Documento de gestión de la integración continua

decide-part-chiquito

Grupo 2
Curso escolar: 2023/2024
Evolución y gestión de la configuración

Integrantes

Juan Jesús Campos Garrido	decide-part-chiquito-1
Alejandro Campano Galán	decide-part-chiquito-1
Antonio Carretero Díaz	decide-part-chiquito-1
Pablo Mera Gómez	decide-part-chiquito-1
Carlos Baquero Villena	decide-part-chiquito-1
David Cortabarra Romero	decide-part-chiquito-1
Alejandro Pérez Vázquez	decide-part-chiquito-2
Juan Carlos Ramírez López	decide-part-chiquito-2
Carmen Ruiz Porcel	decide-part-chiquito-2
Alejandro Santiago Félix	decide-part-chiquito-2
Sergio Santiago Sánchez	decide-part-chiquito-2
María Vico Martín	decide-part-chiquito-2

PROCESO DE INTEGRACIÓN CONTINUA (CI)

La integración continua es una práctica de desarrollo de software mediante la cual los desarrolladores combinan los cambios en el código en un repositorio central de forma periódica, tras lo cual se ejecutan versiones y pruebas automáticas.

El proceso de integración continua utilizado en decide-part-chiquito está configurado con GitHub Actions. A continuación, se describe el ciclo de integración continua y las herramientas utilizadas en el proyecto.

El proceso de integración continua utilizado en el proyecto comienza con la operación de checkout del código fuente, seguido de la configuración de la versión de Python especificada y la instalación de las dependencias del proyecto, junto con la configuración de las variables de entorno necesarias. A continuación, se ejecutan las migraciones de la base de datos. Posteriormente, se lleva a cabo la ejecución de pruebas del proyecto, generando un informe de cobertura en formato XML, el cual se envía a Codacy para su análisis. Finalmente, se realiza un análisis de calidad del código utilizando Pylint, con un enfoque específico para Django. Este enfoque de integración continua permite mantener la calidad del código y detectar posibles problemas de manera temprana en el ciclo de desarrollo.

En este proyecto, se han utilizado varias herramientas para facilitar el proceso de integración continua. GitHub Actions ha sido fundamental para la automatización del flujo de trabajo, permitiendo la ejecución de tareas específicas en respuesta a eventos en el repositorio. Python ha sido la piedra angular para la ejecución del código y las pruebas que se han utilizado son unitarias, de modelo y vistas estáticas con las herramientas que proporciona Django y para las páginas dinámicas y de carga dependencias externas como Selenium y Locust , mientras que PostgreSQL ha servido como la base de datos de servicio. Además, se ha empleado Pylint para realizar un análisis estático del código Python, lo que contribuye a mantener altos estándares de calidad. Por último, Codacy ha desempeñado un papel crucial al proporcionar un reporte detallado de cobertura y análisis de calidad del código, lo que ayuda a identificar posibles áreas de mejora en el desarrollo del proyecto.

Si se ha realizado de forma satisfactoria la integración continua se podrá realizar un despliegue si las condiciones son las correctas.

DESPLIEGUE CON INTEGRACIÓN CONTINUA (CI)

1. Crear una Cuenta en Render y Configurar el Repositorio

- **Registro en Render**: Acceder a Render y crear una cuenta, registrándose con nuestro correo electrónico o mediante la cuenta de GitHub.
- **Vinculación del Repositorio**: En el panel de Render, buscar la opción para conectar tu repositorio lo que permitirá configurar despliegues automáticos desde ese repositorio.

2. Preparar tu Proyecto Django para el Despliegue

- **Archivo requirements.txt**: Asegurar que el proyecto Django tenga el archivo requirements.txt en la raíz del repositorio y dicho archivo lista todas las dependencias de Python que el proyecto necesita.
- **Configuración de la Base de Datos**: Revisar archivo settings.py para asegurarte de que la configuración de la base de datos esté preparada para la producción. Utiliza variables de entorno para las credenciales de la base de datos.
- **Variables de Entorno**: Configurar variables de entorno para cualquier configuración sensible, como SECRET_KEY y DEBUG. Esto se hace por seguridad y para facilitar los cambios sin necesidad de modificar el código.

3. Crear un Nuevo Servicio Web en Render

- **Crear Servicio**: En tu panel de Render, elige "Crear un nuevo servicio web".
- **Selección del Repositorio**: Selecciona el repositorio que contiene tu proyecto Django.
- **Configuración Automática**: Render intentará detectar que es una aplicación Python y sugerirá configuraciones.

4. Configurar el Proceso de Construcción y Despliegue

- **Comando de Inicio**: Define el comando para iniciar tu aplicación, como “gunicorn myproject.wsgi:application”. Este comando se ejecutará para iniciar tu servidor Django.
- **Variables de Entorno**: En la configuración de tu servicio en Render, establece las variables de entorno necesarias.

5. Configurar la Base de Datos

- **Agregar Base de Datos**: En Render, selecciona "Agregar Base de Datos" y sigue los pasos para configurar nuestra base de datos de PostgreSQL.
- **Conexión a Django**: Render proporcionará una URL de base de datos. Utiliza esta URL en tu settings.py de Django para conectar tu aplicación con la base de datos.

6. Despliegue Automático

- **Configuración de Despliegues**: En la configuración de tu servicio en Render, elegimos el despliegue automático de la aplicación al subir los cambios a la rama “master”.

7. Pruebas y Despliegue Inicial

- **Despliegue Manual o Automático**: Iniciar un despliegue manualmente desde Render o realizar un cambio en la master para desencadenar un despliegue automático.