**DELU**

**Plan de Gestión de la Configuración del Software**

Integrantes

Calle Huamantinco, Luis Eduardo

Calongos Jara, Leonid

Flores Cóngora, Paolo Luis

Pariona Molina, Matthew Alexandre

Calderón Matias, Diego Alonso

Luján Vila, Frank José

Versión 1.1

**Control de versión**

| **Fecha** | **Autor** | **Descripción** |
| --- | --- | --- |
| 19/09/2024 | Matthew Pariona Molina | Estudio de la problemática de la empresa |
| 7/11/2024 | Matthew Pariona Moina | Actualización de contenido de las actividades de la SCM |
| 26/11/2024 | Matthew Pariona Molina |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Introducción

## Situación de la empresa y problemática

La empresa "DELU" ha estado trabajando en diversos proyectos durante los últimos años, mostrando un crecimiento significativo recientemente. Los proyectos que DELU posee son DeluStock, FitClue y ProfeSoft. "DeluStock" es una plataforma web diseñada para pequeñas y medianas empresas que facilita el control de productos, el registro de entradas y salidas, alertas de stock bajo y la generación de reportes. Actualmente, se encuentra en fase de mantenimiento, con algunos problemas de rendimiento identificados. La empresa está evaluando la incorporación de nuevas funcionalidades, como la integración con sistemas de pago y facturación electrónica. "FitClue" es una aplicación para dispositivos móviles enfocada en el bienestar y fitness, que ofrece planes de entrenamiento personalizados, seguimiento del progreso, recomendaciones nutricionales y clases en vivo con entrenadores. Actualmente, está en desarrollo, con el diseño de la interfaz de usuario casi finalizado. Sin embargo, la empresa debe optimizar la integración con dispositivos wearables y mejorar la experiencia del usuario. “ProfeSoft” es una plataforma web destinada a los estudiantes de la Facultad de Ingeniería de Sistemas e Informática de la UNMSM. Permite evaluar anónimamente a los profesores, calificar y dejar comentarios sobre su desempeño. Su objetivo es proporcionar información útil para que los alumnos tomen decisiones informadas al elegir a sus docentes y fomentar la mejora en la calidad de la enseñanza.

A pesar del crecimiento, la gestión de desarrollo en DELU presenta varios desafíos. El desarrollador que estuvo trabajando solo durante estos años no utilizaba ningún sistema de control de versiones, lo que significa que no hay un historial organizado de cambios, versiones o colaboraciones. Además, no existía un plan claro para la colaboración en proyectos ni un enfoque para el trabajo en equipo, lo que dificulta la transición hacia un equipo más grande. Las pruebas del software y los despliegues en producción se hacían de manera manual, lo que introduce errores y riesgos en cada entrega, y podría ser un obstáculo importante para el crecimiento futuro de la empresa y la escalabilidad de sus proyectos.

## El objetivo del plan

Para enfrentar la falta de control de versiones, la dificultad de trabajo en grupo y la gestión manual de despliegues y pruebas, “DELU” establecerá una organización en GitHub. Esta organización permitirá alojar todos los proyectos en repositorios individuales bajo un único nombre de organización. Con esta estructura, “DELU” podrá asignar roles administrativos adecuados, mejorar la colaboración entre el equipo de desarrollo y automatizar funciones y procesos clave, como el control de versiones, despliegues y pruebas, aumentando así la eficiencia y el rendimiento del equipo.

# Gestión de Configuración de Software

## Roles y responsabilidades

| Roles | Responsabilidades |
| --- | --- |
| Gestor de la configuración | Define, aplica y supervisa el proceso de gestión de versiones y cambios en el software. |
| Inspector de Aseguramiento de Calidad | Auditar la configuración y asegurar que el software cumple con los estándares de calidad. |
| Desarrollador Principal | Implementar las configuraciones y coordinar los aspectos técnicos del desarrollo de software |
| Bibliotecario | Definir y mantener las bibliotecas usadas en la gestión de configuración. Asegurarse de que los aspectos prácticos de la gestión de configuración funcionen adecuadamente entre sí. |
| Comité de Control de Cambios | Evaluar registro de eventos, gestionar solicitudes de cambios relevantes y hacer seguimiento de su ciclo de vida. Proveer apoyo y coordinar con los otros comités de control de cambios. |

## Herramientas, entorno e infraestructura

| **CRITERIOS** | **Github** | **Gitlab** | **Bitbucket** |
| --- | --- | --- | --- |
| Gratuito | Gratuito | Gratuito | Gratuito |
| Curva de aprendizaje | Moderada | Moderada | Moderada |
| Centralizado | Sí | Sí | Sí |
| Distribuido | Sí | Sí | Sí |
| Control de roles y permisos | Sí | Sí | Sí |
| Automatización | Sí | Sí | Sí |
| Integraciones | Amplio soporte con herramientas de CI/CD externas (Jenkins, CircleCI, etc.) y GitHub Actions integrado | GitLab CI/CD nativo y robusto, permite pipelines complejos sin necesidad de herramientas externas | Integración fuerte con Jira y otros productos de Atlassian |

## Herramienta elegida

Como herramienta de control de versiones utilizaremos GitHub. Si bien las tres alternativas son similares en funcionamiento y flujo de trabajo, GitHub aparece como la más popular y familiar para la mayoría de desarrolladores de hoy en día. Además, ofrece características sobresalientes cuando se trata de automatización y DevOps gracias a sus integraciones y el ecosistema que rodea GitHub actions, posicionándose como la herramienta ideal para nuestro trabajo.



# Actividades de la SCM

## Identificación de la configuración

### Definición de nomenclaturas de ítems

Item: Cronograma del proyecto

Nomenclatura: PEP-CP

Caso 1: Para ítems que pertenecen a un proyecto:

| Acrónimo del proyecto + “-”+ Acrónimo del ítem + ”.”+ extensión |
| --- |

Caso 2: Si los nombres son iguales:

| Acrónimo del proyecto + “-”+ Acrónimo del ítem + Última letra de la última palabra del nombre del ítem + ”.” + extensión |
| --- |

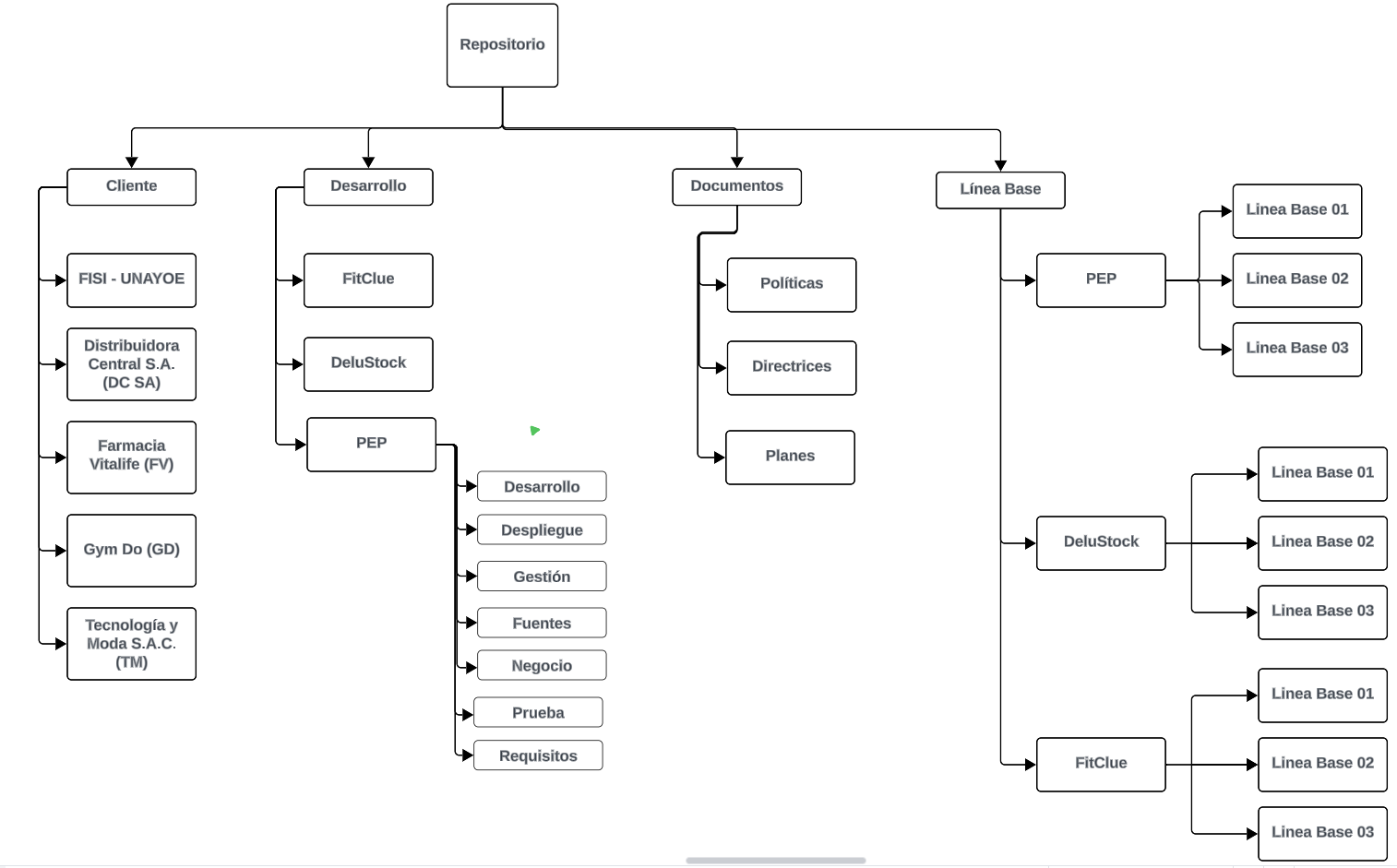
Caso 3: Si los nombres aún siguen iguales después del caso 2:

| Acrónimo del proyecto + “-”+ Acrónimo del ítem +Última letra de la última palabra del nombre del ítem + “-” + número de repetición + ”.” + extensión |
| --- |

### Clasificación de los ítems del proyecto PEP

| **Tipo**  (E= Evolución  F= Fuente  S= Soporte) | **Nombre del Item** | **Nomenclatura** | **Extensión** | **Proyecto** |
| --- | --- | --- | --- | --- |
| E | Project Charter | PEP-PC | docx | PEP |
| E | Tablero de Trello | PEP-TT | docx | PEP |
| E | Historia de Usuario 01 | PEP-HU01 | docx | PEP |
| E | Historia de Usuario 02 | PEP-HU02 | docx | PEP |
| E | Historia de Usuario 03 | PEP-HU03 | docx | PEP |
| E | Historia de Usuario 04 | PEP-HU04 | docx | PEP |
| E | Historia de Usuario 05 | PEP-HU05 | docx | PEP |
| E | Historia de Usuario 06 | PEP-HU06 | docx | PEP |
| E | Documento Final de Requisitos | PEP-DFR | docx | PEP |
| E | Informe de Control de Calidad | PEP-ICC | docx | PEP |
| E | Diseño de Diagramas UML | PEP-DUML | docx | PEP |
| E | Diseño de Diagrama de Componentes y Deployment | PEP-DDCD | docx | PEP |
| E | Diseño de Arquitectura de Gestión de Identidades y Accesos | PEP-DAGIA | docx | PEP |
| E | Diseño de Wireframe de baja fidelidad | PEP-DWBF | docx | PEP |
| E | Documentación de la API | PEP-DAPI | docx | PEP |
| E | Diseño de Mockup | PEP-DM | docx | PEP |
| E | Diseño del Prototipo de Alta fidelidad de la Interfaz de Usuario | PEP-DPAFIU | docx | PEP |
| E | Informe de Validación del Diseño | PEP-IVD | docx | PEP |
| E | Informe de la Base de Datos | PEP-IBD | docx | PEP |
| F | Código Backend | //PEP-DB |  | PEP |
| F | Código Frontend | //PEP-DF |  | PEP |
| S | Postgres 14 | - | - | PEP |
| S | Instalador de Rust | - | - | PEP |
| S | React | - | - | PEP |
| E | Informe de Pruebas | PEP-IP | docx | PEP |
| E | Diseño de Arquitectura del Sistema | PEP-DAS | docx | PEP |
| E | Informe UAT | PEP-IUAT | docx | PEP |
| E | Manual de Usuario | PEP-MU | docx | PEP |
| E | Informe de Despliegue | PEP-ID | docx | PEP |

### Diseño de la estructura del repositorio



Repositorio: DELU

Clientes: FISI - UNAYOE, DC SA, FV, GD, TM

Documentos: Políticas, directrices y planes. Documentos del negocio.

Nuestra repo contiene los diversos proyectos de DELU, una empresa que ha estado trabajando en plataformas tecnológicas innovadoras para diferentes sectores. Aquí, administramos tres proyectos principales: DeluStock, FitClue, y ProfeSoft.

### Definición de Líneas Base

| **Hito** | **Nombre de línea base** | **Items (nomenclatura)** |
| --- | --- | --- |
| Hito 1: 10/09/24 | Línea Base 1 | * Project Charter (PEP-PC.docx) * Tablero de Trello (PEP-TC.docx) * Historia de Usuario 01 (PEP-HU01.docx) * Historia de Usuario 02 (PEP-HU02.docx) * Historia de Usuario 03 (PEP-HU03.docx) * Historia de Usuario 04 (PEP-HU04.docx) * Historia de Usuario 05 (PEP-HU05.docx) * Historia de Usuario 06 (PEP-HU06.docx) * Documento Final de Requisitos (PEP-DFR.docx) * Informe de Control de Calidad (PEP-ICC.docx) |
| Hito 2: 24/10/24 | Línea Base 2 | * Diseño de Diagramas UML (PEP-DUML.docx) * Diseño de Diagrama de Componentes y Deployment (PEP-DDCD.docx) * Diseño de Arquitectura de Gestión de Identidades y Accesos (PEP-DAGIA.docx) * Diseño de Wireframe de baja fidelidad (PEP-DWBF.docx) * Documentación de la API (PEP-DAPI.docx) * Diseño de Mockup (PEP-DM.docx) * Diseño del Prototipo de Alta fidelidad de la Interfaz de Usuario (PEP-DPAFIU.docx) * Informe de Validación del Diseño (PEP-IVD.docx) |
| Hito 3: 14/11/24 | Línea Base 3 | * Informe de la Base de Datos (PEP-IBD.docx) * Documentación Backend (PEP-DB.docx) * Documentación Frontend (PEP-DF.docx) * Informe de Pruebas (PEP-IP.docx) * Diseño de Arquitectura del Sistema (PEP-DAS.docx) * Informe UAT (PEP-IUAT.docx) * Manual de Usuario (PEP-MU.docx) * Informe de Despliegue (PEP-ID.docx) |

## Control de la configuración

### Ejemplos de Solicitudes de cambio

Se presenta la documentación de solicitud de cambio significativa para asegurar la correcta gestión de este cambio y la minimización de riesgos, para ello se describe a continuación cada elemento clave de una solicitud de cambio:

* Id Solicitud: Es un identificador único asignado a cada solicitud de cambio, permite un seguimiento individual y eficiente de cada petición dentro del sistema de gestión de cambios.
* Fecha de Creación: Indica la fecha en que se registró formalmente la solicitud de cambio.
* Sistema: Especifica el sistema, aplicación, o proyecto afectado por la solicitud de cambio.
* Fuente (Stakeholder): Identifica a la persona o grupo que originó la solicitud de cambio.
* Autor (Dueño del proceso): Indica la persona con la autoridad para solicitar el cambio y aceptar la responsabilidad por las consecuencias del mismo.
* Descripción: Ofrece una explicación concisa y precisa del cambio propuesto.
* Justificación: Explica el razonamiento detrás de la solicitud de cambio.

A continuación se muestran seis ejemplos de solicitudes de cambio:

| **Id Solicitud** | **SC015** |
| --- | --- |
| **Fecha de creación** | 25/12/2024 |
| **Sistema** | Plataforma de Evaluación de Profesores (PEP) |
| **Fuente (Stakeholder)** | Miguel Romero (Secretario de la UNAYOE) |
| **Autor (Dueño del proceso)** | Doctora Luisa (Jefe de UNAYOE) |
| **Descripción** | Añadir una moderación automática basada en IA para restringir el acceso a usuarios con comentarios inadecuados. |
| **Justificación** | La moderación manual requiere de un personal dedicado monitoreando todas las interacciones, cosa que hasta la fecha ha sido ineficiente y lenta. Una moderación automática basada en IA permitirá agilizar este proceso. |

| **Id Solicitud** | **SC016** |
| --- | --- |
| **Fecha de creación** | 25/12/2024 |
| **Sistema** | Plataforma de Evaluación de Profesores (PEP) |
| **Fuente (Stakeholder** | Miguel Romero (Secretario de la UNAYOE) |
| **Autor (Dueño del proceso)** | Doctora Luisa (Jefe de UNAYOE) |
| **Descripción** | Los alumnos con segunda y tercera repitencia no deben poder participar en los foros hasta regularizar su matrícula. Solo observar. |
| **Justificación** | Si un alumno tiene repitencia, indica que no se está tomando en serio el curso/profesor y puede recurrir a los foros con intenciones fuera de una discusión informativa. |

| **Id Solicitud** | **SC017** |
| --- | --- |
| **Fecha de creación** | 25/12/2024 |
| **Sistema** | Plataforma de Evaluación de Profesores (PEP) |
| **Fuente (Stakeholder** | Doctora Luisa (Jefe de UNAYOE) |
| **Autor (Dueño del proceso)** | Doctora Luisa (Jefe de UNAYOE) |
| **Descripción** | Añadir un campo para que los profesores puedan responder a las evaluaciones y comentarios de los estudiantes. |
| **Justificación** | Permite un diálogo constructivo y la posibilidad de clarificar malentendidos o brindar respuestas a las críticas. |

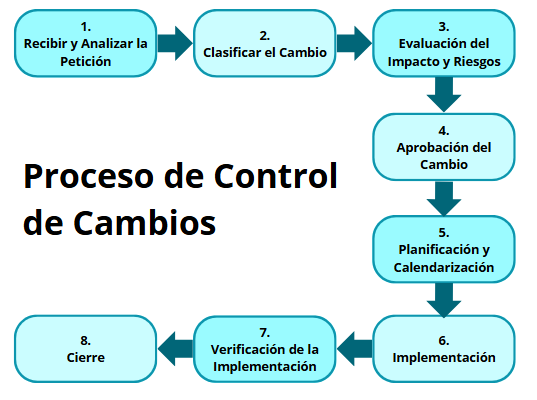
| **Id Solicitud** | **SC018** |
| --- | --- |
| **Fecha de creación** | 25/12/2024 |
| **Sistema** | Plataforma de Evaluación de Profesores (PEP) |
| **Fuente (Stakeholder** | Miguel Romero (Secretario de la UNAYOE) |
| **Autor (Dueño del proceso)** | Doctora Luisa (Jefe de UNAYOE) |
| **Descripción** | Implementar un sistema de registro de auditoría que rastree todas las acciones realizadas en la plataforma (creación de cuentas, evaluaciones, modificaciones, etc.). |
| **Justificación** | Garantizar la trazabilidad y transparencia de todas las acciones dentro del sistema. |

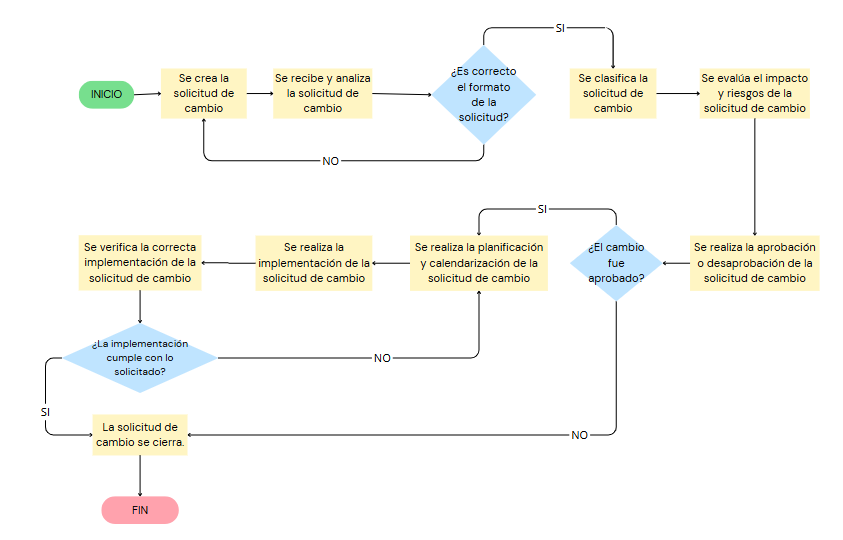
| **Id Solicitud** | **SC019** |
| --- | --- |
| **Fecha de creación** | 25/12/2024 |
| **Sistema** | Plataforma de Evaluación de Profesores (PEP) |
| **Fuente (Stakeholder** | Julisa Sanchez (Secretaria de la UNAYOE) |
| **Autor (Dueño del proceso)** | Doctora Luisa (Jefe de UNAYOE) |
| **Descripción** | Implementar un sistema de ponderación de calificaciones, permitiendo asignar diferentes pesos a diferentes tipos de evaluación (ej: participación en clase, exámenes, trabajos). |
| **Justificación** | Una evaluación más granular y ponderada refleja mejor el desempeño del profesor en distintas áreas de su enseñanza. |

| **Id Solicitud** | **SC020** |
| --- | --- |
| **Fecha de creación** | 25/12/2024 |
| **Sistema** | Plataforma de Evaluación de Profesores (PEP) |
| **Fuente (Stakeholder** | Romina Garcia (Secretaria de la UNAYOE) |
| **Autor (Dueño del proceso)** | Doctora Luisa (Jefe de UNAYOE) |
| **Descripción** | Modificar el sistema de visualización de comentarios para mostrar primero los comentarios más recientes o mejor calificados. |
| **Justificación** | Permitir priorizar la información más relevante para los usuarios que buscan una rápida visión general. |

### Proceso de Control de Cambios

1. Gráfica de proceso





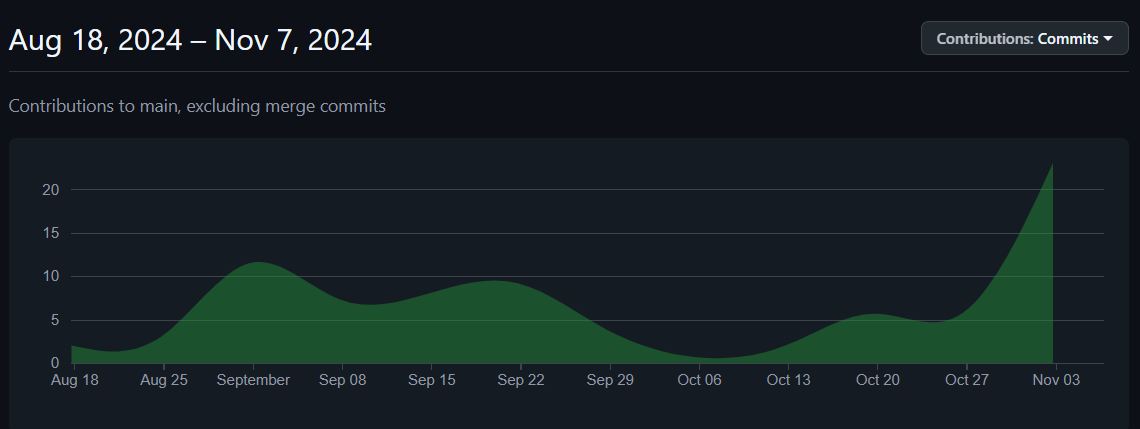
1. Tipos de estados de la solicitud de cambio

Para asegurar la transparencia y el control del proceso de gestión de solicitudes de cambio, es fundamental definir claramente los diferentes estados por los que pasa una solicitud, desde su creación hasta su cierre, por ello se presentan a continuación los estados de una solicitud de cambio, describiendo su significado y la etapa del proceso que representa cada uno, para permitir un seguimiento preciso del progreso y facilitar la comunicación entre las partes involucradas.

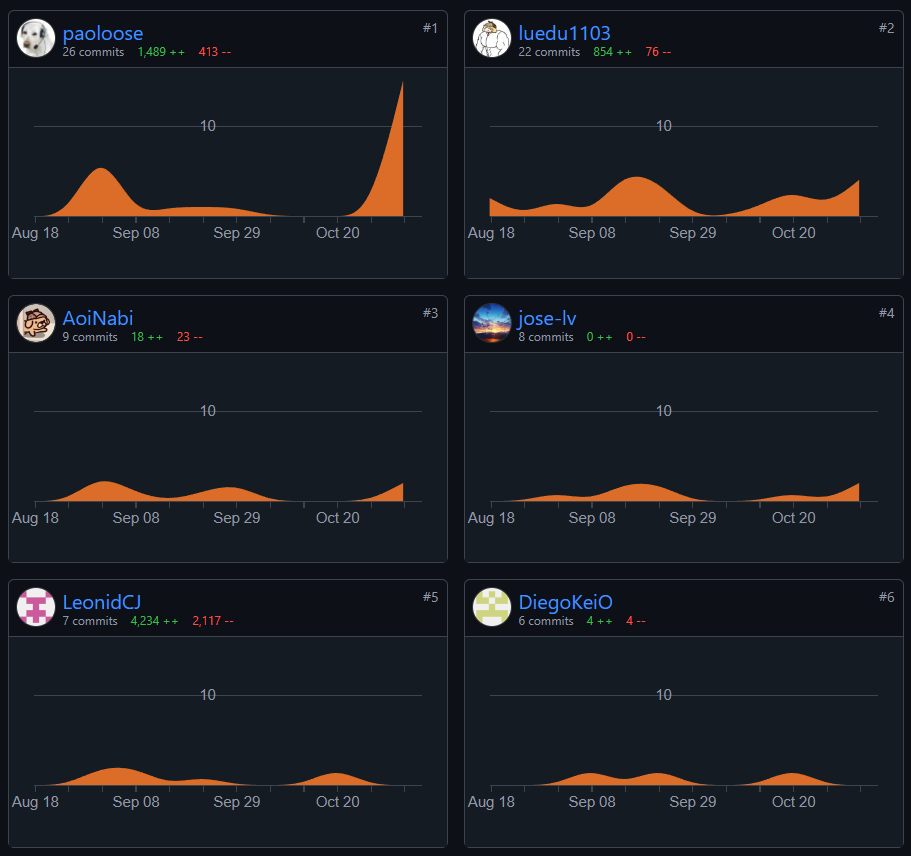
1. Creado: Estado designado a aquellas peticiones de cambios que han sido creados, pero que todavía no han sido enviados para su revisión preliminar.
2. Recibido: Estado para aquellas peticiones de cambios que después de enviados han sido recibidos para su análisis preliminar.
3. Rechazado: Estado para aquellas peticiones de cambios que después de enviados y después de su análisis preliminar, fueron rechazados por no cumplir con los parámetros necesarios o su formato es incorrecto.
4. Clasificado: Estado para aquellas peticiones de cambios que después de ser recibidos han sido categorizadas en función de su prioridad.
5. Evaluado: Estado para aquellas peticiones de cambios que después de ser clasificados, fueron evaluados en cuanto al impacto y riesgos, y esperan ser aprobados o desaprobados.
6. Aprobado: Estado para aquellas peticiones de cambios que después de evaluar su impacto y los riesgos que contraen, fueron aprobados para su ejecución.
7. Desaprobado: Estado para aquellas peticiones de cambios que después de evaluar su impacto y los riesgos que contraen, fueron desaprobados y descartados.
8. En planificación: Estado para aquellas peticiones de cambios que después de aprobarse se encuentran planificando y gestionando su calendario.
9. Planificado: Estado para aquellas peticiones de cambios que después de haberse planificado, se encuentran en estado de espera para ser implementados.
10. En implementación: Estado para aquellas peticiones de cambios que después de planificarse, están en proceso de ejecutarse e implementar su cambio.
11. Implementado: Estado para aquellas peticiones de cambios que después de implementarse, esperan la aprobación de los interesados.
12. Verificado: Estado para aquellas peticiones de cambios que han sido implementadas y verificadas por los interesados, si sus cambios fueron hechos según lo planificado.
13. En corrección: Estado que es la implementación de las correcciones que se indicaron al verificarse y que volverá al estado de verificación.
14. Cerrado: Estado final para aquellas peticiones de cambios que han sido implementadas, verificadas, y las correcciones (si las hay) implementadas y verificadas. La solicitud de cambio se cierra y se registra la información relevante.

## Estado de la configuración

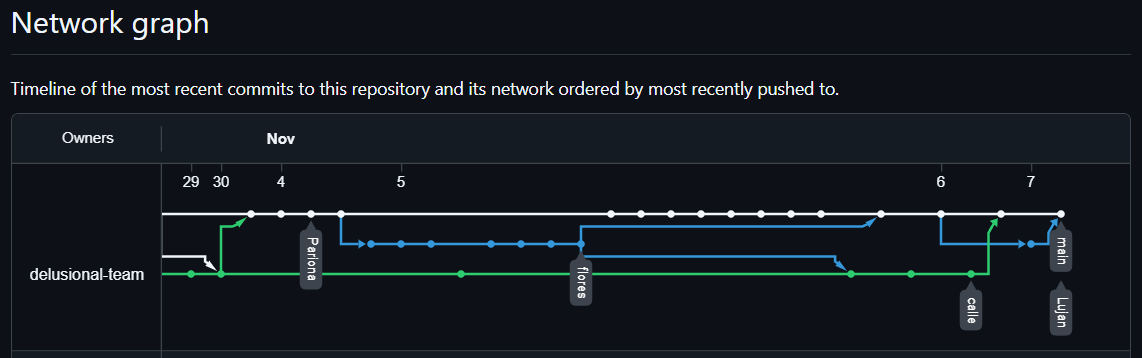
### Contribución grupal hasta la fecha



### Contribución por integrantes

****

### Network

****

## Auditoría de la GCS

### Auditoría física

| **Ítem** | **Fecha de auditoría** | **Auditor** | **Estado Realizado (✔) No realizado (X)** |
| --- | --- | --- | --- |
| Informe de la Base de Datos | 26/11/24 | Matthew Pariona | **✔** |
| Documentación Frontend | 26/11/24 | Matthew Pariona | **✔** |
| Documentación Backend | 26/11/24 | Matthew Pariona | **✔** |
| Informe de Pruebas | 26/11/24 | Matthew Pariona | **✔** |
| Diseño de Arquitectura del Sistema | 26/11/24 | Matthew Pariona | **✔** |
| Informe UAT | 26/11/24 | Matthew Pariona | **X** |
| Manual de Usuario | 26/11/24 | Matthew Pariona | **X** |
| Informe de Despliegue | 26/11/24 | Matthew Pariona | **✔** |

### Auditoría funcional

| **Historia de usuario** | **Fecha de auditoría** | **Auditor** | **Estado Cumple (✔) No cumple (X)** |
| --- | --- | --- | --- |
| HU01 - Notificaciones automáticas | 26/11/24 | Matthew Pariona | **X** |
| HU02 - Recolección de feedback de estudiantes | 26/11/24 | Matthew Pariona | **✔** |
| HU03 - Comentarios y calificación de los alumnos | 26/11/24 | Matthew Pariona | **✔** |
| HU04 - Búsqueda de profesor | 26/11/24 | Matthew Pariona | **✔** |
| HU05 - Plataforma intuitiva y fácil de navegar | 26/11/24 | Matthew Pariona | **✔** |
| HU06 - Seguridad y protección de datos | 26/11/24 | Matthew Pariona | **✔** |

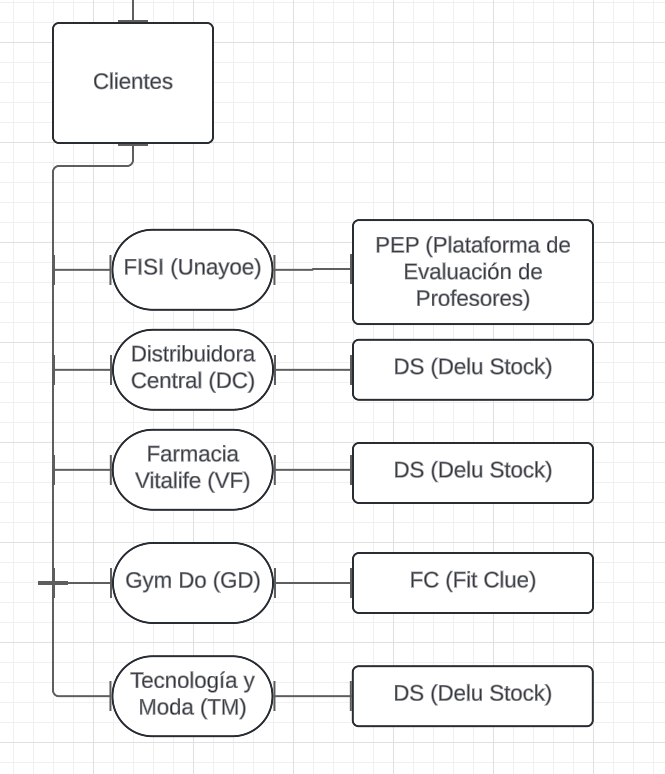
# Gestión de release y entrega del software

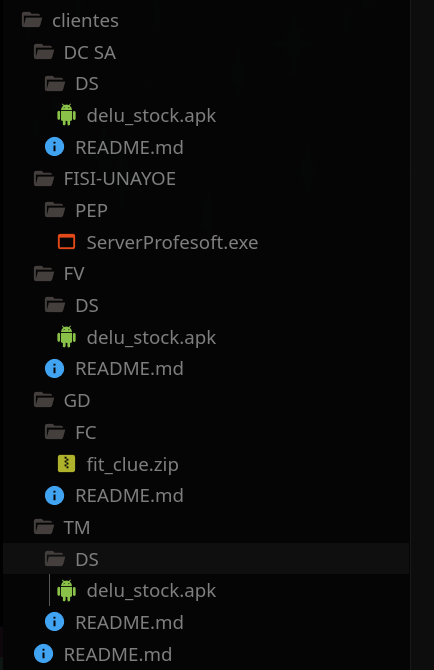
## Gestión de release

Todos los releases de Software que se entregan a los clientes se almacenan en la carpeta de Clientes. Por cada cliente, se crea una carpeta con su nomenclatura. Cada una de estas carpetas contiene las releases de los softwares separadas por proyecto.

En el caso de releases que contengan múltiples carpetas, estas se comprimirán en un zip. En el caso de releases de archivos únicos, esas se almacenarán tal cual.

A continuación, se adjuntan ejemplos de la estructura de esta carpeta.





## Entrega del software (Pase a producción)

Para el pase a producción distinguimos tres tipos de softwares:

* APIs, microservicios

Todos los servicios y sus dependencias son dockerizados. Para esto, una carpeta docker/ se crea en la raíz, que contendrá todos los Dockerfiles necesarios para levantar el proyecto.

Por cada servicio, se creará una carpeta dentro de /opt en nuestro VPS con la nomenclatura del proyecto.

Cada release consistirá de un listado de archivos a enviar documentados para su revisión.

Una vez culminado el empaquetamiento de los archivos en un archivo comprimido, los subiremos en nuestro VPS, los descomprimimos, y construimos el contenedor con los comandos:

docker compose build

docker compose up --force-recreate -d

Verificamos que los servicios se hayan levantado adecuadamente corriendo:

docker compose logs

Y si todo es correcto, salimos del VPS.

De ocurrir algún error, regresamos a la versión anterior e investigamos el problema.

Los artefactos resultantes son subidos en la carpeta Clientes en el proyecto correspondiente.

* Páginas web

Las releases de las páginas web se realizan utilizando Cloudflare Pages. Por cada proyecto web, se escribe una GitHub Action (de workflow dispatch) dentro de la carpeta .github/workflows. Este archivo especificará los pasos necesarios para la compilación del proyecto y su envío a Cloudflare.

Cada release consistirá de un listado de archivos a enviar documentados para su revisión.

Una vez aprobada la revisión, se ejecutará manualmente el action para desplegar el proyecto.

Los artefactos resultantes son subidos en la carpeta Clientes en el proyecto correspondiente.

* Aplicaciones ejecutables

En el caso de aplicaciones ejecutables, el procedimiento es el siguiente.

Cada release consistirá de un listado de archivos a enviar documentados para su revisión.

Una vez aprobada la revisión, se procederá a compilar el proyecto. Cada proyecto deberá incluir documentación sobre su compilación, y, de ser posible, un script que facilite la misma.

El binario es finalmente distribuido en las tiendas virtuales o entregado al cliente, según se requiera.

Los artefactos resultantes son subidos en la carpeta Clientes en el proyecto correspondiente.