Universidad de Sevilla

Escuela Técnica Superior de Ingeniería Informática

ACTA FUNDACIONAL

Grado en Ingeniería Informática – Ingeniería del Software

Evolución y Gestión de la Configuración

EGC2425-RaboDeToro-hub-1

Curso 2024 – 2025

17-10-2024

Índice

1. Resumen ejecutivo

En este documento se tratan la definición de los distintos procesos del proyecto

2. Contenido
a. Integración continua

Para la integración continua, el proyecto utiliza **Codacy** y **Render**, que permiten automatizar la calidad y el despliegue del código.

1. **Análisis de calidad con Codacy**: Codacy se encarga de revisar el código en cada cambio para asegurar que cumpla con los estándares de calidad, detectar errores, problemas de estilo y vulnerabilidades de seguridad. Este análisis estático permite que los desarrolladores identifiquen y corrijan problemas antes de que impacten en el funcionamiento general de la aplicación.
2. **Despliegue automatizado en Render**: Una vez que el código ha pasado las verificaciones de Codacy, se despliega automáticamente en Render. Render actúa como la plataforma de despliegue donde la aplicación queda disponible para pruebas adicionales o para su uso en un entorno accesible.

Este proceso de CI garantiza que el proyecto mantenga una alta calidad de código y que las actualizaciones estén disponibles en un entorno seguro y controlado, facilitando la colaboración y la detección temprana de problemas antes de que el código se integre en la rama principal.

b. Construcción

El proyecto RaboDeToroHub1 tiene un proceso de construcción el cual garantiza la calidad y la estabilidad del código antes de su despliegue. A continuación, se describen los detalles clave de este proceso:

1. **Configuración del Entorno y Dependencias**:

El proyecto utiliza herramientas de construcción que aseguran que todas las dependencias necesarias se instalen y configuren en el entorno de desarrollo y producción. Al ser un proyecto de Python, el archivo requirements.txt o configuraciones equivalentes permiten la instalación automática de librerías necesarias para la aplicación, facilitando la configuración en distintos entornos.

2. **Análisis de Calidad y Linting**:

Antes de proceder con la construcción, se ejecutan herramientas de análisis de calidad de código (como Codacy, ya integrado en este proyecto) para detectar errores y problemas de estilo en el código. Codacy realiza una revisión estática que ayuda a los desarrolladores a identificar y corregir problemas antes de que el código se compile o se ejecute.

3. **Compilación y Preparación del Código**:

Aunque en proyectos de Python no se realiza una compilación estricta, se asegura que el código esté en un estado listo para ejecutarse sin errores. En esta etapa, también se validan módulos y se configuran scripts de inicio que establecen los parámetros necesarios para que la aplicación funcione correctamente.

4. **Ejecución de Pruebas Automáticas**:

El proyecto incluye pruebas automáticas que se ejecutan después de la construcción para validar que los componentes críticos del sistema funcionen según lo esperado. Estas pruebas pueden incluir pruebas unitarias y de integración. Al ejecutar las pruebas de manera automatizada, se garantiza que cualquier error en el código se detecte y se pueda corregir antes del despliegue.

5. **Integración Continua y Verificación en Render**:

Después de completar las pruebas, la integración continua a través de Render permite un despliegue automatizado. Render asegura que la última versión del proyecto se implemente en un entorno accesible, donde se pueden realizar pruebas adicionales en un entorno real.

6. **Empaquetado y Despliegue Final**:

Una vez superadas las pruebas y la revisión de calidad, el código se prepara para el despliegue. Render gestiona este proceso de empaquetado y despliegue, de modo que la aplicación esté lista y accesible para los usuarios o para su integración en entornos de producción.

c. Cambios

La gestión de cambios se organiza cuidadosamente mediante el uso de ramas, pull requests (PRs) y otras prácticas de control de versiones. Esto permite mantener un flujo de trabajo ordenado y facilita la colaboración entre los desarrolladores, asegurando que el código se integre de manera segura y controlada. A continuación, se describe cómo se implementa este proceso:

Gestión de Cambios en el Proyecto

1. **Ramas**:

El proyecto utiliza ramas para organizar y aislar las diferentes etapas de desarrollo. Las ramas más comunes incluyen:

- **main**: La rama principal, que contiene la versión estable del proyecto y que no debe modificarse directamente.
- **develop**: Esta rama suele ser la base para el desarrollo y la integración de nuevas características antes de que se fusionen en main.
- **feature/***: Ramas creadas para desarrollar nuevas funcionalidades. Cada nueva característica o mejora se desarrolla en una rama feature, lo cual permite a los desarrolladores trabajar en paralelo sin afectar la estabilidad de main o develop.
- **fix/***: Ramas destinadas a resolver errores específicos. Cada corrección se trabaja en una rama fix, asegurando que los cambios necesarios se apliquen de forma aislada y controlada.

También se pueden generar en casos especiales otras ramas las cuales tengan una utilidad relevante a la hora de implementar o definir algún elemento o concepto dentro de la aplicación o para facilitar la gestión del cambio.

Esta estructura de ramas facilita la organización y el seguimiento de cada cambio, permitiendo que el equipo desarrolle nuevas características o corrija errores sin interferir con el trabajo de los demás.

2. **Pull Requests (PRs)**:

Cada cambio importante o nueva funcionalidad en el proyecto se introduce mediante un pull request. Un PR permite que los desarrolladores propongan cambios en una rama y soliciten una revisión antes de fusionarlos con develop o main.

Los PRs son una parte fundamental del proceso de revisión de código, ya que permiten que otros miembros del equipo revisen el código, hagan sugerencias y detecten posibles errores o mejoras antes de que el cambio se integre en una rama principal.

El uso de PRs ayuda a asegurar la calidad y coherencia del código en el proyecto. Cada PR suele incluir una descripción detallada de los cambios y, en algunos casos, pruebas que demuestren que la nueva funcionalidad o corrección funciona como se espera.

3. **Revisión y Aprobación de Cambios**:

Antes de fusionar un PR en develop o main, al menos uno de los miembros del equipo debe revisar y aprobar los cambios. Esta revisión puede incluir pruebas de funcionamiento, verificación de estilo y consistencia, así como asegurar que los cambios cumplen con los estándares del proyecto.

Los revisores pueden hacer comentarios en el PR, sugiriendo ajustes o cambios adicionales. Una vez que el autor del PR aborda estos comentarios, el PR puede ser aprobado y fusionado en la rama de destino.

4. **Integración de Cambios en Ramas Principales**:

Después de que los PRs han sido aprobados y probados en develop, los cambios se pueden fusionar en main cuando el equipo esté listo para lanzar una nueva versión estable del proyecto. Este proceso asegura que solo el código completamente probado y revisado llegue a la versión estable.

5. **Automatización y Control de Calidad**:

Con la integración de Codacy y Render, cada cambio propuesto en un PR pasa

automáticamente por análisis de calidad y pruebas antes de ser revisado por un miembro del equipo. Esto ayuda a detectar problemas de calidad, errores o vulnerabilidades en una etapa temprana del ciclo de desarrollo.

Ventajas de este Enfoque

Este enfoque estructurado permite al equipo mantener un flujo de trabajo eficiente y seguro, reduciendo el riesgo de errores y facilitando la cooperación entre desarrolladores. Al utilizar ramas, pull requests y revisiones de código, el proyecto se beneficia de un control riguroso de cambios, manteniendo alta la calidad del código y permitiendo que cada nueva funcionalidad o corrección se integre de manera organizada y controlada.

d. Código

La gestión y organización del código son esenciales para asegurar su calidad, mantenibilidad y escalabilidad. A continuación, se detallan las prácticas y estándares de código usados en este repositorio.

Organización del Código

1. **Estructura de Archivos y Directorios**:

El proyecto está organizado en directorios que separan claramente los diferentes componentes y módulos de la aplicación. Esta estructura facilita la navegación y permite a los desarrolladores encontrar rápidamente el código relevante para cualquier tarea.

Los directorios suelen incluir:

- app: Contiene el núcleo de la aplicación, con la lógica principal del sistema.
- core: Incluye archivos fundamentales y configuraciones esenciales para el funcionamiento del proyecto.
- docs: Documentación adicional que proporciona instrucciones y detalles técnicos para que los desarrolladores comprendan el código y el proyecto en general.
- migrations: Archivos de migración de base de datos, en caso de que el proyecto requiera persistencia de datos, facilitando la actualización y modificación de la estructura de datos.

2. **Estándares de código**:

El proyecto sigue un estándar de codificación para mantener la consistencia y calidad del código. Este estándar cubre aspectos como el formato, el uso de nombres de variables y funciones, y la organización de módulos y clases.

Codacy, una herramienta de análisis de calidad de código está integrada en el repositorio y ayuda a asegurar que el código cumpla con las mejores prácticas y estándares de calidad. Codacy realiza análisis automáticos para detectar problemas de estilo, vulnerabilidades de seguridad y posibles errores lógicos.

3. **Commits Estandarizados**:

El repositorio sigue un patrón de commits estandarizado para facilitar el seguimiento de los cambios. Cada commit se realiza con un mensaje claro y descriptivo, que ayuda a otros miembros del equipo a entender qué modificaciones se realizaron y por qué.

Las especificaciones de los mensajes de commits vienen determinadas en el documento de acta de constitución

4. **Buenas Prácticas**:

Cada módulo y función incluye comentarios y documentación interna que explican su propósito y funcionamiento. Esto es fundamental para el mantenimiento del proyecto, ya que permite a otros desarrolladores entender rápidamente el propósito de cada fragmento de código.

e. Despliegue

Para el despliegue del proyecto es importante seguir un enfoque que facilite la integración continua (CI) y el despliegue continuo (CD) para mantener el flujo de desarrollo estable y actualizado. Aquí algunos puntos clave para el despliegue de este tipo de proyecto:

1. **Configuración de CI/CD**:
Utilizar herramientas de CI/CD como GitHub Actions, Travis CI o Jenkins para automatizar las pruebas y despliegues. Configura un archivo de flujo de trabajo en `.github/workflows` para definir los pasos de integración y despliegue.

2. **Contenerización con Docker**:
Implementar un `Dockerfile` y, si es necesario, un `docker-compose.yml` para definir los servicios y sus dependencias. Esto facilita la creación de entornos de desarrollo y producción replicables. Verifica que las dependencias estén correctamente instaladas y configuradas.

3. **Gestión de Dependencias**:
Define y actualiza las dependencias en un archivo como `requirements.txt` (Python) o un archivo específico según el lenguaje. Esto asegura que las dependencias sean las mismas en cada despliegue.

4. **Estrategias de Despliegue**:
Dependiendo de los requisitos, puedes optar por despliegues en entornos de prueba (staging) antes del despliegue en producción. Servicios como AWS, Google Cloud, o Heroku pueden facilitar este proceso. Define las variables de entorno necesarias para la conexión con bases de datos u otros servicios externos en el entorno de despliegue.

5. **Monitoreo y Logs**:
Implementar soluciones de monitoreo para rastrear el rendimiento y errores del sistema en tiempo real. Herramientas como Prometheus o servicios de logging pueden integrarse para mejorar la visibilidad del sistema.

Si ya existe alguna configuración en el repositorio, como archivos de Docker, scripts de despliegue o configuraciones de CI/CD, adaptar esta guía general a esos detalles específicos permitirá ajustar el despliegue de manera efectiva. ¿Hay alguna configuración ya preestablecida en este proyecto que te gustaría que revisáramos en más detalle?