

## ACTA FUNDACIONAL

<https://github.com/berenjena-hub/berenjena-hub>

EGC



# Índice:

1. Nombre del Proyecto.....	2
2. Integrantes del Equipo.....	2
3. Hitos.....	2
4. Work Items Asignados a los Integrantes.....	3
5. Política de Commits.....	3
6. Política de Issues.....	4
7. Política de Ramas (Branches).....	4
8. Política de Pull Requests (PR).....	5
9. Política de Sanciones.....	5
10. Gestión del cambio.....	6
11. Gestión del código fuente.....	7
12. Política de despliegue.....	8
13. Ciclo de integración continua.....	8
14. Casos de prueba.....	9
15. Estrategias de gestión de equipos.....	9
10. Aprobación y Firma de los Integrantes.....	6

## 1. Nombre del Proyecto

- Nombre del proyecto: berenjena-hub (**UVLHub single**)
  - Fecha de inicio: 19/10/2024
- 

## 2. Integrantes del Equipo

- Luis Giraldo Santiago ([luigirsan1@alum.us.es](mailto:luigirsan1@alum.us.es))
    - Usuario de GitHub: [Luis-Giraldo-Santiago](#)
  - Sheng Chen ([sheche1@alum.us.es](mailto:sheche1@alum.us.es)):
    - Usuario de Github: [sheche1](#)
  - Francisco Fernández Mota ([frafermot@alum.us.es](mailto:frafermot@alum.us.es))
    - Usuario de GitHub: [frafermot](#)
  - David Vargas Muñoz ([davvarmun@alum.us.es](mailto:davvarmun@alum.us.es))
    - Usuario de github: [davvarmun](#)
  - Felipe Solís Agudo ([felsolagu@alum.us.es](mailto:felsolagu@alum.us.es))
    - Usuario de Github: [Fsolis19](#)
  - Enrique García Abadía ([enrgaraba@alum.us.es](mailto:enrgaraba@alum.us.es))
    - Usuario de Github: [Enrgaraba](#)
-

### 3. Hitos

- **M0: Inscripción de los equipos**
    - Fecha de entrega: 04/10/2024
  - **M1: Sistema funcionando y pruebas**
    - Fecha de entrega: 23/10/2024
  - **M2: Sistema funcionando y con incrementos**
    - Fecha de entrega: 18/12/2024
  - **M3: Entrega de proyectos y defensas**
    - Fecha de entrega: 18/12/2024
- 

### 4. Work Items Asignados a los Integrantes

- **Sheng Chen:**
  - WI: Dashboard (Medium)
- **Francisco Fernández Mota:**
  - WI: Download in different formats (Medium)
- **David Vargas Muñiz:**
  - WI: Rate datasets / models (Medium)
- **Felipe Solís Agudo:**
  - WI: Improve UI (Medium)
- **Enrique García Abadía:**
  - WI: Advanced Filtering (High)
- **Luis Giraldo Santiago:**
  - WI: Social Module(High)

#### WI adicional: Fakenodo

Los miembros asignados a cada WI serán los responsables de su implementación, y estos determinarán el estado en el que estos se encuentran, comunicandoselo al resto de miembros del grupo. Los estados serán “En progreso”, “En testeo” y “Finalizados”. El WI adicional será realizado por cualquiera de los miembros de este grupo. Se le dará prioridad a los WI en el proceso de gestión del cambio definido más adelante, a no ser que se deba corregir una incidencia. Los gestores de los WI son los miembros asignados a cada uno de ellos.

---

### 5. Política de Commits

Los mensajes de cada commit deben ser claros y concisos, siguiendo la estructura:

Tipo(WI): [Breve descripción del cambio] #Nº de Issue

- **Tipos de commits:**

- **feat**: Nuevas características
  - **fix**: Corrección de errores
  - **style**: Formateo de código, espacios, comas, etc.
  - **refactor**: Refactorización del código
  - **test**: Agregar o modificar tests
- 
- WI indica el ítem de trabajo con el que se está trabajando. Se utilizarán los nombres de los WI definidos en el punto 4 de este documento.
  - En el caso de la documentación, la forma de estructurar los commit será:  
**Tipo(docs): [Breve descripción del cambio] #Nº de Issue**
  - Los commits deben realizarse de manera atómica, es decir, un commit debe abarcar un solo cambio significativo.
  - Todos los commits deben estar vinculados a un issue correspondiente para un mejor seguimiento.
  - Ser corto y conciso en los títulos. Se deberán realizar, como mínimo, tres commits con cambios significativos semanales por cada uno de los integrantes de este equipo.
- 

## 6. Política de Issues

Los issues serán creados para todas las tareas, bugs, y mejoras del proyecto, siguiendo la siguiente estructura:

**(WI): [Título conciso de la Issue]**

En caso de incidencia, la estructura será la siguiente:

**Problema [descripción del problema]**

En caso de la documentación, se seguirá la estructura de los WI sustituyendo el título de este por "Docs".

- **Etiquetas y Clasificación**: Usar etiquetas como "Incidencia", "Feature", "Test", "Documentación", con un sistema de colores para identificar cada tipo.
- **Formato de Descripción**: Incluir resumen, pasos para reproducir (si es un bug), comportamiento esperado y actual, y requisitos específicos.
- **Criterios de Aceptación**: Definir claramente los criterios para considerar un issue cómo completado.
- **Revisión y Cierre**: Revisar cada issue antes de cerrarlo y dejar un comentario sobre la solución implementada.
- **Priorización**: Asignar una prioridad (Muy Alta, Alta, Media, Baja, Muy Baja) a cada issue para planificar mejor el trabajo.

- **Asignar a un responsable:** Cada Issue deberá tener asignado un responsable para su resolución.
  - **Vinculación con Pull Requests:** Relacionar issues con los PR correspondientes para facilitar el seguimiento de los cambios.
  - **Actualización del Estado:** Mantener el estado actualizado (e.g., "En progreso", "En revisión") para un seguimiento claro.
- 

## 7. Política de Ramas (Branches)

- La rama principal será **main**, y se considerará estable y lista para producción.
  - Para el desarrollo de nuevas características, se utilizarán ramas con la siguiente nomenclatura:
    - **main/(WI)**, para el desarrollo de cada WI concreto, por lo que habrá 6 ramas derivadas de **main**. Estas ramas se borrarán cuando el WI esté completo.
    - **main/(Nombre-Apellido)**, para el desarrollo de funcionalidades o contenido adicional a los WI. También para la resolución de incidencias si la rama del WI a la que referencia la incidencia está eliminada. Se borrarán una vez se completen las incidencias o tareas adicionales.
    - **main/docs**, para la documentación. Se borrará cuando se encuentre toda la documentación subida.
  - Usar ramas **release/vX.Y.Z** (donde X.Y.Z corresponde a la versión) para preparar una nueva versión antes de desplegarla en producción. En estas ramas se realizan pruebas finales y ajustes menores antes de fusionarlas a **main** y etiquetar la versión.
  - Configurar protección en **main** para requerir revisiones y pasar checks automáticos.
  - Todas las ramas deben ser eliminadas una vez que el trabajo ha sido integrado en **main** y aprobado a través de un pull request.
- 

## 8. Política de Pull Requests (PR)

- Crear un PR para cualquier cambio en la rama **main**.
- Se seguirán distintos formatos de nombres para las pull requests:
  - **WI:** Se asignará como nombre de la pull request el nombre establecido del WI cuando se quiera integrar a **main** cambios respectivos a dicho WI.
  - **Nombre del integrante del grupo:** Se asignará como nombre de la pull request el nombre de uno de los miembros del grupo cuando se deba realizar una pull request a **main** para realizar otro tipo de funcionalidades, como por ejemplo la modificación de los workflows.

- **fix(Nombre del integrante del grupo): Breve descripción del error:** Se asignará este nombre a la pull request cuando se detecte un posible fallo o error registrado para corregirlo y subirlo a **main**.
  - Requiere la revisión de al menos un integrante antes de aprobar aquel PR dirigido a **main**, que afecte a más de un módulo o aquel cambio que haya surgido por la colaboración de más de un integrante.
  - Las pull requests se realizarán a la rama **main** cuando haya los suficientes cambios significativos.
  - Incluir en cada PR: descripción del cambio.
- 

## 9. Política de Sanciones

- **Incumplimiento de las políticas de commits:**
    - Primera vez: Advertencia verbal.
    - Segunda vez: Revisión obligatoria del código antes de cada commit durante una semana.
    - Tercera vez: Reasignación de tareas y reporte al líder del proyecto.
  - **Incumplimiento de la política de issues:**
    - Primera vez: Reasignación del issue a otro integrante.
    - Segunda vez: Reducción de la asignación de nuevos issues durante una semana.
    - Tercera vez: Revisión y evaluación del desempeño del integrante.
  - **Incumplimiento de la política de PRs:**
    - Primera vez: Suspensión de la posibilidad de hacer merges durante dos días.
    - Segunda vez: Revisión obligatoria de todos los PRs por un miembro del equipo durante dos semanas.
    - Tercera vez: Evaluación del rol del integrante en el proyecto.
- 

## 10. Gestión del cambio

Esta es la política de gestión de cambios e incidencias que se seguirá en el desarrollo de este proyecto

### 1. Identificación del Cambio

- **Revisión inicial:** Cualquier cambio debe estar documentado en una nueva **Issue** (realizada como se expone en el punto 6 de este documento). La persona que propone el cambio debe:

- Explicar el propósito del cambio en la descripción de dicha Issue (mayor detalle de explicación si la Issue corresponde a una incidencia).

## 2. Clasificación y Priorización

- **Evaluar impacto:** La persona asignada a dicha Issue determinará el alcance del cambio.
- **Priorizar:** Se priorizará en función del grado de importancia que tenga dicha Issue.

## 3. Planificación

- **Definición de tareas:**
  - Divide el cambio en tareas más pequeñas si es necesario.

## 4. Implementación

- **Creación de una rama específica si es necesaria:**
  - Se creará una rama con el nombre del miembro del grupo asignado a dicha Issue si es necesario para su resolución debido a la magnitud del cambio. En caso de un cambio menor, dicha solución se implementará en una de las ramas del WI afectado por dicho cambio.
- **Desarrollo en la rama:**
  - Se implementa el cambio siguiendo las directrices de este documento
- **Comprobación de solución:**
  - Dicha persona se asegura de que se ha solventado la Issue correctamente

## 5. Revisión del Cambio

- **Pull Request (PR):**
  - Una vez que el cambio esté implementado, abre un PR hacia la rama main si se hizo una rama específica para su resolución.
- **Revisión de código:**
  - Al menos un miembro del equipo debe revisar el PR.

## 6. Aprobación y Fusión

- **Criterios de aprobación:**
  - El PR debe ser aprobado por una persona antes de fusionarse.

## 7. Seguimiento y Cierre

- **Validación:**
  - Confirma que el cambio funciona como se espera en el entorno en vivo.

## 11. Gestión del código fuente:

Respecto a la gestión del código fuente, se seguirán los siguientes estándares:

- **Servicio de alojamiento y tecnología de control de repositorios:** Se utilizará GitHub como sistema de almacenamiento para el repositorio del proyecto y Git como sistema de control de repositorios.
- **Archivos de Configuración:** Se utilizarán archivos de configuración estándar (`.gitignore`, `.moduleignore`) para asegurar consistencia en el desarrollo.
- **Rama Principal y Ramas Secundarias:** Se seguirá la política de ramas establecida en el punto 7.
- **Pruebas:** Se desarrollarán pruebas unitarias, de carga y de interfaz para garantizar el correcto funcionamiento de las funcionalidades del proyecto. En algunos casos no será necesario incluir pruebas de alguno de los tipos mencionados anteriormente.
- **Pull request:** Se utilizarán pull request según lo establecido en el punto 8.
- **Commits:** Se utilizarán commits según lo establecido en el punto 5.
- **Control de versiones:** Se utilizarán etiquetas para marcar versiones importantes (`v1.0.0`, `v1.1.0-beta`).
- **Integración Continua (CI):** Se configurará un sistema de CI como GitHub Actions para ejecutar pruebas automáticamente en cada PR y en cada commit.
- **Despliegue Continuo (CD):** Se automatizará despliegues a la rama `main` mediante la plataforma de despliegue render.

## 12. Política de despliegue:

Respecto a las estrategias de despliegue que se seguirá en este proyecto, se realizará un archivo `render.yml` estableciendo un workflow el cual para cada pull request aceptada en `main` se realizará un despliegue de la aplicación, al cual podremos acceder mediante un enlace (dicho enlace se irá actualizando progresivamente).

Además, al finalizar el proyecto, se realizará una `release` de la rama `main`.

## 13. Ciclo de integración continua:

El proceso de integración continua que se pretende llevar a cabo será:

- **Desarrollo de la funcionalidad/corrección de errores/ en ramas locales.**
- **Push a main**, disparando automáticamente el pipeline de CI mediante GitHub Actions.



- **Ejecución del pipeline de CI:** El pipeline realizará las siguientes acciones.
  - Comprobará la integridad de los mensajes de commit subidos a main y si siguen el formato definido en el punto 5.
  - Escanea el código en busca de vulnerabilidades o problemas de seguridad mediante un workflow de Codacy.
  - Se ejecutará otro workflow para lanzar las pruebas realizadas de manera automática.
  - Por último, se realizará automáticamente el despliegue explicado en el punto 12.
- **Revisión de Código:** Se revisará el código de las pull request hacia main por parte de al menos un miembro del grupo comprobando la calidad y claridad del código, cumplimiento de estándares, impacto en otras partes del proyecto y los resultados del pipeline de CI.
- **Integración en la Rama Principal:** Al aprobarse y fusionarse el PR, el pipeline de CI se ejecutará nuevamente para validar la estabilidad del proyecto en su conjunto.

## 14. Casos de prueba:

Para los casos de prueba, las técnicas de diseño que se utilizarán serán:

- **Basado en la Sintaxis:** Esta técnica se centra en diseñar casos de prueba utilizando la estructura y la sintaxis del código fuente de la aplicación.
- **Basado en Grafo:** En esta técnica, se modela el comportamiento del sistema a través de un grafo, identificando nodos como estados y aristas como transiciones entre estados.

Además, se contará con un workflow el cual lanzará las pruebas desarrolladas cada vez que se acepte una pull request a main.

## 15. Estrategias de gestión de equipos:

Para la gestión del equipo de trabajo de este proyecto, se llevará a cabo las siguientes estrategias:

- **Estructura democrática:** Las decisiones se tomarán en conjunto y todos los miembros tienen las mismas responsabilidades. Además, se tomarán en cuenta las opiniones de todos los miembros de este.

- **Canales de comunicación:** Se utilizará Discord como canal principal de comunicación para el equipo.
- **Reuniones regulares:** Se realizarán reuniones semanales a través del canal de comunicación mencionado anteriormente para comentar y debatir el avance del proyecto.
- **Herramientas de gestión:** Se utilizará GitHub Project como principal herramienta de gestión y priorización.
- **Resolución de disputas:** Las disputas ocasionadas por Conflictos basados en tareas, Conflictos de liderazgo, Conflictos de estilo de trabajo y/o Choques de personalidad se resolverán de manera pacífica y mediante el diálogo, en presencia del resto de miembros del grupo.

## 16. Aprobación y Firma de los Integrantes

Por la presente, los integrantes del equipo aceptan y se comprometen a seguir las políticas establecidas en este acta.

- Felipe Solís Agudo - Firma: 
- David Vargas Muñiz - Firma: 
- Sheng Chen - Firma: 
- Luis Giraldo Santiago - Firma: 
- Francisco Fernández Mota - Firma: 
- Enrique García Abadía - Firma: 