BORROO

PLAN DE GESTIÓN DE LA DOCUMENTACIÓN Y CÓDIGO

Devising a project (DP) – G4

Repositorio: https://github.com/ISPP-2425-G4/

20/02/2025

Miembros:

- David Blanco Mora
- Pablo Díaz Ordóñez
- Pablo Espinosa Naranjo
- Jesús Fernández Rodríguez
- Francisco Fernández Mota
- Javier García Rodríguez
- Miguel González Ortiz
- Álvaro Martín Muñoz
- Ignacio Naredo Bernardos
- Javier Nieto Vicioso
- Marco Padilla Gómez
- Miguel Palomo García
- Luis Javier Periáñez Franco
- Alexander Picón Garrote
- Santiago Rosado Raya
- Julia Sánchez Márquez
- Alejandro Sevillano Barea

Índice

GESTIÓN DE LA DOCUMENTACIÓN

HERRAMIENTAS PARA LA GESTIÓN DE LA DOCUMENTACIÓN

Para la gestión de la documentación se utilizarán las siguientes herramientas:

- **Gestor de documentos en la nube** (OneDrive) para almacenar y compartir documentos.
- **Sistema de control de versiones** (GitHub) para documentación técnica y código fuente.
- **Plataformas de comunicación** (Discord, WhatsApp) para la notificación de actualizaciones y discusión de documentación.
- **Plataforma de base de conocimiento** (Docusaurus) para almacenar toda la documentación acerca del grupo y del proyecto.

CLASIFICACIÓN DE DOCUMENTOS

Los documentos del proyecto se clasificarán en las siguientes categorías, en las que encontraremos principalmente los siguientes tipos de documentos:

1. Planes de Gestión

- Plan de gestión de la calidad
- Plan de gestión de comunicaciones
- Plan de gestión de riesgos
- Plan de gestión de la documentación y código

2. Actas de Reunión

- Resumen de decisiones tomadas
- Acciones y responsables
- Fecha y participantes

3. Documentación de Feedback

- Encuestas a usuarios
- Reportes de pruebas
- Retroalimentación del equipo

4. Backlog

- Requisitos y funcionalidades
- Historias de usuario
- Priorización de tareas
- Planificación del desarrollo

5. Acuerdos y protocolos

- Protocolos de conflictos
- Commitment Agreement

RESPONSABILIDADES

En cada subgrupo de trabajo, está presente una persona con el rol de secretario, quien se encarga de generar las actas de reuniones internas de dichos subgrupos. Este mismo encargado será quien lleve el control de la documentación que genere y/o repercuta a su subgrupo.

ALMACENAMIENTO Y CONTROL DE VERSIONES

- Se establecerán permisos de acceso para garantizar seguridad y control de cambios.
- Se emplearán convenciones de nomenclatura para facilitar la organización.
- Se mantendrán versiones históricas para rastrear modificaciones y recuperar información cuando sea necesario

REVISIÓN Y ACTUALIZACIÓN

- La documentación será revisada periódicamente según el avance del proyecto.
- Se definirán intervalos regulares para la actualización de documentos críticos.
- Los cambios importantes serán comunicados a todo el equipo mediante las herramientas de comunicación establecidas.

GESTIÓN DEL CÓDIGO

HERRAMIENTAS PARA LA GESTIÓN DEL CÓDIGO

A lo largo del desarrollo del proyecto, se usarán varias herramientas que darán apoyo a la gestión del código.

- Para alojar los repositorios compartidos se usará **GitHub**, así como el sistema de control de versiones Git.
- Se usará **GitHub Actions** como plataforma de integración y despliegue continuos (CI/CD).
- Para la gestión de tareas se usará **GitHub Project** y poder monitorizar el trabajo del equipo.
- Como editor de código, **Visual Studio Code** es la herramienta estándar que el equipo de desarrollo usará.
- Para la implementación del backend se usará el framework **Django**, escrito en Python.
- Para la implementación del frontend se usará **React**, biblioteca de JavaScript.
- La herramienta para los despliegues será **Docker**.
- Para la monitorización del tiempo se usará la herramienta **Clockify**.
- La Base de Datos se desarrollará den **MariaDB**.

POLÍTICA DE VERSIONADO

- Las versiones seguirán el formato "vX.Y.Z", siendo X, Y y Z tres números naturales y el cambio de uno de estos números supondría una nueva versión.
- El valor X se incrementará en 1 cuando la aplicación presenta una nueva funcionalidad de gran importancia o un conjunto de estas que hagan de la aplicación un gran cambio.
- El valor Y se incrementará en 1 cuando la aplicación presente nuevas funcionalidades que no son fundamentales o cambios menores.
- El valor Z se incrementará en 1 cuando la aplicación presente correcciones puntuales, parches o hotfixes.
- El cambio en uno de los números haría que los posteriores tomen el valor 0.
- Ejemplos de versiones son: v0.2.1, v1.0.0, v1.2.1, v2.0.4.

POLÍTICA DE COMMITS

En este apartado, se detallan una serie de características que deben de estar presentes en todos los commits del repositorio, con vistas a que sean homogéneos y con una fácil y rápida lectura e interpretación.

NORMAS DE COMMITS

- **Commits atómicos:** cada commit realizado debe representar un cambio atómico y significativo, facilitando así la revisión del código e identificación de posibles errores.
- **Mensajes Descriptivos:** los mensajes han de ser descriptivos y claros, dando una explicación de qué cambios se realización y su por qué, ayudando así a entender el propósito de dichos cambios de forma intuitiva.
- **Frecuencia regular:** se anima a realizar commits de forma regular y frecuente, en lugar de acumular muchos cambios en uno solo, para reducir así el riesgo de conflictos.
- **Revisión de Código:** paso previo a hacer un commit a la rama principal, se debe llevar a cabo una revisión del código por parte de uno o varios compañeros del equipo que no haya participado en dicho commit, para mantener la calidad deseada en el código e identificar posibles problemas o errores de forma previa a la fusión de todos los cambios.
- **Integración Continua:** utilizar herramientas de integración continua con vistas a automatizar pruebas y despliegues. Los commits han de pasar estas pruebas de manera previa a su fusión.
- **Resolución de Conflictos:** si se dieran conflictos al fusionar, se deben de resolver de manera efectiva en el menor tiempo posible, comunicándose en todo momento con el equipo.

Así, definimos una plantilla de commits de la siguiente forma:

tipo: asunto
Donde:

tipo:

- **feat**: nueva funcionalidad que se añade.
- **fix**: corrección de bugs encontrados.
- **refactor**: refactorización de código.
- **docs**: añadir o actualizar documentación.
- **test**: incorporación o modificación de pruebas.
- **conf**: modificación de archivos de configuración.

asunto:
Breve descripción de lo que se ha tratado, comenzando por un verbo en participio.

Así, un ejemplo de ello puede ser:

docs: Añadido Plan de Gestión Del Código.pdf

POLÍTICA DE BRANCHING

Se empleará **GitFlow** como estrategia de versionado para la gestión de ramas y releases. Esta decisión viene dada por las ventajas que puede brindar a nuestra aplicación, además de la familiaridad que nuestro grupo tiene con este sistema. Se

definirán las siguientes ramas principales:

- **main**: es la rama principal y estable. Debe de contener únicamente el código listo para ser desplegado en producción. Proporciona una línea base estable para la versión más actual del proyecto, garantizando que la versión que se despliegue sea funcional y confiable.
- **develop**: rama donde se integran todas las características completadas, además de ser donde se realiza la mayor parte del desarrollo. Ofrece un entorno centralizado para la integración continua y colaboración de todo el equipo. Permite desarrollar pruebas de integración de forma más temprana y mantener una evolución consistente en temas de código.
- **feature/*:** por cada nueva característica de la aplicación, se desarrollará una nueva rama de feature a partir de develop. Permite un desarrollo en paralelo de distintas características sin producir interferencias entre ellas. También facilita la revisión del código, además de la realización de pruebas de esa funcionalidad de manera previa a su integración con el resto del código.
- **hotfix/*:** por cada problema critico que surja, se crea una rama de este tipo a partir de develop, para poder llevar a cabo una solución inmediata. Una vez corregido, se fusiona tanto en main como en develop. Así, permite abordar problemas urgentes en producción de forma rápida y controlada.
- **release/*:** cuando se prepara una nueva versión de lanzamiento de la aplicación, se crea una rama de este tipo desde develop, realizándose aquí los últimos preparativos previos a dicho lanzamiento. Proporciona un entorno controlado para el lanzamiento, permitiendo corregir errores y realizar pruebas finales en el último momento sin interferir en el resto del desarrollo.

POLÍTICA DE ISSUES

Las **issues** aportan información al proyecto acerca del desarrollo necesario para dar con el producto final. El **título** de la issue mostrará de forma breve el cambio a implementar, en caso de requerir información adicional, esta estará en la **descripción** de la issue.

- Tendrán un estado asociado, este puede ser To do, In Progress, In Review o Done que determinan el punto de desarrollo en el que se encuentra la issue.
    - **To do**: La issue no ha comenzado su fase de desarrollo.
    - **In Progress**: La issue está en desarrollo.
    - **In Review**: La issue ha finalizado su fase de desarrollo y se encuentra en revisión (Pull Request).
    - **Done**: La issue se ha revisado e implementado correctamente.
- Tendrán una etiqueta que proporcione información adicional acerca del tipo de issue. Estas etiquetas pueden ser **Feature, Documentation, Test** o **Incident**.
- En caso de incidencia, es necesario que la issue proporcione toda la información necesaria para su resolución

POLÍTICA DE PULL REQUEST

Cada cambio en el código que aporte nuevas funcionalidades o cambios significantes, se somete a un proceso de revisión por pares antes de integrarse en las ramas principales. El proceso incluye los siguientes pasos:

1. Solicitar revisión a un miembro del equipo.
2. El revisor realiza comentarios o sugiere cambios.
3. El autor implementa las modificaciones solicitadas.
4. Si el revisor aprueba, la Pull Request se fusiona con la rama correspondiente.

FORMATO DE PULL REQUEST

El **título** de la Pull Request vendrá dado por el **mismo nombre** que tenga la **issue asociada** a ella, evitando así variaciones que puedan dar a confusiones.

Si la pull request necesitara algún tipo de información adicional que aclare ciertos temas sobre lo que incluye la pull request, se desarrollará el campo de **descripción** indicando la información que se preste oportuna.

NOTA: Si se tratara de un cambio menor, que no repercute en cambios de funcionalidades o del comportamiento de la aplicación, puede no necesitarse la participación de otro miembro del grupo en el proceso de Pull Request, en búsqueda de agilizar todo el proceso para algo tan menor, siempre y cuando se notifique al resto del grupo sobre ello.

Este sistema asegura que cada cambio sea revisado desde múltiples perspectivas, lo que reduce errores y mejora la calidad general del proyecto, gracias a la revisión obligatoria de al menos un miembro del equipo y la resolución de conflictos previa a la fusión de los cambios.

POLÍTICAS DE CÓDIGO

Atendiendo al proceso de realización del código, se establecen unas pautas a seguir con el fin de dar claridad y eficiencia:

- **Nombres claros y descriptivos:** se requieren de nombres significativos y que describan su uso para funciones, variables y clases. Se recomienda evitar abreviaturas complejas o nombres que puedan crear confusión, así como números mágicos.
- **Funciones y métodos pequeños y específicos:** se busca crear funciones cortas que se centren en una tarea lo más específica posible, dentro de lo que cabe, en búsqueda de una buena legibilidad del código y entendimiento. Si se desarrollan funciones demasiado largas, se valorará su división.
- **Comentarios útiles y concisos:** usar comentarios únicamente donde sea necesario para aclarar algo, evitando líneas de texto que no aporten nada.
- **Evitar código duplicado:** mantener lo más mínimo de código repetido. Para ello, se realizarán funciones, clases y/o módulos reutilizables.
- **Refactorización constante:** búsqueda constante de mantener el código lo más limpio posible. Para ello, se valorará las posibles refactorizaciones de código que nos permitan llegar a ese objetivo.
- **Consistencia en el estilo de codificación**: Se deben seguir convenciones de estilo establecidas para mantener coherencia en todo el código.
- **Modularidad y reutilización**: Diseñar el código en módulos independientes y reutilizables para facilitar su mantenimiento y evitar la repetición innecesaria.
- **Seguridad en el código**: No exponer información sensible (como credenciales o claves API) en el código fuente. Validar y sanitizar cualquier entrada de usuario para prevenir ataques como inyección de código.
- **Pruebas y validaciones**: Gran parte del código debe estar acompañado de pruebas automatizadas (unitarias y/o de integración) para garantizar su correcto funcionamiento antes de ser desplegado.

ESTÁNDARES DE FORMATEO Y ESTILO

- Toda documentación va a ser realizada con herramientas como Word o ficheros markdown presentando una estructura común y organizada y respetando el estilo de títulos, tablas y otros componentes presentes.
- Se hará uso de buenas prácticas para la indentación, comentarios y nombrado de variables y funciones.

INTEGRACIÓN Y DESPLEIGUE CONTINUO

- Se usarán herramientas como GitHub Actions.
- Para el despliegue se hará uso de Docker.
- Se realizarán scripts sobre la automatización de pruebas para que den paso al despliegue en caso de pasar correctamente.
- Se realizarán scripts sobre el despliegue para realizarlo de forma automática o facilitar el proceso, así como informar la gestión de dependencias.
- Se automatizarán las pruebas de diversos tipos y se usarán herramientas de análisis de código y reportes de cobertura de código si es posible.

ORGANIZACIÓN DEL REPOSITORIO EN GITHUB

El código y la información del proyecto estarán centralizados en **dos** repositorios pertenecientes a la organización **ISPP-2425-G4**. En el caso del **repositorio del código de la aplicación**, contaremos con la siguiente estructura:

- **Código fuente**: Desarrollo de la aplicación siguiendo estándares de nomenclatura, estilo y versionado.
- **Documentación del proyecto**: Visión, objetivos, requisitos y casos de uso.
- **Gestión del código fuente**: Estándares, flujos de trabajo y guías de desarrollo.

- **Planificación**: Roadmap del proyecto, backlog y asignación de tareas.
- **Decisiones tecnológicas y arquitectónicas**: Diagramas UML, diagramas de capas, estructura de la base de datos.

En el caso del **repositorio de la base del conocimiento**, encontraremos:

- **Código fuente**: Desarrollo de la base del conocimiento siguiendo estándares de nomenclatura, estilo y versionado.
- **Documentación de la base del conocimiento:** Acuerdos, protocolos, actas, feedback, planes y todo aquel documento que aporte información sobre la creación y desarrollo de la aplicación.