ACUERDO DE COMPROMISO

Índice

Definición de Hecho

- Desarrollo terminado
    - Completada la funcionalidad.
    - Code review aprobado.
    - Doc Swagger para endpoints.
- Testing
    - Tests unitarios con un mínimo de 70% de cobertura.
    - Pruebas postman para los endpoints.
    - Los test unitarios serán realizados por la persona que desarrolle la funcionalidad.

Política de Commits

Para mantener un estilo similar entre los commits usaremos “conventional commits”, siendo el cuerpo de estos **siempre en ESPAÑOL**. La estructura seguida es la siguiente:

<tipo>: <descripción breve>
[opcional] Cuerpo detallado del mensaje
[opcional] Pie de mensaje

El <tipo> especifica la naturaleza del cambio realizado. Los tipos más comunes incluyen:
- feat: Una nueva funcionalidad.
- fix: Corrección de errores.
- docs: Cambios en la documentación (no relacionados con el código).
- style: Cambios que no afectan la lógica del código (formato, espacios en blanco, etc.).
- refactor: Cambios en el código que no corrigen errores ni añaden funcionalidades.
- test: Adición o modificación de pruebas.

La descripción debe ser concisa y clara, expresando en una sola línea el del commit. Se recomienda utilizar el modo imperativo (por ejemplo, "añadir...”, "arreglar...”) y evitar el uso de mayúsculas al inicio, salvo para nombres propios.

El cuerpo se utiliza para detallar el contexto del cambio, explicar el por qué detrás del commit, y describir cualquier implicación importante.

El pie del mensaje puede incluir:
- Referencias a tickets o tareas: Refs #123.

Ejemplo de commit:
****no añadir lo que pone en cursiva, son los apartados****

Título: feat: añadir soporte para la autenticación por token

Cuerpo: Este cambio introduce un nuevo sistema de autenticación basado en tokens JWT. El módulo de usuario ha sido actualizado, y se ha añadido una nueva dependencia.

Pie de mensaje: Refs #123

Política de Ramas

Se hará uso de GitFlow.

- feature/<funcionalidad-que-se-implementa>: el nombre de las funcionalidades se hará en español y con guiones.
- develop
- hotfix
- main (Solo se suben versiones finales)

Desarrollo de ramas que añadan funcionalidad:

1. Crear una rama feature/ desde develop.
2. Desarrollar la funcionalidad en la rama creada.
3. Hacer Pull Request solicitando la revisión de un/unos revisor/es cualquiera del equipo antes de fusionar con develop.

Política de Código

- Uso de camelCase
- Toda la codificación se realizará en inglés, es decir, los nombres de clases, de variables, de atributos y de los métodos.
- Nada de *autowireds* con Spring, usar el constructor.
- Recomendación: inyección de dependencias con maps o lists para el patrón strategy
- Uso de la etiqueta Qualifier.
- Cuando sea necesario, utilizar el patrón Strategy.

Política de Pull Request

- Hacer comentarios constructivos al código de los compañeros.
- Cualquier miembro o miembros del equipo puede revisar el código de otro compañero.
- Al menos una revisión es necesaria para mergear la PR.
- Uno mismo no puede aceptar sus propios pull request.
- Nombre de las Pull Request (Merge: Rama que se quiere mergear into rama a mergear).
- Solo se puede hacer Pull Request cuando se considere que la tarea cumple con todo lo mencionado en Definición de Hecho.

Arquitectura

- Clásica por capas.

Metodología

La metodología a seguir va a ser Scrum.

- Se realizarán Sprints de 3 días.
- Utilizaremos Github para repartirnos las tareas.