UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

Grado en Ingenieria Informatica

Generación automática de ficheros de configuración para el despliegue de servicios en Docker

Automatic generation of configuration files for deploying services in Docker.

Realizado por
Marcos Domínguez Moreno

Tutorizado por
José Miguel Horcas Aguilera

Departamento
Lenguajes y Ciencias de la Comunicación

MÁLAGA, Octubre de 2024

UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADUADA/O EN TECNOLOGÍAS DE LA INFORMACIÓN

Generación automática de ficheros de configuración para el despliegue de servicios en Docker

Automatic generation of configuration files for deploying services in Docker

Realizado por
Marcos Domínguez Moreno

Tutorizado por
José Miguel Horcas Aguilera

Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA
MÁLAGA, OCTUBRE DE 2024

Fecha defensa: Diciembre de 2024

Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Keywords: **A, B, C**

Resumen

En la actualidad, el desarrollo de software y la implementación de servicios en la nube han experimentado un crecimiento exponencial. Las empresas buscan constantemente mejorar sus procesos de desarrollo y despliegue para ofrecer productos y servicios de alta calidad en el menor tiempo posible. En este contexto, herramientas como Docker y metodologías como la Integración Continua (CI) se han convertido en componentes esenciales para lograr estos objetivos.

Docker permite la creación de contenedores que encapsulan aplicaciones y sus dependencias, asegurando que el software se ejecute de manera consistente en cualquier entorno. Por otro lado, la Integración Continua facilita la detección temprana de errores mediante la automatización de pruebas y despliegues, lo que contribuye a un ciclo de desarrollo más ágil y eficiente.

Este trabajo de fin de grado se centra en la generación automática de archivos de configuración para el despliegue de servicios en Docker. El objetivo principal es desarrollar una herramienta que simplifique y automatice el proceso de configuración, permitiendo a los desarrolladores centrarse en la creación de funcionalidades y mejoras en lugar de en tareas repetitivas y propensas a errores.

A lo largo de este documento, se presentará una revisión de la literatura relacionada con Docker, Integración Continua y generación automática de configuraciones. Además, se describirá el diseño y la implementación de la herramienta propuesta, así como los resultados obtenidos y las conclusiones derivadas del proyecto.

Palabras clave: SPL, Docker , Despliegues Servicios , Desarrollo web, Integración Continua , Configuración , feature models , Linea de Productos

Índice

1 Introduction

1.1. Motivación

Las herramientas de automatización de tareas son fundamentales en el desarrollo de software permiten mejorar la eficiencia, la calidad y la consistencia en la producción de código. Al relegar tareas repetitivas y propensas a errores a herramientas automatizadas, los desarrolladores pueden centrarse en tareas más creativas y de mayor valor añadido.

Parte del ciclo de vida en un desarrollo de software implica la configuración y el despliegue de servicios en entornos de producción y desarrollo. La configuracion y despliegue requiere de la creación de archivos de configuración específicos, como los archivos Dockerfile y .dockerignore, que definen cómo se construye una imagen de un contenedor Docker.

En el contexto de la tecnología de contenedores, Docker se ha convertido en una de las plataformas más populares para la creación, el despliegue y la gestión de aplicaciones en entornos contenerizados. Sin embargo, la configuración de servicios en Docker puede ser una tarea compleja y propensa a errores, especialmente cuando se manejan múltiples servicios y configuraciones.

La motivación para realizar este proyecto surge de la necesidad de simplificar y automatizar el proceso de generación de archivos de configuración para el despliegue de servicios en Docker. En la actualidad, la creación manual de estos archivos dockerfile puede ser una tarea repetitiva y propensa a errores al no conocer el usuario de toda la informacion necesaria para la correcta configuracion de los servicios,

Al desarrollar una herramienta que automatice este proceso, se busca mejorar la eficiencia y la precisión en la configuración de entornos de contenedores, facilitando así el trabajo de desarrollo y administracion.

1.2. Objetivo

La meta principal de este proyecto es desarrollar una herramienta que permita la generación automática de archivos de configuración para el despliegue de servicios en Docker. Siguiendo las pautas recomendadas para la generacion de dichos archivos

1.3. Resultados Esperados

Dada la gran cantidad de opciones de configuración disponibles en para la generacion de ficheros de configuracion, el desarrollo de esta herramienta presenta varios desafíos técnicos y conceptuales. El primero de ellos es la magnitud de las opciones de configuración disponibles, que pueden variar en función de las necesidades y los requisitos de cada servicio. dar una covertura a todos los escenarios posibles es una tarea inabarcable, por lo que se ha optado por centrarse en los casos de uso más comunes y en las configuraciones más utilizadas en la práctica. Sin embargo se espera que la herramienta sea lo suficientemente flexible y extensible como para permitir la incorporación de nuevas plantillas y configuraciones en el futuro.

1.4. Estructura del documento

Por definir

2

Estado del Arte

En este apartado se presentan los conceptos clave necesarios para comprender la situacion actual, asi como las soluciones existentes.

2.1. Contenedores

Los contenedores son paquetes ligeros que incluyen el código de las aplicaciones junto con sus dependencias, como versiones concretas de entornos de ejecución de ciertos lenguajes de programación y bibliotecas para ejecutar los servicios de software.

2.1.1. Porque se usan los contenedores

Los contenedores constituyen un mecanismo de empaquetado lógico en el que las aplicaciones pueden extraerse del entorno en que realmente se ejecutan. Esta desvinculación facilita el despliegue uniforme de las aplicaciones basadas en ellos con independencia de que el entorno sea un centro de datos privado, la nube pública o el portátil personal de un desarrollador.

2.2. Imagenes

Las imágenes de contenedores son plantillas de solo lectura que contienen el código de la aplicación, las bibliotecas, las dependencias y otros archivos necesarios para ejecutar un contenedor.

2.3. Docker

Docker es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos Docker es una

plataforma de código abierto que permite a los desarrolladores crear, desplegar y ejecutar aplicaciones en contenedores.

2.3.1. Dockerfile

El archivo Dockerfile es un archivo de texto que contiene una serie de instrucciones que Docker utilizará para construir una imagen de contenedor. Usualmente la construccion de un archivo de configuracion se realiza a partir de una imagen base, que puede ser una imagen oficial de Docker o una imagen personalizada creada por el usuario. A dicha imagen se añaden las instrucciones necesarias para instalar las dependencias y configurar el entorno de ejecución de la aplicación, lo que viene a representar el contenido de un archivo Dockerfile.

2.3.2. Buenas Prácticas Docker Images

Para la creacion de los ficheros dockerfile es recomendable seguir una serie de buenas practicas para garantizar la eficiencia y la seguridad en el despliegue de servicios en Docker. Estas practicas se recojen en la documentacion oficial de Docker CITA

2.4. Problematica

La dificultad para crear imagenes reside en la naturaleza de la aplicacion que queremos llevar a un contenedor, existen una gran cantidad de escenarios donde si bien la tarea es sencilla, esta puede escoder una complejidad implicita de la que el usuario no es consciente.

Estos apartados repercuten en el tiempo de compilación, tamaño de la imagen, mantenibilidad, seguridad y repetibilidad en las imagenes que se crean.

La calidad de las imagenes pueden marcan una diferencia que puede llegar a ser crítica para posteriores etapas en el desarrollo de aplicaciones como son la Orquestacion de contenedores CITA.

2.5. Soluciones Actualmente Disponibles

En la actualidad hay disponibles varias soluciones para disminuir la complejidad implicita que reside en en la configuracion de dichos ficheros dockerfile

2.5.1. Docker init

Docker init es el comando de Docker Desktop CITA que permite inicializar un proyecto con un archivo de configuración básico. su funcionamiento consta de elegir una plantilla de proyecto y seleccionar entre las opciones de configuracion.

Las opciones de configuracion se limitan a elegir el lenguaje de programacion, puerto en el que despliega la aplicacion y que comando se ejecutara al iniciar el contenedor.

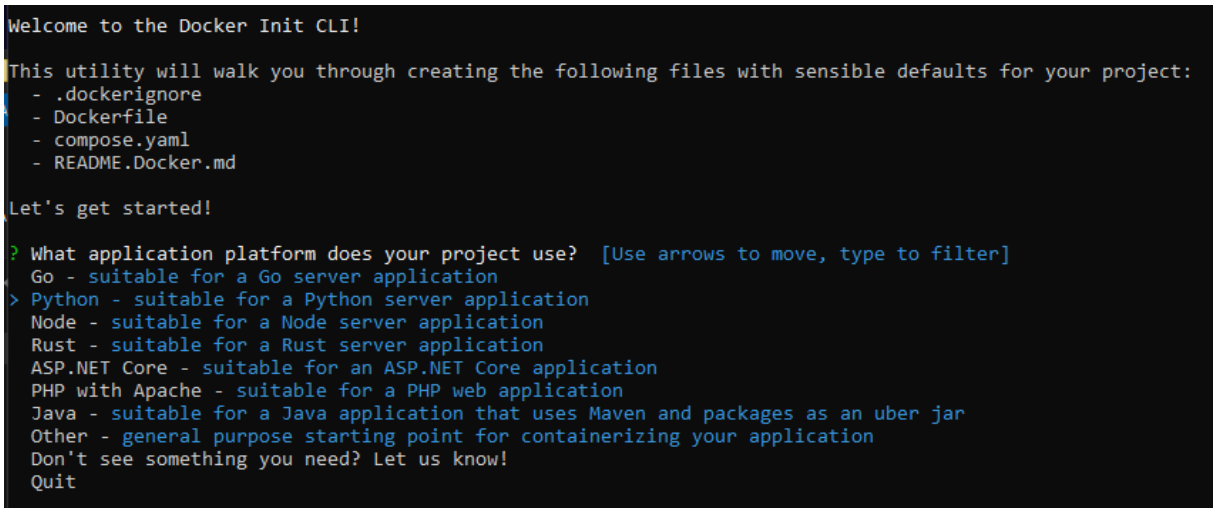

Figura 1: Captura de las opciones de Plantillas disponibles Docker Init.

2.5.2. Docker Extension for Visual Studio Code

La extension de Docker para visual estudio code CITA permite entre sus funcionalidades la generacion de los ficheros dockerfile y dockerignore. su comportamineto es similar al de Docker init, permitiendo entre otras opciones elegir el lenguaje de desarrollo del proyecto y que comando se ejecutara al iniciar el contenedor.

2.5.3. Inteligencia Artificial

En la actualidad existen soluciones que se apoyan en la inteligacia Artificial para la generacion de archivos de configuracion, y para la creacion de los fichero dockerfile. La calidad de estas soluciones es variable y depende de la calidad de los modelos de lenguaje natural empleados y de la calidad del promp de entrada. como https://magickpen.com/templates/128/ CITA o alternativas con propositito más general como chatgpt CITA

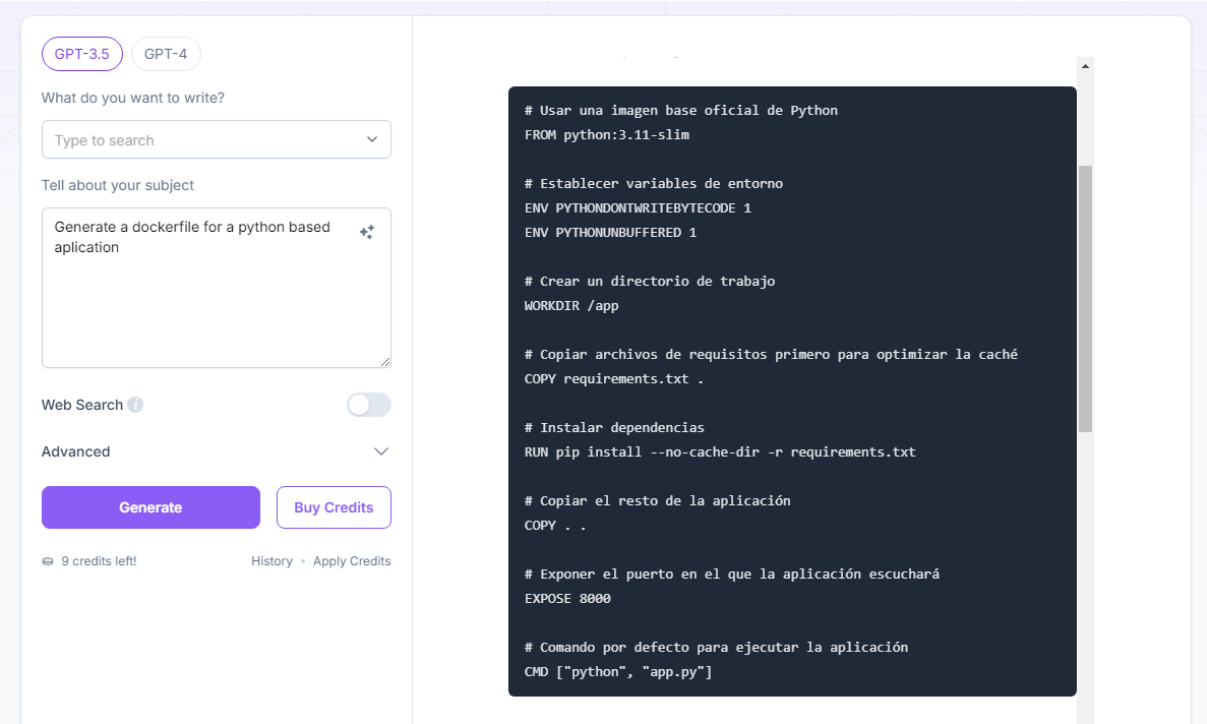

Figura 2: Captura de Magickpen.

2.5.4. Starter

Starter es una herremienta open source CITA https://www.startwithdocker.com/ capaces de reconocer el lenguaje de programacion del proyecto y generar un archivo de configuracion. La herramienta es capaz de forma autonoma reconocer el contenido de un proyecto y generar los ficheros dockerfile y dockercompose

2.6. Conclusiones

En este estudio se han recogido algunas de las principales herramientas y tecnologías disponibles en la actualidad para la generación de archivos de configuración. Todas las herraminetas hacen uso de plantillas la generacion de los ficheros dockerfile y dockerignore, estas plantillas si bien son correctas son por naturaleza limitadas y no permiten la personalizacion más final de las opciones de configuracion. No obstante suponen un punto de partida para el desarrollo de aplicaiones

2.7. Que se aporta en este contexto:

Las opciones actualemtne disponibles para la generacion de archivos de configuracion para el despliegue de servicios en Docker son limitadas y no permiten la personalizacion de las opciones de configuracion. Se deja de lado la variabilidad y las posibles opciones que existen a la hora de construir las imagenes, el proyecto proporcinara una nueva forma de construir plantillas dotando capacidad al usuario de una personalizacion más profunda y detallada de las opciones de configuracion de los servicios en Docker.

3

Tecnologías Empleadas

Este apartado describe las tecnologías y herramientas empleadas más relevantes para el desarrollo del proyecto, así como su función y su relación con el objetivo del proyecto. Para el interes del lector

3.1. Universal Variability Language UVL

CITA La Universal Variability Language (UVL) es un lenguaje de modelado diseñado específicamente para representar y manejar la variabilidad en sistemas software. La variabilidad en software se refiere a las partes de un sistema que pueden personalizarse, configurarse o cambiarse para adaptarse a diferentes necesidades o contextos. UVL se usa comúnmente en el contexto de ingeniería de líneas de productos de software (Software Product Line Engineering o SPLE), donde se trabaja con una familia de productos que comparten un núcleo común, pero pueden tener variaciones.

3.1.1. Feature Models

CITA El principal artefacto para modelar la variabilidad en SPL son los modelos de variabilidad. Existen muchos modelos de variabilidad pero los más extendidos y usados en la práctica son los modelos de características (feature models). Un feature model (FM) es un modelo para representar la variabilidad de una SPL en base a características comunes y variables (véase Figura 1), especificando qué características se pueden seleccionar en una configuración para generar un producto válido de la SPL

3.2. Docker

Con el objetivo de desarrollar una aplicaicon con animo de ejecutarse en contendores se ha optado por la tecnologia de Docker.

3.2.1. Docker Desktop

Para el manejo de los contenedores en el entorno de desarrollo se ha optado por la utilizacion de Docker Desktop.

3.3. Node

CITA para el desarrollo de la aplicacion web se ha optado por la utilizacion de Node como entorno de ejecucion de JavaScript.

3.4. Angular

CITA Para el desarrollo de la aplicacion web se ha optado por la utilizacion de Angular como framework de desarrollo de aplicaciones web.

3.4.1. Angular Material

CITA CITA Para el diseño de la interfaz grafica de la aplicacion web se ha optado por la utilizacion de Angular Material como libreria de componentes.

3.4.2. Monaco Editor

CITA Para la edicion de los archivos de configuracion se ha optado por la utilizacion de Monaco Editor como editor de codigo.

3.4.3. ngx-json-viewer

CITA CITA Para la visualizacion de los archivos de configuracion se ha optado por la utilizacion de ngx-json-viewer como visor de archivos json.

3.5. Nginx

CITA Para el despliegue de la aplicacion web se ha optado por la utilizacion de Nginx como servidor web.

3.5.1. Reverse Proxy

CITA Para el despliegue de la aplicacion y requerimientos adicionales se ha empleado Nginx como un proxy inverso.

3.5.2. Web Server

CITA Para el despliegue de la aplicacion web se ha optado por la utilizacion de Nginx como servidor web.

3.6. Python

Para el desarrollo de los servicios de backend se ha optado por la utilizacion de Python como lenguaje de programacion.

3.6.1. Flask

Para el desarrollo de los servicios de backend se ha optado por la utilizacion de Flask como framework de desarrollo de aplicaciones web.

3.6.2. guinicorn

Para el despliegue de los servicios de backend se ha optado por la utilizacion de guinicorn como servidor web.

3.6.3. Pypi

Para la gestion de dependencias se ha optado por la utilizacion de Pypi como repositorio de paquetes de Python.

3.7. Jinja

3.7.1. Jinja2

3.8. Git

Para el control de versiones se ha optado por la utilizacion de Git como sistema de control de versiones.

3.9. GitHub

Para el almacenamiento de codigo se ha optado por la utilizacion de GitHub como plataforma de desarrollo colaborativo.

3.9.1. GitHub Public Api

Para la gestion de los repositorios de GitHub se ha optado por la utilizacion de la API publica de GitHub. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

3.10. Visual Studio Code

Para el desarrollo de la aplicacion se ha optado por la utilizacion de Visual Studio Code como entorno de desarrollo integrado.

3.10.1. UVLS extension

Para la edicion de los archivos de configuracion se ha optado por la utilizacion de la extension de UVLS para Visual Studio Code. ademas de las siguientes extensiones: EXTENSIONES VISUAL ESTUDIO -UVLS https://github.com/Universal-Variability-Language/uvl-lsp -Docker -Flama https://marketplace.visualstudio.com/items?itemName=diversolab.flamapy -Graphiz Inetractive Preview https://marketplace.visualstudio.com/items?itemName=tintinweb.graphviz-interactive-preview -Latex Worshop -Pylance -Python -Python Debugger -Better Jinja

4

Metodologia del trabajo. (Fases) Implementación:

4.1. Introduction

El enfoque se ha dado en desarrollo de este trabajo ha consistido en combinar un trabajo previo de analisis e investigacion con un desarrollo iterativo y agil. La investigacion y analisis previo teniea el objetivo de conocer en profundidad las tecnologias y herramientas empleadas para porstesiromete estudicar una posible aplicacion de estas ideas y conceptos en el desarrollo de las plantillas. una vez resuelta esta primera parte se ha procedido a una toma de requisitos y a la planificacion del desarrollo de la aplicacion. para la implementacion de la aplicacion se ha optado por la utilizacion de una metodologia de desarrollo agil que permita la adaptacion a los cambios y la evolucion del proyecto.

4.2. Metodología del Desarrollo

Al encontrarnos ante un trabajo individual toda la responsabilidad y capacidad del desarrollo recae sobre una persona. Para el desarrollo de código optaremos por una metodología Scrum simplificada. De forma iterativa se ejecutará un Sprint con unos objetivos de desarrollo y planificación muy limitados. Con el objetivo de dar cabida a la complejidad del proyecto, con animo de encajar nuevas ideas y revisiones que pudieran sucederse. Tras cada Sprint se evaluará la progresión y dirección del proyecto con el tutor.

4.2.1. Justificación

El desarrollo de las plantillas requiere un estudio con el fin de limitar las opciones de configuración y centrarse en los casos de uso más comunes y en las configuraciones más utilizadas en la práctica. De entrada aportar una solucion tecnica optima a un problema es extremadamente improbable, por lo que se opta por una metodologia de desarrollo agil que permita la adaptacion a los cambios y la evolucion del proyecto.

4.2.2. Procedimiento

Tras cada Sprint se evaluará la progresión y dirección del proyecto con el tutor.

4.2.3. Limitaciones

El desarrollo de un proyecto de estas características requiere de una planificación y una metodología de trabajo adecuada. En este sentido, como se han identificado anteriormente una serie de limitaciones y restricciones que pueden afectar al desarrollo del proyecto. para el caso el desarrollo de las plantillas tiene que verse limitado en cuanto a formato y contenido, ya que no se puede abarcar todas las posibles opciones de configuracion.

4.2.4. Fases de Estudio

4.3. Estudio del Estado del Arte :

En esta fase se realiza una revisión exhaustiva de la literatura y tecnologías existentes relacionadas con el tema del proyecto. El objetivo es comprender el panorama actual, identificar tendencias, mejores prácticas y tecnologías relevantes que puedan influir en el desarrollo del proyecto.

4.4. Investigación opciones de configuración para Docker

Esta fase implica investigar y evaluar las diferentes opciones de configuración disponibles en Docker para el desarrollo y despliegue de aplicaciones. Se exploran las características, herramientas y técnicas que pueden optimizar el uso de Docker en el proyecto [2].

4.5. Desarrollo Modelos plantillas dockerfile y dockerignore ...

Esta fase implica la construcción el árbol de características y las restricciones textuales que orienten el espacio de configuraciones posibles del fichero Dockerfile

4.6. Fase desarollo Scrum

4.6.1. Requisitios de la aplicación

Aquí se definen y documentan los requisitos funcionales y no funcionales de la aplicación. Se recopilan y analizan las necesidades de los posibles usuarios, los objetivos del proyecto y las restricciones técnicas para establecer una base sólida para el desarrollo.

4.6.2. Desarrollo del la aplicación de uvengine

En esta fase se desarrolla la aplicación de uvengine, que permite la generación automática de archivos de configuración

4.6.3. Desarrollo de la web

En esta fase se desarrolla la aplicación web que permite la interacción con el usuario y la visualización de los archivos de configuración generados.

4.6.4. Desarrollo de los servicios de backend

En esta fase se desarrollan los servicios de backend que permiten la comunicación entre la aplicación web y la aplicación de uvengine y los repositorios.

4.6.5. Elaboración de la guía de uso

En esta fase se elabora una guía de uso que explica cómo utilizar la aplicación y las funcionalidades disponibles. Se elabora una guía detallada que describe cómo utilizar la aplicación, incluyendo instrucciones paso a paso, capturas de pantalla y ejemplos.

4.6.6. Elaboración de un manual de instalación

En esta fase se elabora un manual de instalación que explica cómo instalar y configurar la aplicación en un entorno local.

4.7. Elaboración de la memoria

En esta fase se redacta la memoria o informe final del proyecto, que documenta todo el proceso de desarrollo, desde la planificación hasta la implementación y las lecciones aprendidas. La memoria incluirá el análisis de los resultados, conclusiones y recomendaciones para trabajos futuros.

5

Arquitectura y Modelado del sistema

5.1. Vision general de la arquitectura

5.2. Arquitectura de backend

5.3. Arquitectura de Frontend

6
Extensibilidad

6.1. Plantillas

6.1.1. Plantillas de Dockerfile

6.1.2. Plantillas de Dockerignore

7

Despliegue de la aplicación

7.1. Despliegue local

7.2. Despliegue en la nube

8
Resultados

8.1. Resultados Obtenidos

8.2. Viabilidad del proyecto

Estamos limitados por la generacion de los fiechors de configuracion que nos proporciona la extension uvls

8.3. Escalabilidad

9

Conclusiones y líneas Futuras

9.1. Conclusion Personal,

9.2. Conclusiones del Desarrollo del proyecto

9.3. Contribuciones / Aplicaciones Prácticas

9.4. Desarrollo de nuevas plantillas

9.5. Continuación en el desarrollo de plantillas ya existentes

9.6. Mejorar la experiencia del usuario

Apéndice A
Manual de Instalación

A.1. Requisitos Previos

A.2. Descargar el codigo fuente

A.3. Configuracion del archivo Dockercomose

A.3.1. Nginx reverse proxy

A.3.2. frontend

A.3.3. Repository Manager

A.3.4. UV Engine resolver

A.4. Construccion de los contenedores

A.4.1. Github API

A.5. Terminar el servicio

Apéndice B
Manual de Uso

B.1. Inicio de la aplicacion

B.2. Barra de Navegación

B.2.1. About Page

B.2.2. Help Page

B.3. Seleccionar plantilla

B.3.1. Seleccionar version de la platilla

B.3.2. Cargar configuracion

B.3.3. Descargar Fichero Generado

B.3.4. Copiar Fichero Generado

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga

E.T.S. DE INGENIERÍA INFORMÁTICA