Acta Fundacional del Proyecto

Grupo: 2

Tutor: David Ramos

Repositorio: https://github.com/Lidiajim/montaito-hub.git

Tipo: Single

Integrantes:

Nombre	Apellidos	Correo Corporativo
Óscar	Menéndez Márquez	oscmenmar@alum.us.es
Jun	Yao	junyao@alum.us.es
Lidia	Jiménez Soriano	lidjimsor@alum.us.es
Álvaro	Ruiz Gutiérrez	alvruigut@alum.us.es
Carlos	Martín de Prado Barragán	carmarbar9@alum.us.es
Francisco	Capote García	fracapgar@alum.us.es

ÍNDICE

1. Descripción del Proyecto

El proyecto consiste en un fork o una evolución de la aplicación UVLHub, en el cual se desarrollarán nuevas características para mejorar su funcionalidad. El objetivo principal será integrar los flujos de trabajo (workflows) actuales, configurarlos adecuadamente para ajustarse a las nuevas necesidades y diseñar otros workflows que complementen o extiendan los ya existentes.

Además, se explorará la posibilidad de utilizar herramientas de integración y entrega continua (CI/CD) que no han sido implementadas previamente en el sistema. Esto permitirá optimizar los procesos de despliegue y asegurar una mejor automatización en el ciclo de desarrollo. A lo largo del proyecto, se priorizará la innovación y la eficiencia, garantizando que los nuevos workflows y herramientas CI/CD contribuyan a un desarrollo más ágil y robusto de la aplicación.

2.Tabla de versiones

Fecha	Versión	Descripción de los cambios	Sprint
16/10/2024	1.0	Creación del documento	1
18/10/2024	1.1	Desarrollo del documento	1

3. Hitos

ID	Descripción	Fecha de Entrega
M0	Inscripción de los equipos	04/10/2024
M1	Sistema funcionando y pruebas	23/10/2024
M2	Sistema funcionando y con incrementos	13/11/2024
M3	Entrega de proyectos y defensas	18/12/2024

4. Políticas de Mensajes de Commit

Los mensajes de commit deberán ser descriptivos y concisos, incluyendo información relevante sobre los cambios realizados. Se siguen las siguientes convenciones:

- Uso de verbos en modo imperativo en el título.
- Prefijar los mensajes de commit con uno de los siguientes tipos:
  - feature: para las tareas que hemos hecho individualmente.
  - fix: para correcciones de errores.
  - docs: para cambios en la documentación.
  - test: para cuando se añadan tests.
- No usar puntos finales ni suspensivos al final del título.
- El mensaje de commit no deberá contener ningún mensaje de espacios en blanco.
- Ser corto y conciso en los títulos, para ello se establecen un máximo de 50 caracteres.
- Uso de mayúsculas al inicio del título y por cada párrafo del cuerpo de

5. Políticas de Ramas

El proyecto tendrá tantas ramas como issues creadas. Cada rama estará dedicada al desarrollo de la tarea correspondiente a la issue asignada. Estas son algunas pautas clave a seguir en cuanto a las políticas de ramas:

- **Nomenclatura de las ramas**: Las ramas deben nombrarse de manera consistente y descriptiva. Se sugiere un formato que incluya el número del issue y una breve descripción de la tarea, por ejemplo: `feature/#45-mejora-login`.
- **Ramas principales**: Se mantendrán dos ramas principales:
  - `main`: Esta rama contendrá siempre el código en producción, estable y sin errores.
  - `develop`: Esta rama se utilizará para integrar el trabajo de las ramas de características antes de pasarlo a `main`. Es la versión candidata para la siguiente liberación.
- **Ramas de características (feature branches)**: Para cada nueva funcionalidad o cambio, se creará una rama a partir de `develop`, y al finalizar el trabajo, se realizará una fusión a `develop` mediante un pull request. Ejemplo de nombre de rama: `feature/#45-mejora-login`.
- **Ramas de corrección de errores (bugfix branches)**: Cuando se identifiquen errores en el código, se crearán ramas específicas para su corrección. Estas ramas pueden derivarse de `main` (si es un error en producción) o de `develop` (si es un error en desarrollo). Ejemplo de nombre de rama: `bugfix/#53-correccion-bug-login`.
- **Ramas de hotfix**: Si es necesario corregir un error crítico en producción, se creará una rama de hotfix directamente desde `main` y, una vez corregido, se fusionará tanto en `main` como en `develop`. Ejemplo de nombre de rama: `hotfix/#101-parche-seguridad`.

6. Políticas de Issues

Se creará una issue para cada WI, entregable o problema que surja durante el proyecto.

Con esto crearemos ramas independientes con el nombre de estas tareas, marcándola inicialmente como issues abiertas.

A cada issue se le asignará una etiqueta o label que identificará el problema a abordar:

- Feature
- Documentation
- Test
- Bug
- Fix

Además, se asignarán uno o varios miembros a cada issue, con la finalidad de tener un control sobre la carga equitativa de trabajo y sobre el desarrollo de cada tarea, marcando en todo momento el estado en el que se encuentra.