціалізоване програмне забезпечення для обробки та аналізу OEM-даних.

Ці дані можуть бути імпортовані в системи управління польотами або використані для аналізу потенційних зіткнень на орбіті.

Технічні деталі

Часові інтервали:

Орбітальні дані зазвичай подаються для конкретних часових інтервалів, що дозволяє точно відстежувати зміни в орбіті.

Формати збереження:

Дані можуть бути збережені у форматах, сумісних з різними астрономічними програмами та інструментами

Golang.

Golang, також відомий як Go, є відкритою мовою програмування, створеною в Google у 2007 році Робертом Грісемером, Робом Пайком, і Кеном Томпсоном. Мова була спроектована, щоб забезпечити ефективну роботу з системами, які вимагають високої продуктивності, такими як мережеві сервери або великі розподілені системи.

Основні особливості Golang.

Простота та швидкість розробки:

Golang має чітку та просту синтаксичну структуру, що забезпечує швидке освоєння та розробку програм.

Ця мова спроектована таким чином, щоби скоротити кількість коду, необхідного для реалізації складних задач.

Статична типізація:

Go використовує статичну типізацію, що забезпечує помилки на етапі компіляції, замість помилок на етапі виконання програми.

Конкурентність:

Одна з ключових особливостей Go — це підтримка вбудованої конкурентності через горутини (горутины). Горутина — це легковаговий потік, управління яким здійснюється Go runtime.

Використання каналів для комунікації між горутинами забезпечує безпечний та ефективний обмін даними.

Швидкість виконання:

Програми, написані на Go, компілюються в машинний код, що забезпечує швидке виконання.

Go відомий своїми оптимізованими компіляторами, що генерують високопродуктивний двійковий код.

Сучасні можливості:

Мова має велику стандартну бібліотеку, що покриває багато аспектів програмування, включаючи мережеве з'єднання, паралельну обробку та роботу з текстами.

Автоматичне управління пам'яттю через збірник сміття.

Інструментарій:

Go постачається з обширним набором інструментів для форматування коду, документування та виявлення помилок.

Спільнота та підтримка:

Golang має активну спільноту розробників, яка постійно розширюється та покращує існуючі інструменти та бібліотеки.

Застосування

Веб-сервери та API:

Go широко використовується для створення високопродуктивних веб-серверів та RESTful API через його сильні мережеві бібліотеки.

Мікросервіси:

Його легкість, разом з підтримкою конкурентності, робить його відмінним вибором для мікросервісних архітектур.

Командні утиліти:

Швидке виконання та простота розповсюдження створених на Go програм також робить його популярним вибором для написання командних утиліт.

Обґрунтування вибору Golang як мови програмування

Вибір Golang як мови програмування для розробки проектів, особливо тих, що потребують високої продуктивності та ефективності управління конкурентними процесами, може бути обґрунтований низкою ключових переваг, які ця мова пропонує. Нижче представлені основні пункти, які підкріплюють вибір Golang:

1. Простота та Читабельність

Golang відомий своєю чистотою та простотою синтаксису, що сприяє легкому засвоєнню мови новими розробниками та забезпечує високу читабельність коду.

Це значно спрощує процес великомасштабної командної розробки та підтримки коду.

2. Конкурентність

Однією з вирізняючих особливостей Golang є його рідна підтримка конкурентності за допомогою горутин (легковагові потоки керовані мовою). В порівнянні з традиційними потоками, горутина вимагає менше пам'яті та має менші витрати на створення та управління, що робить їх ідеальними для реалізації великої кількості паралельних завдань.

3. Швидкість роботи

Програми, написані на Go, компілюються у машинний код, що забезпечує швидке виконання. Компіляція в машинний код дозволяє Go працювати з такою ж ефективністю, як у більш традиційних мовах, таких як C або C++.

4. Робота з мережами

Завдяки вбудованій підтримці множини мережевих протоколів та засобів, Go є відмінним вибором для розробки мережевого програмного забезпечення. Мова включає обширні бібліотеки для реалізації HTTP-серверів, клієнтів та інших мережевих сервісів, що робить її ідеальною для сучасних веб-додатків та мікросервісів.

5. Портативність та Залежності

Go пропонує великі можливості для створення статично зв'язаних бінарних файлів, що не мають зовнішніх залежностей. Це означає, що ви можете створити програму на одній платформі і легко розгортати її на інших платформах без додаткових налаштувань.

Фреймворк Fyne

Fyne є відкритим фреймворком для створення графічних користувальницьких інтерфейсів (GUI) з використанням мови програмування Go. Основна мета Fyne полягає в тому, щоб надати простий та інтуїтивно зрозумілий спосіб створення кросплатформних програм з рідними інтерфейсами на всіх підтримуваних платформах. Це робить Fyne цікавим вибором для розробників, які бажають використовувати можливості Go для створення сучасних додатків.

Основні характеристики фреймворку Fyne

1. Кросплатформність:

- Fyne дозволяє створювати додатки, які працюють на macOS, Windows, Linux, iOS та Android із єдиною кодовою базою. Завдяки цьому фреймворк забезпечує велику гнучкість для розробників.

2. Простота у використанні:

- Синтаксис Fyne розроблений таким чином, щоб спростити процес розробки GUI. Він пропонує інтуїтивно зрозумілі віджети та API для роботи з графікою, подіями тощо.

3. Легкість дизайну:

- Fyne має вбудовані теми оформлення, які дозволяють легко створювати естетично привабливі додатки без необхідності глибоких знань у графічному дизайні.

4. Модульність:

- Архітектура Fyne побудована навколо модульності, що дозволяє легко інтегрувати сторонні бібліотеки та розширювати функціональність додатків.

5. Відкритий код:

- Fyne є проєктом з відкритим вихідним кодом, що сприяє співпраці розробників з усього світу і постійному вдосконаленню фреймворку.

6. Вбудована підтримка графіки:

- Фреймворк має потужні засоби для роботи з 2D графікою, що включає відображення зображень та векторної графіки.

Приклади застосування Fyne

- Настільні додатки: Від простих утиліт до складних редакторів та інструментальних панелей.

- Мобільні додатки: Оскільки Fyne підтримує iOS та Android, його можна використовувати для розробки кросплатформних мобільних додатків.

WebView

WebView — це компонент, який дозволяє вбудовувати веб-сторінки всередині додатків як частину інтерфейсу користувача. Він використовується у

розробці мобільних і настільних програм для показу веб-контенту без необхідності виходити з додатку до браузера.

WebView може використовуватися для відображення будь-якого онлайн-контенту, такого як веб-сторінки, або для розробки цілих частин інтерфейсу на основі HTML, CSS і JavaScript.

Основні можливості та застосування WebView:

1. Інтеграція веб-контенту: WebView дозволяє розробникам вбудовувати веб-сторінки безпосередньо в мобільні або настільні додатки, забезпечуючи плавне відображення веб-контенту в межах нативного інтерфейсу.

2. Кросплатформність: Застосування WebView спрощує створення кросплатформних додатків, оскільки веб-технології є універсальними і підтримуються на більшості платформ і пристроїв.

3. Взаємодія з JavaScript: Більшість реалізацій WebView дозволяють інтерактивну взаємодію з JavaScript, даючи можливість додаткам викликати JavaScript-функції та отримувати від них дані, що робить можливим динамічну модифікацію веб-сторінок.

4. Розширення функціоналу додатків: Використання WebView дозволяє розширити можливості нативних додатків, інтегруючи в них складні веб-базовані інструменти та системи, як-от платіжні шлюзи, інтерактивні картографічні сервіси, соціальні медіа та інше.

5. Локальний веб-контент: WebView також може використовуватися для показу локально збереженого HTML-контенту, що корисно для додатків, які потребують роботи офлайн або мають вбудовану інструкцію чи допомогу.

Виклики та обмеження:

1. Безпека: Використання WebView може відкрити додаток для певних веб-загроз, таких як міжсайтовий скриптинг (XSS) або перехоплення сесій, особливо якщо вміст завантажується з ненадійних джерел.

2. Продуктивність: WebView може споживати значні ресурси системи, особливо коли він відображає складні веб-сторінки, що може призвести до зниження загальної продуктивності додатку.

3. Сумісність із функціями: Не всі можливості веб-браузера доступні у WebView, тому деякий функціонал може бути обмежений або недоступний.

Платформи, які підтримують WebView:

- Android: через клас `WebView` у Android SDK.

- iOS: за допомогою `WKWebView` в iOS SDK.

- Windows: через `WebView` контроль у Windows UI бібліотеці.

Розуміння потенціалу та обмежень WebView є ключовим для ефективного використання цього інструменту в розробці додатків.

HTML

HTML, або HyperText Markup Language, є стандартною мовою розмітки, яка використовується для створення веб-сторінок.

HTML описує структуру веб-сторінки, використовуючи різні елементи та теги, які дозволяють вбудовувати текст, зображення, відео, форми, і інші мультимедійні елементи.

CSS.

CSS є стандартною мовою розмітки, яка використовується для створення веб-сторінок. HTML описує структуру веб-сторінки, використовуючи різні елементи та теги, які дозволяють вбудовувати текст, зображення, відео, форми, і інші мультимедійні елементи.

## 2.2 Модель багатопоточності

Модель багатопоточності в мові програмування Go, також відомій як Golang, є однією з найсильніших і найбільш визначальних рис цієї мови. Вона базується на концепціях горутин (goroutines) і каналів (channels), які дозволяють легко створювати і управляти конкурентними процесами. Нижче наведено детальний опис цієї моделі, включаючи основні принципи, особливості та приклади використання.

Основні принципи моделі багатопоточності Go

Горутини (goroutines):

Що таке горутина? Горутина - це легка, керована мови Go нитка виконання. Вона схожа на потоки (threads), але є значно легшою у використанні та менш ресурсомісткою.

Створення горутин: Горутини створюються за допомогою ключового слова go, після якого слідує виклик функції або анонімної функції. Наприклад:

```
go func() {
    fmt.Println("Hello from a goroutine!")
}()
```

Виконання: Горутини виконуються конкурентно з іншими горутинами в одній і тій же адресному просторі. Go runtime автоматично керує їх виконанням.

Канали (channels):

Що таке канал? Канал - це типізований конвеєр, який дозволяє горутинам спілкуватися між собою, обмінюючись даними.

Створення каналів: Канали створюються за допомогою функції make. Наприклад:

```
ch := make(chan int)
```

Відправка і отримання даних: Дані можуть бути відправлені в канал за допомогою оператора <-. Наприклад:

```
ch <- 42  // відправка даних
value := <-ch  // отримання даних
```

Буферизовані і небуферизовані канали: Канали можуть бути буферизованими (мають певну ємність) і небуферизованими (без ємності). Буферизовані канали дозволяють відправнику не блокуватись, поки буфер не заповниться.

Особливості моделі багатопоточності Go

Легкість горутин:

Горутини є значно легшими за традиційні системні потоки. Створення тисяч горутин є звичайною практикою і не призводить до значного споживання пам'яті або ресурсів процесора.

Автоматичне управління горутинами:

Go runtime автоматично розподіляє горутини по доступним процесорним ядрам, що дозволяє ефективно використовувати апаратні ресурси. Механізм планувальника (scheduler) Go відповідає за прозоре управління виконанням горутин.

Модель CSP (Communicating Sequential Processes):

Модель конкурентності Go базується на CSP, що заохочує спілкування між процесами за допомогою передачі повідомлень (каналів), а не спільного використання пам'яті.

Відсутність явного блокування:

Канали та горутини дозволяють уникати явного використання примітивів синхронізації, таких як м'ютекси або семафори, що робить код менш схильним до помилок, пов'язаних з блокуванням.

Приклади використання багатопоточності в Go

Простий приклад з горутиною:

```
package main

import (
    "fmt"
    "time"
)
func say(s string) {
    for i := 0; i < 5; i++ {
        time.Sleep(100 * time.Millisecond)
        fmt.Println(s)
    }
}
```

```
func main() {
    go say("Hello")
    say("World")
}
```

У цьому прикладі функція say виконується як горутина, що дозволяє паралельне виконання коду.

Приклад з каналами:

```
package main
import "fmt"
func sum(s []int, c chan int) {
    sum := 0
    for _, v := range s {
        sum += v
    }
    c <- sum // відправка результату в канал
}

func main() {
    s := []int{7, 2, 8, -9, 4, 0}

    c := make(chan int)
    go sum(s[:len(s)/2], c)
    go sum(s[len(s)/2:], c)
    x, y := <-c, <-c // отримання результатів з каналу
    fmt.Println(x, y, x+y)
}
```

У цьому прикладі використовується канал для синхронізації і обміну даними між горутинами.

Модель багатопоточності в Go забезпечує ефективний і легкий спосіб створення конкурентних програм. Використання горутин і каналів дозволяє значно спростити розробку паралельних додатків, забезпечуючи при цьому високу продуктивність і надійність. Go runtime автоматично управляє розподілом горутин і ефективно використовує апаратні ресурси, що робить Go чудовим вибором для розробки програм, що потребують високої продуктивності і масштабованості.

# 3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## 3.1 Вибір технологій

Розробка програмного забезпечення для відеотрансляції Міжнародної Космічної Станції (МКС) включає створення системи, яка дозволяє користувачам отримувати відео-трансляції в реальному часі, коли МКС пролітає максимально близько до їхнього місцезнаходження. У цьому розділі буде розглянуто етапи розробки, архітектуру системи, вибір технологій, а також основні компоненти та їх функціональні можливості.

При виборі технологій для розробки програмного забезпечення було враховано наступні критерії:

Висока продуктивність та ефективність обробки даних.

Можливість обробки великих обсягів даних у режимі реального часу.

Зручність використання та можливість розширення функціональності.

Кросплатформенність.

На основі цих критеріїв було обрано наступні технології:

Мова програмування Golang: забезпечує високу продуктивність, конкурентність та легкість розробки.

Орбітальні ефемеридні дані NASA: використовуються для точного визначення положення МКС.

HTML/CSS/JavaScript: для створення користувацького інтерфейсу та вбудованих веб-компонентів.

Архітектура системи

Архітектура програмного забезпечення для відеотрансляції МКС складається з декількох основних компонентів:

Серверна частина: відповідальна за обробку запитів користувачів, обчислення позиції МКС та управління відео-потоком.

Клієнтська частина: веб-інтерфейс, який дозволяє користувачам взаємодіяти з системою та переглядати відео-трансляцію.

База даних: зберігає інформацію про користувачів, їхні запити та історію переглядів.

Основні компоненти системи

Серверна частина

Обчислювальний модуль: Використовує орбітальні ефемеридні дані та формулу гаверсинуса для розрахунку точного часу і місця проліту МКС.

Модуль обробки відео: Отримує відео-потоки з камер МКС, обробляє їх і транслює користувачам у реальному часі.

API сервер: Забезпечує комунікацію між серверною і клієнтською частинами, обробляє запити від клієнтів.

Клієнтська частина

Інтерфейс користувача: Реалізований за допомогою HTML, CSS і JavaScript.

Забезпечує зручний доступ до функціоналу програми, дозволяє переглядати відео-трансляції та отримувати сповіщення про проліт МКС.

Модуль інтерактивності: Взаємодіє з API сервером для отримання актуальної інформації про МКС та відео-потоки.

Етапи розробки

Планування

Визначення вимог до системи та функціональних можливостей.

Розробка технічного завдання та вибір відповідних технологій.

Проектування

Створення архітектурної схеми системи.

Визначення основних компонентів та їх взаємодії.

Реалізація

Розробка серверної частини на Golang.

Розробка клієнтського інтерфейсу за допомогою HTML, CSS та JavaScript.

Інтеграція з орбітальними ефемеридними даними NASA.

Тестування

Тестування всіх компонентів системи на відповідність вимогам.

Виявлення та усунення помилок.

Розгортання

Розгортання системи на сервері.

Налаштування бази даних та інших компонентів.

Підтримка та оновлення

Регулярне оновлення даних та програмного забезпечення.

Підтримка користувачів та вдосконалення функціональності системи.

Розробка програмного забезпечення для відеотрансляції МКС є складним, але цікавим процесом, що включає вибір відповідних технологій, проектування архітектури системи, реалізацію та тестування всіх компонентів. Використання мови програмування Golang, орбітальних ефемеридних даних NASA та веб-технологій дозволяє створити ефективну та зручну систему для користувачів, що забезпечує високу продуктивність та надійність роботи.

### 3.2 Принципи побудови архітектури

Опишемо основні принципи архітектурного проектування програмного застосунку, які забезпечують його ефективність, масштабованість та надійність. Архітектура програмного застосунку не тільки впливає на його продуктивність і стабільність, але й визначає легкість підтримки та розширення в майбутньому.

Переличимо принципи архітектури програмного застосунку.

Модульність.

Модульна архітектура дозволяє розділити програмний продукт на незалежні блоки (модулі), кожен з яких відповідає за певну функціональність. Це спрощує розробку, тестування та підтримку кожного компонента окремо. Модулі можуть розвиватися незалежно один від одного, що робить можливим паралельну роботу різних команд над різними аспектами проекту.

Використання шаблонів проектування.

Шаблони проектування, такі як MVC (Model-View-Controller) або MVP (Model-View-Presenter), допомагають структурувати програмний код таким чином, що він стає більш зрозумілим, легшим для тестування і підтримки. Використання цих шаблонів також сприяє відділенню бізнес-логіки від інтерфейсу користувача, що є важливим для веб-додатків та додатків з багатошаровою архітектурою.

Масштабованість.

Архітектура має бути спроектована до зростаючих обсягів даних та кількості користувачів. Це може включати використання розподілених систем, кешування, балансування навантаження та інші технології, які дозволяють розширювати ресурси без зниження продуктивності системи.

На рис. 3.1 наведено діаграма класів процесори.

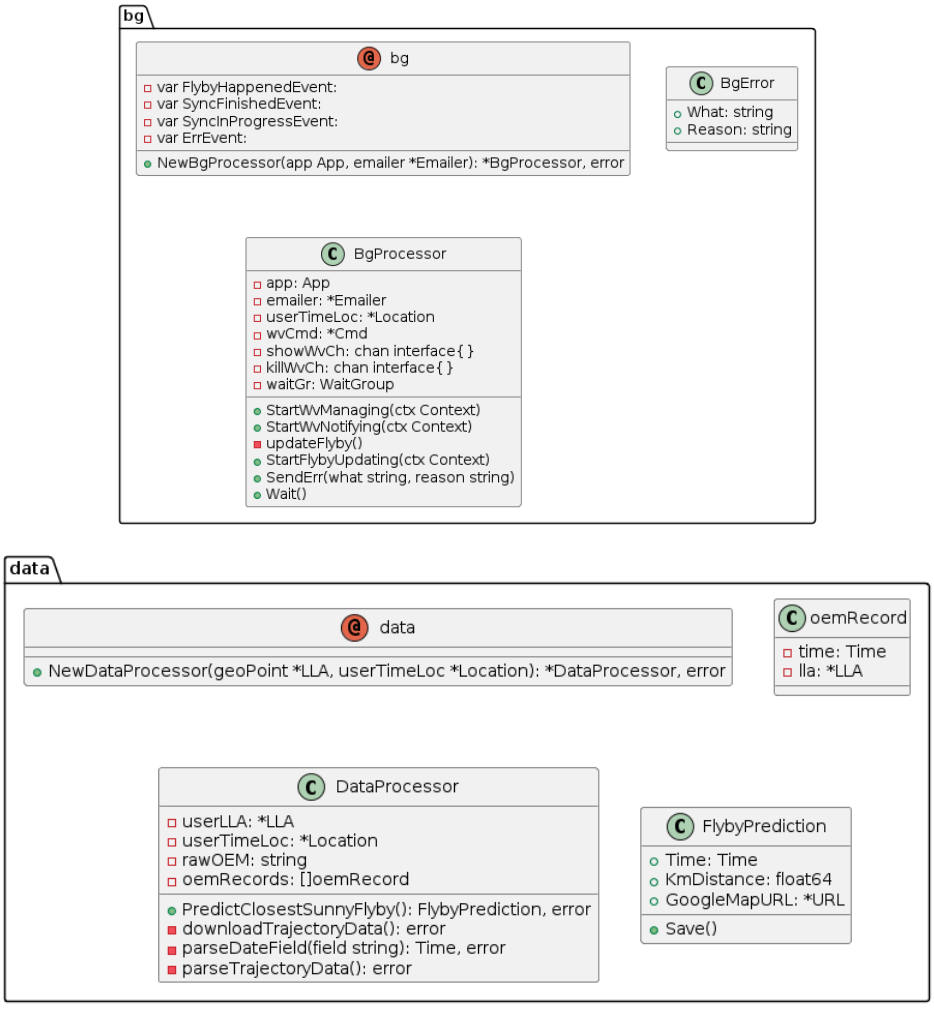



Рис. 3.1 Діаграма класів процесори

Надійність та відновлення після збоїв.

Архітектура повинна забезпечувати високий рівень надійності та підтримувати механізми для швидкого відновлення після збоїв. Це може включати реплікацію використання стійких до збоїв компонентів і автоматизоване перемикання на резервні системи у випадку відмови.

На рисунку 3.2 представлена діаграма послідовності.

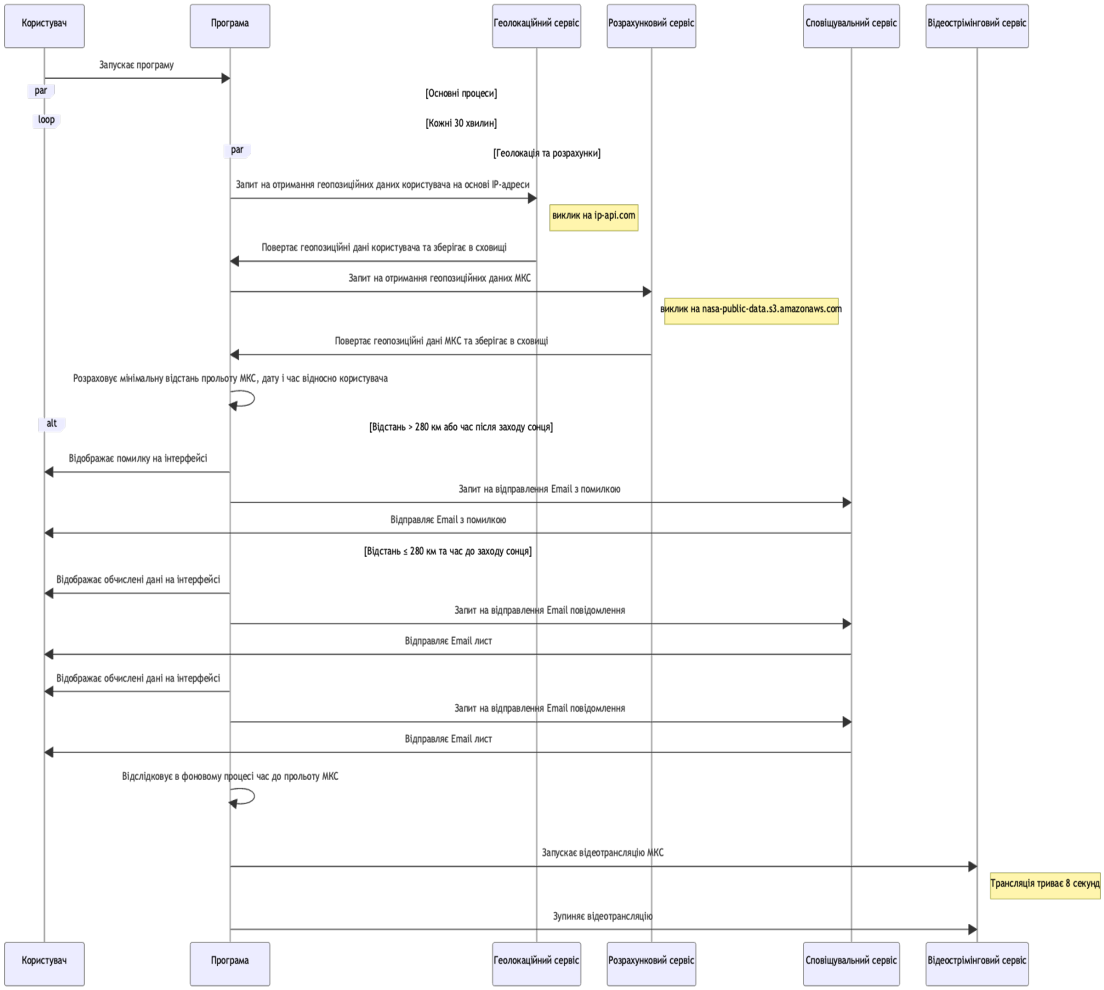



Рис. 3.2 Діаграма послідовності

Безпека.

Оскільки системи резервного копіювання часто обробляють конфіденційну інформацію, архітектура має включати комплексні заходи безпеки. Це охоплює шифрування даних, управління доступом, аудит та відповідність нормам і стандартам безпеки.

Інтеграція.

Архітектура має передбачати можливість інтеграції з іншими системами і сервісами. Це включає розробку API, які дозволяють легко підключати зовнішні модулі або інтегруватися з іншими застосунками, що підвищує гнучкість та розширюваність системи.

Екранні форми представлені на рис. 3.3, рис.3.4.

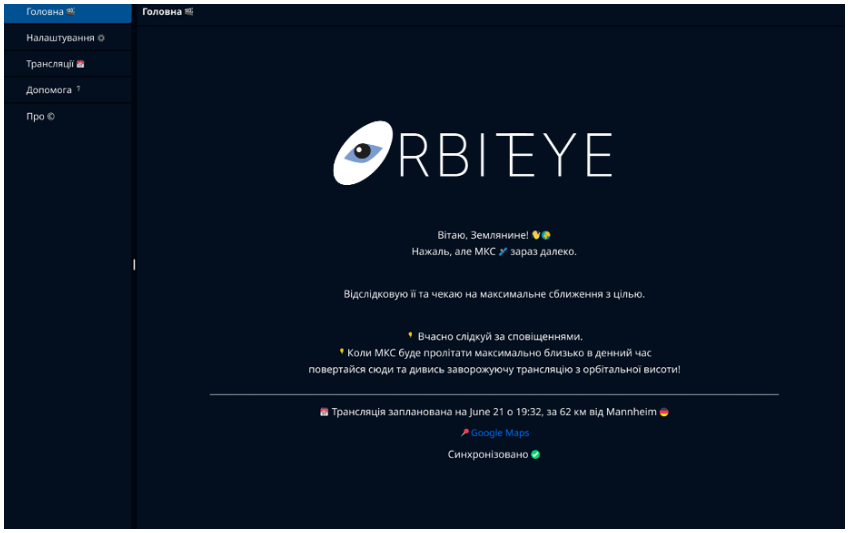



Рис. 3.3 Екранні форми

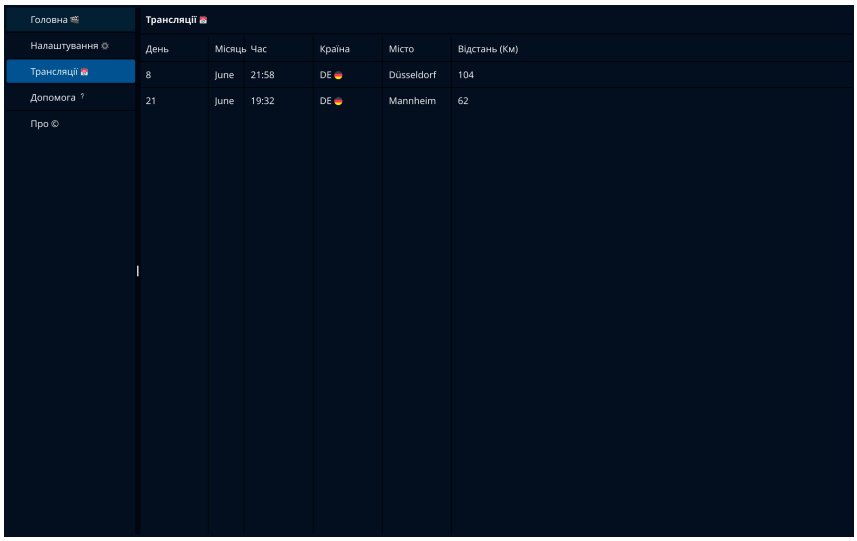



Рис. 3.4 Екранні форми

Розробка архітектури програмного забезпечення з урахуванням цих принципів забезпечує створення стійкої, ефективної та легко адаптованої системи, яка зможе відповідати потребам бізнесу як у короткостроковій так і в довгостроковій перспективі.

# 4 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## 4.1 Тестування десктопного програмного забезпечення

Тестування десктопного програмного забезпечення є важливим етапом у розробці програм, призначених для роботи на персональних комп'ютерах. Це включає в себе ряд методик і підходів для забезпечення якості, функціональності та надійності програмного забезпечення. Нижче наведено основні аспекти та види тестування десктопного ПЗ:

Основні аспекти тестування десктопного ПЗ:

Функціональне тестування:

Перевіряє, чи всі функції програми працюють відповідно до специфікацій.

Використовуються різні техніки, такі як модульне тестування, інтеграційне тестування, системне тестування.

Нефункціональне тестування:

Включає тестування продуктивності, безпеки, зручності використання та сумісності.

Перевіряє, наскільки добре програмне забезпечення працює під різними умовами.

Тестування продуктивності:

Оцінює

продуктивність програмного забезпечення під навантаженням.

Включає в себе тестування швидкодії, стрес-тестування, тестування стабільності та масштабованості.

Тестування безпеки:

Виявляє вразливості в програмному забезпеченні.

Включає аналіз ризиків, перевірку захисту даних, тестування на проникнення (пентестинг).

Тестування зручності використання (usability testing):

Оцінює зручність інтерфейсу користувача.

Включає юзабіліті-тестування, A/B-тестування, опитування користувачів.

Тестування сумісності:

Перевіряє, як програма працює на різних операційних системах та конфігураціях апаратного забезпечення.

Включає крос-платформне тестування, тестування на різних версіях операційних систем.

Види тестування десктопного ПЗ:

Модульне тестування (Unit Testing):

Перевірка окремих модулів або компонентів програми.

Зазвичай виконується розробниками програмного забезпечення.

Інтеграційне тестування (Integration Testing):

Тестування інтеграції між модулями.

Перевіряє, як модулі взаємодіють один з одним.

Системне тестування (System Testing):

Тестування повної інтегрованої системи.

Охоплює як функціональні, так і нефункціональні аспекти.

Регресійне тестування (Regression Testing):

Перевірка, чи нові зміни не вплинули на раніше працюючі функції.

Важливо при випуску оновлень та патчів.

Прийомне тестування (Acceptance Testing):

Тестування, яке проводиться з метою визначення, чи відповідає програма вимогам і готова до випуску.

Включає альфа- і бета-тестування.

Інструменти для тестування десктопного ПЗ:

Автоматизація тестування:

Selenium: В основному використовується для веб-додатків, але також може бути адаптований для десктопних додатків.

TestComplete: Підтримує автоматизацію тестування для різних видів додатків, включаючи десктопні.

Ranorex: Інструмент для автоматизації тестування, який підтримує десктопні, веб та мобільні додатки.

Інструменти для регресійного тестування:

JUnit, NUnit: Використовуються для модульного та регресійного тестування.

Jenkins: Система CI/CD, яка може інтегрувати різні інструменти автоматизації.

Інструменти для тестування продуктивності:

JMeter: Хоча здебільшого використовується для тестування веб-додатків, також може бути налаштований для десктопних додатків.

LoadRunner: Підтримує тестування продуктивності для різних типів додатків.

Важливі етапи тестування десктопного ПЗ:

Планування тестування:

Визначення обсягу і підходів до тестування.

Розробка тестового плану та сценаріїв.

Розробка тестових випадків:

Створення докладних тестових випадків, які охоплюють усі можливі сценарії використання.

Виконання тестування:

Виконання тестів вручну або за допомогою автоматизованих інструментів.

Документування результатів.

Аналіз результатів:

Оцінка знайдених дефектів.

Розробка плану виправлення та повторне тестування.

Звітування про тестування:

Підготовка звітів про результати тестування.

Обговорення результатів з командою розробників та замовниками.

Виклики тестування десктопного ПЗ:

Різноманітність платформ і конфігурацій:

Необхідність тестування на різних операційних системах і апаратних конфігураціях.

Проблеми з продуктивністю:

Потреба в тестуванні на продуктивність у різних умовах.

Управління тестовими даними:

Підтримка актуальності і релевантності тестових даних.

Сумісність з іншими програмами:

Перевірка взаємодії з іншими встановленими програмами та службами.

Тестування десктопного програмного забезпечення є критично важливим етапом розробки, який допомагає забезпечити високу якість продукту, його надійність та відповідність вимогам користувачів. Використання різних видів тестування та інструментів допомагає виявити і виправити помилки на ранніх етапах, знижуючи ризики і витрати на подальші виправлення.

Тестування десктопного програмного забезпечення є важливою складовою частиною процесу розробки програмних продуктів. Десктопні програми відрізняються від веб-додатків і мобільних додатків своїми особливостями, такими як різноманітність платформ, конфігурацій, необхідність встановлення на комп'ютер користувача і взаємодія з іншими локальними програмами. Тестування цих програм включає в себе безліч аспектів, кожен з яких спрямований на забезпечення максимальної якості та функціональності кінцевого продукту.

## 4.2 Основні аспекти та види десктопного програмного забезпечення

Основні аспекти тестування десктопного ПЗ

Одним з основних аспектів тестування десктопного програмного забезпечення є функціональне тестування. Цей вид тестування передбачає перевірку всіх функцій програми на відповідність вимогам та специфікаціям. Функціональне тестування включає в себе різні методики, такі як модульне тестування, інтеграційне тестування та системне тестування.

Модульне тестування спрямоване на перевірку окремих модулів або компонентів програми, інтеграційне тестування перевіряє взаємодію між цими модулями, а системне тестування оцінює роботу всієї системи в цілому.

Нефункціональне тестування, в свою чергу, охоплює аспекти, які не стосуються безпосередньої функціональності програми, але мають велике значення для її успішної роботи. Це включає тестування продуктивності, безпеки, зручності використання та сумісності.

Тестування продуктивності передбачає перевірку роботи програми під навантаженням, визначення її швидкодії, стійкості до великих обсягів даних та масштабованості. Тестування безпеки спрямоване на виявлення вразливостей у програмному забезпеченні, захист даних та забезпечення цілісності і конфіденційності інформації.

Тестування зручності використання оцінює, наскільки інтерфейс користувача є інтуїтивно зрозумілим і легким у використанні. Тестування сумісності перевіряє, як програма працює на різних операційних системах та конфігураціях апаратного забезпечення, а також як вона взаємодіє з іншими встановленими програмами.

Види тестування десктопного ПЗ.

Тестування десктопного програмного забезпечення може бути розділене на кілька видів, кожен з яких має свої особливості та методики. Модульне тестування (Unit Testing) передбачає перевірку окремих модулів програми на відповідність специфікаціям. Це один з перших етапів тестування, який зазвичай виконується розробниками програмного забезпечення.

Інтеграційне тестування (Integration Testing) спрямоване на перевірку взаємодії між різними модулями програми. Це дозволяє виявити проблеми, які можуть виникнути при об'єднанні окремих компонентів у єдину систему. Системне тестування (System Testing) перевіряє роботу всієї інтегрованої системи, охоплюючи як функціональні, так і нефункціональні аспекти.

Регресійне тестування (Regression Testing) є важливим етапом, який перевіряє, чи не призвели нові зміни в коді до появи нових помилок або до

порушення роботи раніше працюючих функцій. Це особливо важливо при випуску оновлень та патчів, коли необхідно переконатися, що нововведення не погіршили якість програмного забезпечення.

Прийомне тестування (Acceptance Testing) виконується з метою визначення, чи відповідає програма вимогам користувачів і чи готова вона до випуску. Це включає альфа- і бета-тестування, які проводяться на пізніх етапах розробки. Альфа-тестування зазвичай виконується внутрішньою командою тестувальників, тоді як бета-тестування передбачає участь реальних користувачів, які допомагають виявити залишкові помилки і недоліки.

Інструменти для тестування десктопного ПЗ.

Існує безліч інструментів, які можуть бути використані для автоматизації та спрощення процесу тестування десктопного програмного забезпечення. Одним з таких інструментів є Selenium, який, хоча і здебільшого використовується для веб-додатків, також може бути адаптований для тестування десктопних додатків. Інші популярні інструменти включають TestComplete, який підтримує автоматизацію тестування для різних видів додатків, включаючи десктопні, та Ranorex, який дозволяє автоматизувати тестування десктопних, веб та мобільних додатків.

Інструменти для регресійного тестування, такі як JUnit і NUnit, використовуються для модульного та регресійного тестування, допомагаючи забезпечити стабільність програмного забезпечення після внесення змін. Jenkins, система для CI/CD, може інтегрувати різні інструменти автоматизації, забезпечуючи безперервне тестування та релізи.

Для тестування продуктивності можуть бути використані такі інструменти, як JMeter, який, хоча здебільшого використовується для веб-додатків, також може бути налаштований для десктопних додатків, та LoadRunner, який підтримує тестування продуктивності для різних типів додатків.

Важливі етапи тестування десктопного ПЗ.

Процес тестування десктопного програмного забезпечення включає кілька важливих етапів, кожен з яких має своє значення і мету. Перший етап — це

планування тестування, де визначається обсяг і підходи до тестування, розробляється тестовий план та сценарії. На цьому етапі важливо врахувати всі можливі сценарії використання програми, визначити ключові області, які потребують особливої уваги, та розробити відповідні методики тестування.

Розробка тестових випадків є наступним важливим етапом. Це включає створення детальних тестових випадків, які охоплюють усі можливі сценарії використання програми. Тестові випадки повинні бути достатньо детальними, щоб забезпечити всебічну перевірку всіх функцій та компонентів програми.

Виконання тестування — це етап, на якому тестові випадки виконуються вручну або за допомогою автоматизованих інструментів. На цьому етапі важливо ретельно документувати всі результати тестування, фіксувати знайдені помилки та недоліки, а також аналізувати причини їх виникнення.

Аналіз результатів тестування включає оцінку знайдених дефектів, визначення їх критичності та розробку плану виправлення. Після виправлення помилок необхідно провести повторне тестування, щоб переконатися, що всі проблеми були успішно вирішені.

Звітування про тестування є заключним етапом, який передбачає підготовку звітів про результати тестування та обговорення їх з командою розробників і замовниками. Це дозволяє визначити, наскільки програма готова до випуску, які ще недоліки потребують уваги та які заходи потрібно вжити для підвищення якості програмного забезпечення.

Виклики тестування десктопного ПЗ.

Тестування десктопного програмного забезпечення пов'язане з багатьма викликами, які можуть вплинути на якість і ефективність процесу. Одним з головних викликів є різноманітність платформ і конфігурацій, на яких може працювати програма. Це вимагає тестування на різних операційних системах, апаратних конфігураціях та версіях програмного забезпечення.

Проблеми з продуктивністю також є значним викликом, оскільки програма повинна забезпечувати стабільну і швидку роботу під різними умовами навантаження. Управління тестовими даними є ще одним важливим аспектом,

оскільки тестові дані повинні бути актуальними і релевантними для забезпечення точності результатів тестування.

Сумісність з іншими програмами також є важливим фактором, оскільки програма повинна коректно взаємодіяти з іншими встановленими програмами та службами. Це включає перевірку взаємодії з різними бібліотеками, драйверами, базами даних та іншими компонентами системи.

Тестування десктопного програмного забезпечення є комплексним процесом, який включає безліч різних аспектів і методик. Використання різних видів тестування та інструментів допомагає забезпечити високу якість продукту, його надійність та відповідність вимогам користувачів. Кожен етап тестування має своє значення і мету, від планування і розробки тестових випадків до виконання тестування, аналізу результатів і звітування.

Забезпечення якості десктопного програмного забезпечення є важливим завданням, яке вимагає ретельного підходу та уваги до деталей. Виклики, пов'язані з різноманітністю платформ, продуктивністю, управлінням тестовими даними та сумісністю з іншими програмами, роблять цей процес ще більш складним і важливим. Однак, з правильним підходом та використанням сучасних інструментів тестування, можна досягти високого рівня якості і забезпечити успішний випуск програмного забезпечення, яке задовольняє всі вимоги і очікування користувачів.

# ВИСНОВКИ

В процесі виконання кваліфікаційної роботи, було досягнуто наступних результатів:

1. Проведено аналіз існуючих методів визначення геопозиції МКС та відео-трансляції її руху, виявлено основні переваги та недоліки існуючих рішень, що дозволило визначити ключові напрями вдосконалення.

2. Орбітальні ефемеридні дані NASA були вивчені та адаптовані для використання у програмному забезпеченні, що забезпечило точність розрахунків МКС.

3. Сформулювані вимоги до програмного забезпечення для відео-трансляції МКС з урахуванням недоліків існуючих рішень.

4. Спроектуване програмне забезпечення для відео-трансляції МКС.

5. Розроблено програмне забезпечення для відео-трансляції МКС.

6. Протестовано програмне забезпечення для відео-трансляції МКС.

7. Робота пройшла апробацію:

1. Бєлий Р.Д. Аналіз наявних рішень для відеотрансляції Міжнародної Космічної Станції / Р.Д Бєлий, В.В Залива // Матеріали XI Всеукраїнської науково–практичної конференції молодих учених. Збірник тез. 16.04.2024, Київський столичний університет імені Бориса Грінченка, м. Київ – К:Київський столичний університет імені Бориса Грінченка, 2024, С. 5.

2. Бєлий Р.Д. Розробка ПЗ для відеотрансляції Міжнародної Космічної Станції / Р.Д Бєлий, В.В Залива // Застосування програмного забезпечення в інформаційно-комунікаційних технологіях: Матеріали III всеукраїнської науково-технічної конференції. Збірник тез. 24.04.2024, ДУІКТ, м. Київ – К:ДУІКТ, 2024, С. 28.

# ПЕРЕЛІК ПОСИЛАНЬ

1. https://www.mdpi.com/2072-4292/15/1/133
2. https://link.springer.com/article/10.1007/s10291-021-01167-8
3. https://eol.jsc.nasa.gov/ESRS/HDEV/Final_Report.pdf
4. https://data.nasa.gov/
5. https://ssd.jpl.nasa.gov/tools/sbdb_lookup.html
6. https://software.nasa.gov/software/NPO-41161-1
7. https://github.com/umahmood/haversine
8. https://pds.nasa.gov/
9. https://eyes.nasa.gov/apps/orrery/#/home
10. https://eol.jsc.nasa.gov/ESRS/HDEV/
11. https://eol.jsc.nasa.gov/ESRS/HDEV/files/HDEV-Jan4-1205-IEEE-Muri.pdf
12. https://www.researchgate.net/figure/Specifications-of-the-HDEV-ISS-cameras-and-application-in-school-lessons_tbl1_319008517
13. https://www.semanticscholar.org/paper/The-High-Definition-Earth-Viewing-(HDEV)-payload-Muri-Runco/b52bffaca44378a2d463685a1f9c7fae2875773c
14. https://space.skyrocket.de/doc_sdat/hdev.htm
15. https://www.amazon.com/Reference-Guide-International-Space-Station/dp/0160865174/ref=cm_cr_srp_d_product_top?ie=UTF8
16. https://space.stackexchange.com/questions/4771/why-arent-the-iss-hdev-cameras-left-on-at-night
17. https://www.kaggle.com/datasets/prise6/earth-from-iss-hdev-experiment
18. https://www.researchgate.net/figure/The-four-cameras-and-three-perspectives-of-the-High-Definition-Earth-Viewing-Mission_fig1_269335131
19. https://www.researchgate.net/figure/HDEV-cameras-and-views-Source-NASA_fig1_306514488
20. https://www.researchgate.net/publication/330113836_Columbus-Eye_-_Live-Bilder_von_der_ISS_im_Schulunterricht_-_Schlussbericht_01102013-30042017

21. https://www.researchgate.net/figure/Setup-of-the-HDEV-experiment-Source-Stefanov-et-al-2011_fig1_321274414

22.https://www.researchgate.net/publication/269335131_Columbus_Eye_High_Definition_Earth_Viewing_from_the_ISS_in_Secondary_Schools

23.https://www.researchgate.net/publication/367541728_Application_of_ISS_Remote_Sensing_Data_in_New_Media_to_Further_STEM_Education

24.https://www.researchgate.net/publication/321274414_Technical_note_Using_ISS_videos_in_earth_observation_-_Implementations_for_science_and_education

25. https://www.researchgate.net/figure/Commercial-HDEV-Camera-Models_tbl1_317701681

26.https://www.researchgate.net/publication/306514488_Earth_Observation_from_the_ISS_Columbus_Laboratory_-_an_Open_Education_Approach_to_Foster_Geographical_Competences_of_Pupils_in_Secondary_Schools

27. https://zedt.eu/my/nonsense/iss-hd-earth-viewing-experiment/

28. https://ntrs.nasa.gov/api/citations/20170000416/downloads/20170000416.pdf

29. https://ieeexplore.ieee.org/document/7943749

30. https://www.tandfonline.com/doi/pdf/10.1080/22797254.2017.1396880

# ДОДАТОК А
# ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ


**ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ**

**НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

**КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**


Розробка програмного забезпечення для відео-трансляції Міжнародної Космічної Станції (МКС), в момент, коли вона пролітає максимально близько над користувачем з використанням Golang, гаверсинуса, та орбітальних ефемеридних даних NASA

Виконав студент 5 курсу
групи ППЗ-51
Бєлий Роман Денисович
Керівник роботи
Доктор філософії (PhD), старший викладач кафедри ІПЗ Залива Віталій Вікторович

Київ – 2024

## МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи -** спрощення процесу відео-трансляції руху Міжнародної Космічної Станції за допомогою програмного забезпечення, розробленого з використанням мови програмування Golang
- **Об’єкт дослідження -** процес відео-трансляції руху Міжнародної Космічної Станції
- **Предмет дослідження –** програмне забезпечення для відео-трансляції руху Міжнародної Космічної Станції

# ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Вивчити існуючі методи визначення позиції МКС та відео-трансляції її руху.
2. Проаналізувати алгоритми для обчислення відстані між двома геопозиціяними точками на сфері.
3. Сформулювати вимоги до програмного забезпечення для відео-трансляції МКС з урахуванням недоліків існуючих рішень.
4. Спроектувати програмне забезпечення для відео-трансляції МКС.
5. Розробити програмне забезпечення для відео-трансляції МКС.
6. Протестувати програмного забезпечення для відео-трансляції МКС.

3

# АНАЛІЗ АНАЛОГІВ

| **Назва** | **Heavens-Above** | **SkyView** | **NASA's Spot the Station** | **OrbitEYE** |
|---|---|---|---|---|
| Відкритий програмний код | - | - | - | + |
| Десктопне кросплатформне ПЗ | - | - | - | + |
| Прив'язка відео-трансляції до геолокації користувача | - | - | - | + |
| Автоматичне визначення розташування користувача | - | - | - | + |
| Сповіщення по Email | - | - | + | + |

4

# ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

**Функціональні вимоги**

1. Автоматичне визначення розташування користувача.
2. Прив'язка моменту запуску відеотрансляції МКС до геолокації користувача.
3. Сповіщення за допомогою Email та нативно через ОС.

**Нефункціональні вимоги**

1. Підтримка основних десктопних ОС (macOS, Linux, Windows), кросплатформність.
2. Відкритість коду (Open Source).
3. Висока актуальність орбітальних даних.

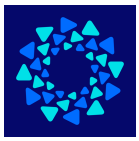
5

# ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

Mailtrap



JSON



Fyne

WebView

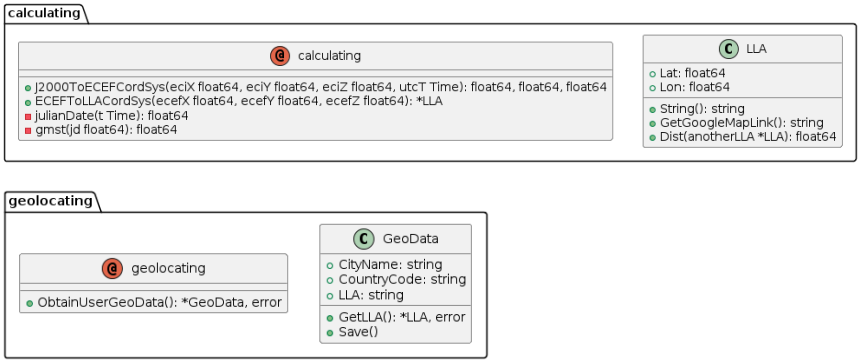

NASA's Trajectory Data

6

# ДІАГРАМА КЛАСІВ – 2 ПРОЦЕСОРИ

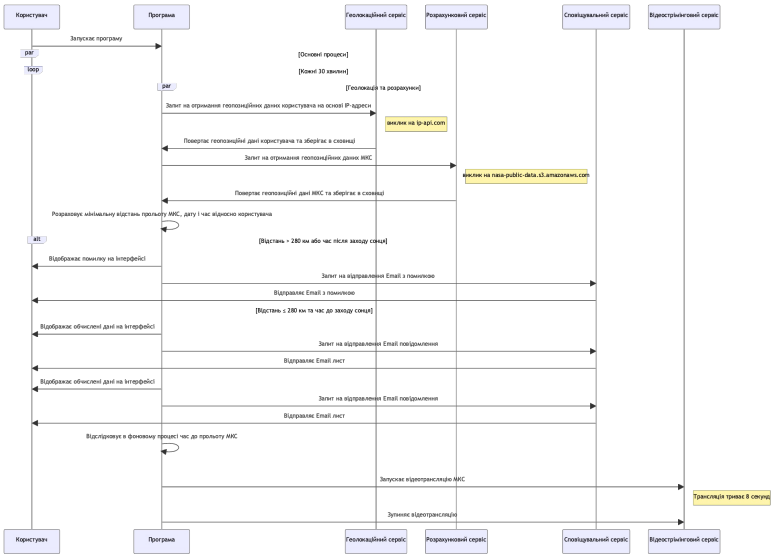

7

# ДІАГРАМА КЛАСІВ – ГЕО & РОЗРАХУНОК



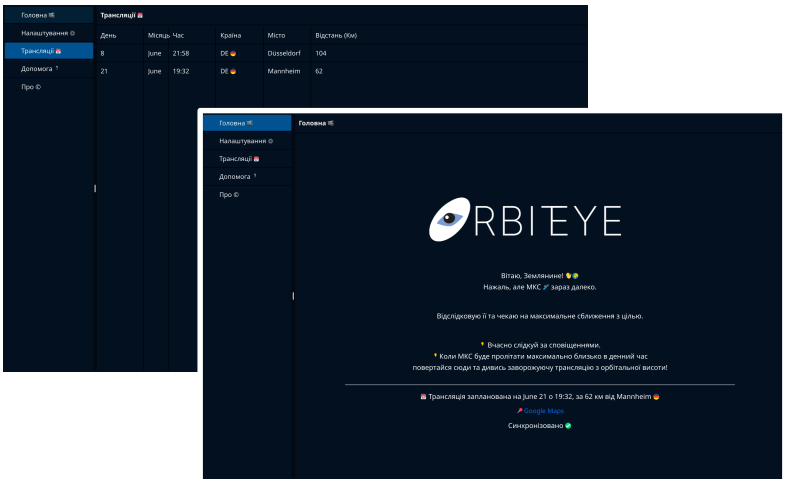

8

# ДІАГРАМА ПОСЛІДОВНОСТІ




9

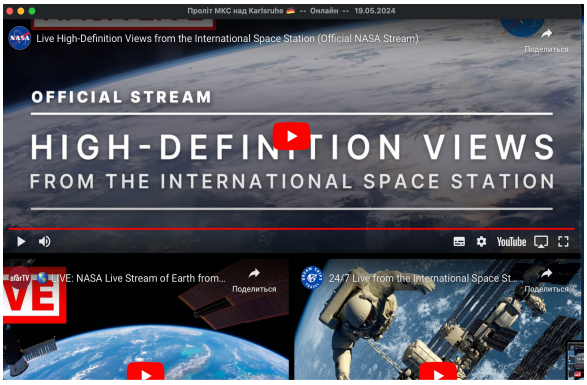
# ЕКРАННІ ФОРМИ - 1




**Головна та спостереження**

10

# ЕКРАННІ ФОРМИ - 2

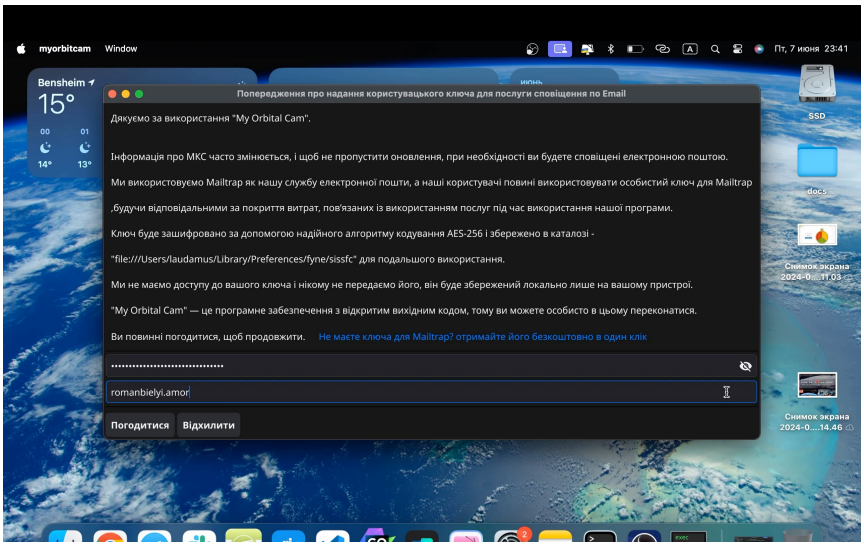



**Трансляції**


**Налаштування**

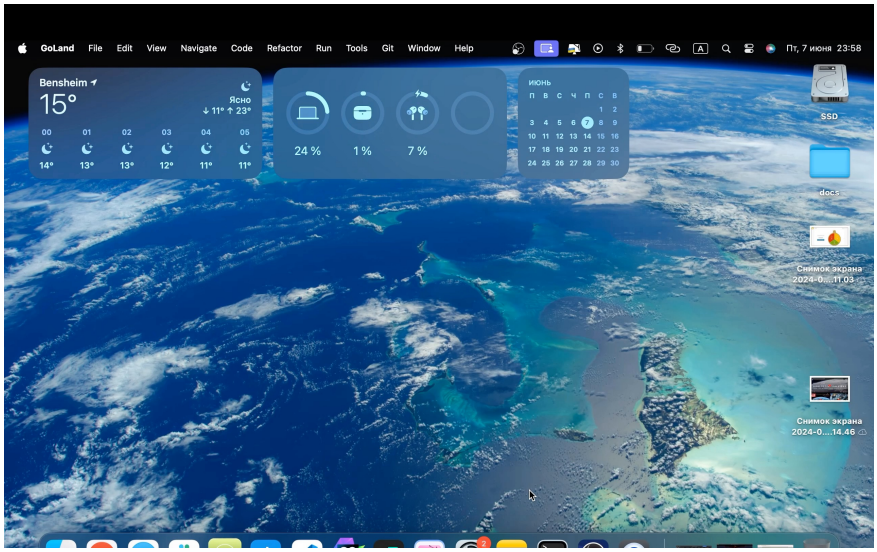
11

# ВІДЕОДЕМОНСТРАЦІЯ - 1




12

# ВІДЕОДЕМОНСТРАЦІЯ - 2




13

# АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Бєлий Р.Д. Аналіз наявних рішень для відеотрансляції Міжнародної Космічної Станції / Р.Д Бєлий, В.В Залива // Матеріали XI Всеукраїнської науково–практичної конференції молодих учених. Збірник тез. 16.04.2024, Київський столичний університет імені Бориса Грінченка, м. Київ – К:Київський столичний університет імені Бориса Грінченка, 2024, С. 5.

2. Бєлий Р.Д. Розробка ПЗ для відеотрансляції Міжнародної Космічної Станції / Р.Д Бєлий, В.В Залива // Застосування програмного забезпечення в інформаційно-комунікаційних технологіях: Матеріали III всеукраїнської науково-технічної конференції. Збірник тез. 24.04.2024, ДУІКТ, м. Київ – К:ДУІКТ, 2024, С. 28.

14

# ВИСНОВКИ

1. Проведено аналіз існуючих методів визначення геопозиції МКС та відео-трансляції її руху, виявлено основні переваги та недоліки існуючих рішень, що дозволило визначити ключові напрями вдосконалення.

2. Сформулювані вимоги до програмного забезпечення для відео-трансляції МКС з урахуванням недоліків існуючих рішень.

3. Спроектуване програмне забезпечення для відео-трансляції МКС.

4. Розроблено програмне забезпечення для відео-трансляції МКС.

5. Протестовано програмне забезпечення для відео-трансляції МКС.