Camyo

Grupo 5

Gestión de Código

Devising a Project (#DP)

Miembros del grupo:

- José Ramón Baños Botón
- Isabel X. Cantero Corchero
- Sheng Chen
- Carlos García Martínez
- Carlos García Ortiz
- Raúl Heras Pérez
- Pedro Jiménez Guerrero
- Claudia Meana Iturri
- Rubén Pérez Garrido
- Lucía Pérez Gutiérrez
- Francisco Pérez Manzano
- Diego José Pérez Vargas
- María C. Rodríguez Millán
- Sonia María Rus Morales
- Adriana Vento Conesa
- Jun Yao

Ingeniería del Software y Práctica Profesional • Curso 2024-25

Contenido

1. Alojamiento el Código

1.1 Alojamiento del Código

Repositorio: Link del Repositorio. Este documento está disponible en la wiki del Repositorio.

Plataforma: GitHub, utilizado por ambos equipos para una mejor cohesión.

1.2 Proceso de Integración y Despliegue Continuo

1. **Repositorio**: El código se almacenará en un repositorio remotos.
2. **Automatización de builds**: Los builds se automatizarán con herramientas de Git.
3. **Frecuencia de commits**: Los commits se realizarán con frecuencia para asegurar la integración continua.
4. **Pruebas**: Las pruebas se realizarán en un entorno de preproducción antes del despliegue.

1.3 Gestión de Tareas

Herramientas: Se utilizarán **Issues** y **Projects** de GitHub para gestionar las tareas.

Organización: Cada equipo tendrá su propio *Project* para organizar y asignar responsables. Las tareas que requieran colaboración se registrarán en Microsoft Planner dónde se gestionan las tareas del proyecto completo.

2. Política de Ramas

2.1 Enfoque Alternativo a GitFlow

El equipo ha decidido crear una rama para cada tarea o incidencia. Las ramas seguirán los siguientes patrones:

- Tareas:

```
<frontend/backend>/tarea_<número_de_tarea>/<nombre_de_tarea>
```

- Incidencias:

```
<frontend/backend>/incidencia_<número_de_tarea>/<nombre_de_tarea>
```

Entre las excepciones, se incluye la rama "main", que funcionará como la rama principal destinada al despliegue, la rama "build", que se empleará para realizar cambios específicos en la configuración del entorno.

2.2 Cierre de tareas:

El encargado de una tarea creará sus propias *pull requests*. Él mismo, o en su defecto, los coordinadores, asignarán a un revisor entre los miembros del proyecto,

para revisar cualitativamente el contenido de la misma y el cumplimiento con los requisitos.

Si el revisor aprueba la *pull request*, éste se encargará en el momento de hacer *merge*. Si no la aprueba, el revisor dejará abierta la *pull request*, y escribirá sus comentarios en ella. Se notificará al encargado de la tarea para que revise los errores. Una vez supuestamente arreglados, los cambios aparecerán en la misma *pull request*, y se volverá a empezar este ciclo.

Una vez se le ha hecho *merge* a una tarea y se ha cerrado, el encargado de la tarea para la cual se ha creado la rama se encargará de borrarla.

3. Política de Commits

El proyecto utilizará Conventional Commits, una especificación que tiene como objetivo hacer que los mensajes de commit sean legibles tanto para humanos como para máquinas. La información correspondiente a este apartado se ha extraído de la página oficial. Por lo tanto, los commits deben seguir la siguiente estructura, y deberán escribirse en inglés mayoritariamente:

```
<type>(<scope>): <subject>
<BLANK LINE>
<body>
<BLANK LINE>
<footer>
```

- **<type>** se refiere a uno de los varios tipos preestablecidos de operaciones que se pueden realizar en el proyecto. Este atributo es **requerido**. Los tipos recomendados y su uso son:
  - **feat:** Nueva funcionalidad.
  - **fix:** Corrección de errores.
  - **docs:** Cambios solo en la documentación.
  - **style:** Cambios que no afectan el significado del código (espacios en blanco, formato, punto y coma faltante, etc.).
  - **refactor:** Un cambio de código que no corrige un error ni agrega una funcionalidad.
  - **perf**: Un cambio de código que mejora el rendimiento.
  - **test:** Modificaciones en el banco de pruebas.
  - **chore:** Cambios en el proceso de construcción o en herramientas auxiliares y bibliotecas, como la generación de documentación.
  - **revert:** Reversión de uno o más commits.
- **<scope>** es cualquier cosa que especifique el lugar del cambio del commit. Este atributo es opcional. Por ejemplo, api, lang, o owner.

- **<subject>** contiene el mensaje del commit, o en otras palabras, la descripción corta del cambio realizado. Este atributo es **requerido**. Se usará minúscula al comienzo, y no se usará punto al final. Empezaremos por un verbo en imperativo, a poder ser.
- **<body>** contiene una explicación más detallada de la motivación del cambio y/o cómo este contrasta con el código anterior. Este atributo es opcional.
- **<footer>** contiene cualquier información extra, como cambios importantes en la API o referencias a problemas de GitHub o commits. Incluirá el ID de la issue (si procede), para poder relacionar el commit con ella.

Se recomienda el uso de la extensión Conventional Commits de Visual Studio Code para evitar errores en el formato de los commits.

Ejemplo:

```
feat(user-profile): add profile picture in backend


It adds a new feature that allows users to upload prophile pictures by themselves.

This commit includes the backend implementation. The frontend one will be made soon.


#12
```

4. Política de Pull Request

Por motivos de claridad y unificación, el equipo ha definido un formato para el nombramiento de las *pull requests*. No es un formato fijo, sino que se basa en los siguientes principios:

- No se usará el nombre por defecto que provee GitHub, el cual parsea el nombre de la rama de origen.
- No se usará el formato de Conventional Commits (es decir, el texto por defecto del commit).
- Se utilizará la herramienta PR Agent
- Será una descripción breve y sin adornos del objetivo de la *pull request*. Se ha de asemejar en parte al título de la *issue* a la que corresponde, si procede.

4.1 Versionado del producto:

El proyecto seguirá las directrices de versionado semántico. La información correspondiente a este apartado se ha extraído de la página oficial. Por lo tanto, el versionado funcionará de la siguiente manera:

- **Versión mayor**: Este número se incrementa cuando se realizan cambios significativos e incompatibles en la API del proyecto. Indica que la nueva versión puede no ser compatible con las versiones anteriores y requiere una cuidadosa consideración durante las actualizaciones. (**Ejemplo:** 2.0.0 -> 3.0.0)
- **Versión menor**: Incrementar este número significa la adición de nuevas funcionalidades al proyecto de manera compatible con versiones anteriores. Los usuarios pueden esperar que las características existentes se mantengan intactas, lo que garantiza transiciones suaves entre versiones. (**Ejemplo:** 2.0.0 -> 2.2.0)
- **Parche**: Los incrementos en la versión de parche están reservados para correcciones de errores que son compatibles con versiones anteriores. Estos cambios abordan problemas sin introducir nuevas funcionalidades o romper la funcionalidad existente, proporcionando a los usuarios una experiencia fluida durante las actualizaciones. (**Ejemplo:** 2.0.0 -> 2.0.2)

5. Gestión de Tareas

5.1 Asignación de actividades

Al inicio de cada sprint, el coordinador dividirá los diferentes temas a tratar en tareas específicas, las cuales serán asignadas a los miembros del equipo. Se buscará equilibrar la carga de trabajo en términos de tiempo y esfuerzo, aprovechando al máximo las habilidades e intereses individuales.

5.2 Tipos de Issues

Una *issue* representará una solicitud de cambio en el sistema. Los tipos de *issues* que se gestionarán son:

- **Actividades:** creadas por el equipo durante la planificación para cumplir con los objetivos.
    - **Tarea:** actividad principal necesaria para completar un objetivo.
    - **Incidencias:** reportadas por el equipo o usuarios para notificar problemas en el funcionamiento del sistema.

Las actividades de documentación, como la actualización del 'Diario del equipo' no se representará mediante issues.

A las issues generadas automáticamente por bots no se les aplicarán estas políticas.

5.3 Roles en Issues

Cada *issue* será asignada a un único miembro del equipo, quien será responsable de su ejecución. En caso de que se necesite involucrar a más personas, se creará una *issue* adicional para repartir el trabajo.

5.4 Priorización de Issues

Se utilizará la siguiente clasificación para definir la prioridad de las *issues*:

- **Alta:** Tareas críticas para el progreso del proyecto. Estas tareas afectan directamente el avance, ya que otras tareas dependen exclusivamente de su finalización.
- **Media:** Tareas que tienen una doble función: dependen de otras tareas para empezar o completarse y, al mismo tiempo, otras tareas dependen de ellas. Son importantes, pero no bloquean el progreso de manera inmediata.
- **Baja:** Tareas independientes que no tienen ninguna otra tarea que dependa de ellas. Su ejecución no afecta directamente el avance del proyecto.

5.5 Estados de Issue

Las *issues* pasarán por los siguientes estados. En GitHub Projects, estos se representarán mediante columnas.

Estado	Descripción
TODO	Tareas que aún no han comenzado.
In Progress	Tareas que estén en proceso.
Failed Pull Request	Tareas cuya *pull request* no ha sido aprobada.
Done	Tareas que han sido aprobadas o completadas.

Si una pull request no se acepta, e trasladará a la columna del estado *Failed Pull Request*. Una vez que las correcciones sean realizadas, pasará al estado *Done*.

5.6 Etiquetas de Issues

Las siguientes etiquetas se utilizarán para clasificar los cambios en las *issues*:

- Tipos de actividad: task, incidence.
- Equipo: frontend, Backend
- Tipo de cambio: feat, fix, refactor, style, test, database, meeting, build, deployment y hotfix

5.7 Nomenclatura de Issues

Para mantener un orden coherente, los títulos de las **issues** seguirán los siguientes patrones :

```
Tarea<Frontend/Backend><número_tarea>-<número_subtarea_(opt)>: <Nombre_tarea>.
```

Ejemplos:

```
Tarea Frontend 13-1: Cambiar diseño de botones.
```

```
Tarea Backend 15: Realizar tests de login.
```

El cuerpo seguirá la siguiente estructura:

```
Description:
> - <primer punto>
> - <segundo punto>
>  [...]
Issues Relacionadas:
> #<número_de_issue>
```

Un ejemplo de cómo se vería:

Descripción:

- Notify the user, onscreen, that an email has been sent.

- Wait for the code to be sent and send it one it is entered.

Issues Relacionadas:

#5

El símbolo ">" se parseará en el markdown de la descripción de las issues en GitHub tal y como se ve en este comentario.

Se realizarán a través de una plantilla de tareas en GitHub.

5.8 Estandarización de incidencias

Para garantizar que las incidencias reportadas contienen la información necesaria para su resolución, se seguirá esta plantilla:

```
   Título

   <título de la incidencia>

 Descripción

    <descripción>

Pasos para reproducir

   1. <primer paso>

   2. <segundo paso>

Resultado esperado

   <resultado esperado>

Resultado obtenido

   <resultado obtenido>

Evidencias (capturas, logs, etc.)

   <evidencias>

Entorno

   - Sistema operativo: <sistema operativo>

   - Versión del producto: <versión del producto>

   - Navegador:

      - Modelo: <modelo de navegador>

      - Versión: <versión del navegador>

   - Detalles adicionales: <detalles adicionales>

Posibles soluciones:

   <ideas sobre cómo solucionar el problema>

Percepción de prioridad o severidad (crítica, alta, media o baja):

   <percepción sobre la prioridad o severidad de la incidencia>

Comentarios Adicionales

   <comentarios adicionales>
```

Se hará a través de una plantilla de incidencias en GitHub.

Un ejemplo de cómo se vería:

Título

Fix selenium tests

Descripción

Some of the selenium tests were very flaky, we need them to be more consistent.

Pasos para reproducir

1. run python -m pytest

Resultado esperado

All tests pass

Resultado obtenido

Some of them fail

Evidencias (capturas, logs, etc.)

N/A

Entorno

- Sistema operativo: Ubuntu 24.04
- Versión del producto: N/A
- Modelo de navegador: Google Chrome
- Versión del navegador: 131.0.6778.108
- Detalles adicionales: ChromeDriver 131.0.6778.85

Posibles soluciones

Fix them

Percepción de prioridad o severidad (crítica, alta, media o baja)

Alta

Comentarios adicionales

N/A