**Зміст**

1. ЗАГАЛЬНІ ПОЛОЖЕННЯ

2. АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3. МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

# ЗАГАЛЬНІ ПОЛОЖЕННЯ

Основної ціллю є створення системи розгортання програмних продуктів на платформі Google Cloud Platform з використанням сервісу GitHub.

Мета розробки - узагальнення теоретичних знань отриманих впродовж учбового процессу для вирішення проблеми інтеграції та розгортання програмних продуктів.

Призначення розробки - використання проекту для прискорення процесу інтеграції та розгортання нових версій програмних продуктів за допомогою сервісу системи контроля версій git GitHub, на платформі Google Cloud Platform.

Задачі:

* отримання інформації про оновлення кодової бази від сервісу GitHub;
* виконання процесу інтеграції оновленого коду;
* виконання процесу побудови оновленого коду;
* виконання процесу розгортання оновленого програмного продукту на платформі GCP;
* ведення конфігурацій розгортання програмних продуктів;

Цілі:

* надати користувачеві (операційному інженерові або програмісту) інтерфейс для зручного ведення конфігурацій розгортання та інтеграції програмних продуктів;
* надати альтернативу наявних системам CI\CD, працюючим по методу SaaS;
* виконувати процес розгортання оновленого програмного продукту, починаючи с етапу оновлення коду до етапу встановлення та налаштування готових програм у середовищі Kubernetes платформи GCP;
* надати користувачеві зручний формат конфігурації розгортання та інтеграції оновленого програмного продукту;

# АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Програмний продукт, що задовольняє вимогам повинен бути написаний на мові, що дозволяє запуск на операційній системі Linux. Він повинен підтримувати HTTP запити по протоколу REST, з кодуванням данних за допомогою JSON, для надання користувацького API та взаємодії з GitHub. Для підтримки роботи з Docker необхідна можливість комунікації з процесом Docker через Unix socket або за допомогою викликів утиліти командного рядку. Робота з кластером kubernetes виконується за допомогою викликів утиліти kubectl або використанням API за допомогою REST або gRPC протоколу.

Для забезпечення високої доступності та зручності масштабування програмний продукт повинен бути розділений на окремі компоненти, що відповідають за окремі процеси, для чого краще за все використовувати мікросервісний підхід, для обміну інформацією між мікросервісами використовувати gRPC з кодуванням данних у форматі Protobuf.

## Аналіз успішних IT-проектів

Серед систем та сервісів неперервної інтеграції даний проект має таких важливих конкурентів як CircleCI та Jenkins

Серед інструментів для автоматизації та поліпшення процесу неперервного розгортання у середовищі Kubernetes даний проект має такого основного конкурента як Helm. Також можна виділити утиліту deploy-to-kube

CircleCI - сервіс неперервної інтеграції з підтримкою GitHub. У порівнянні з даним проектом постачається у двох видах - хмарному та для власних серверів. Завдяки хмарному постачанні адміністратору не потрібно самостійно встановлювати та налагоджувати CircleCI. На відміну від даного проекту має більш гнучку систему управління правами користувачів. Але на відміну від CircleCI, даний проект має пряму підтримку та інтеграцію з Kubernetes, тоді як з CircleCI адміністратор повинен сам налагоджувати інтеграцію. Також для даного проекту не потрібно використовувати додаткові мови опису процесу інтеграції, так як для цього використовується сценарій побудови Docker контейнеру - Dockerfile

Jenkins - програмний набір для неперервної інтеграції. У порівняння з даним проектом має підтримку дуже великої кількості можливостей та інтеграцій за допомогою сторонніх додатків - плагінів. Але на відміну від даного проекту потребує більш важкого процесу встановлення та налагодження, так як постачається у вигляді головного та дочірніх сервісів, що потрібно спочатку встановити на відповідні машини та об’єднати у єдину систему. Також у порівнянні з даним проектом, Jenkins використовує складну мову для опису процесу інтеграції, що уповільнює налагодження неперервної інтеграції. Docker та Kubernetes не підтримуються безпосередньо Jenkins, тому робота з ними відбувається через плагіни, що супроводжується додатковим налаштуваннями та можливими проблемами з роботою самих плагінів.

# Моделювання та конструювання програмного забезпечення

## Моделювання та аналіз програмного забезпечення

Розроблена система виконується під операційними системами на базі ядра Linux. Використання під операційними системами Windows та macOS можливе, але не було протестовано.

Для розробки браузерної частини системи було застосовано мову Typescript, з використанням фреймворку Angular 8. Використано паттерни MVC та Dependency Injection, які дозволяють зручно та швидко додавати новий функціонал та відокремлювати різні частини коду, оскільки MVC дозволяє відокремити дані від бізнес-логіки та від логіки відображення, а Dependency Injection видяляє строгі зв`язки між компонентами, дозволяючи швидко замінити одні компоненти на інші з аналогічним функціоналом. Взаємодія з серверною частиною відбувається по протоколу HTTP REST з кодуванням даних за допомогою JSON.

Для розробки серверної частини було застосовано мову Go. Використано паттерн Dependency Injection. Окремі функціональні компоненти розділені між собою за допомогою мікросервісного підходу. Взаємодія між мікросервісами відбувається по протоколу gRPC з кодуванням даних за допомогою Protobuf, а також за допомогою черги повідомлень RabbitMQ. Даний підхід дозволяє легко підтримувати окремі мікросервіси, зручно та швидко додавати новий функціонал та забезпечує високу надійність та масштабованість системи.