Trabajo Final

Desarrollo de Software en Sistemas Distribuidos

Grupo 19 - integrantes:

Fierro, Santiago (21875/7)

Arbeletche, Ezequiel (8170/4)

Dellarupe, Franco (21239/0)

Introducción

Este informe tiene como objetivo documentar la solución desarrollada para dar soporte a *Project Planning*, una aplicación diseñada para asistir en la gestión logística y estratégica de proyectos internacionales de pequeña y mediana escala.

La aplicación *Project Planning* está concebida para facilitar la colaboración entre una red de Organizaciones No Gubernamentales (ONGs) , desde la generación inicial de un proyecto hasta su ejecución y seguimiento. El proceso central aborda:

- La **creación de proyectos** por una ONG originante, incluyendo la presentación de un plan de trabajo y un plan económico de financiamiento.
- La gestión **colaborativa** de este plan, donde se detallan etapas y se generan pedidos de cobertura (económicos, materiales, mano de obra, etc.).
- El **compromiso de colaboración** por parte de otras ONGs de la red, quienes responden a los pedidos especificando su aporte.
- El **monitoreo** de los proyectos una vez que todas sus etapas han sido cubiertas y pasan a la fase de ejecución, asegurando la transparencia del proceso.

La arquitectura de la solución se construyó utilizando un enfoque de sistemas distribuidos, integrando los siguientes componentes clave:

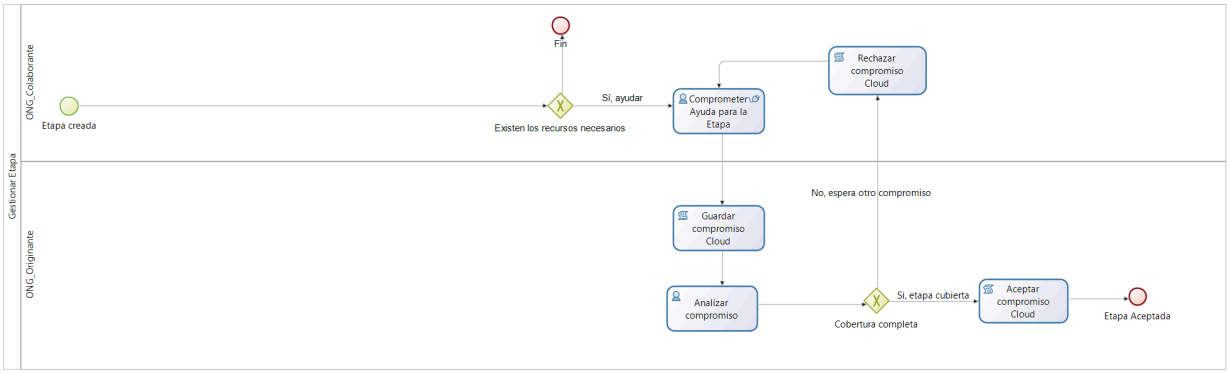
1. Un motor de procesos de negocio implementado con Bonita Open Solution, responsable de orquestar el flujo de las distintas etapas del proyecto.
2. Una capa de servicios web (API REST), desplegada en un entorno PaaS, para gestionar las entidades del negocio.
3. Una aplicación web para la interacción de los diferentes roles de usuario (ONGs originantes, colaboradoras y personal gerencial). La cual interactúa directamente con el proceso de Bonita a través de su API Rest.

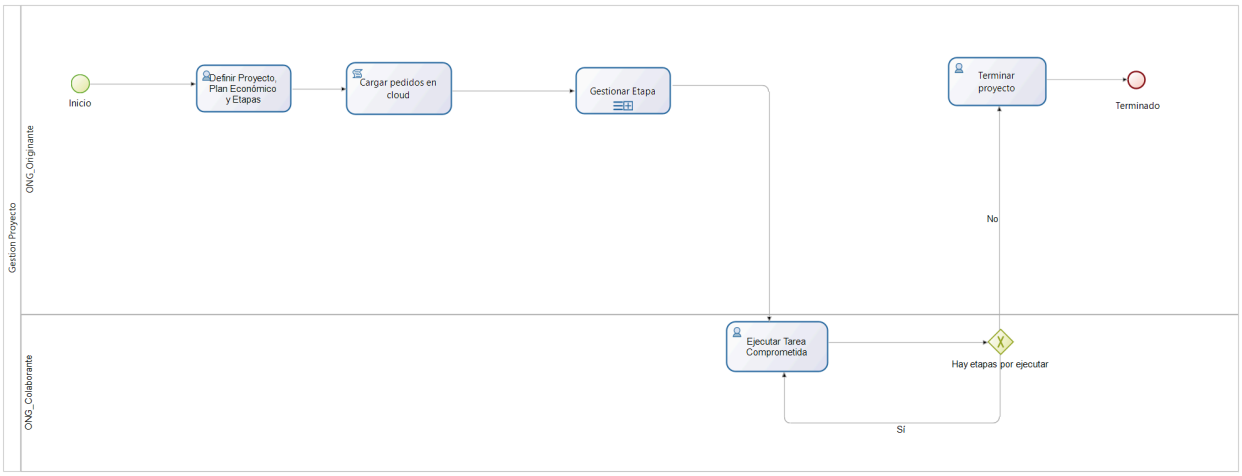
En resumen, el objetivo de esta documentación es detallar las decisiones de diseño, describir los componentes desarrollados (procesos, servicios y aplicación web) y presentar la arquitectura tecnológica completa que implementa y sostiene los proyectos en la red *Project Planning*.

Diagramas de Procesos

Proceso de Gestión del Proyecto

El proceso cumple la función de orquestar la gestión y colaboración en la creación y ejecución de proyectos, centrándose en la colaboración entre la ONG Originante y las ONG Colaboradoras (o la red de ONGs)





El diagrama de proceso se estructura en cuatro carriles ("pools"), representando los roles y proceso principales involucrados:

1. **Gestión Proyecto**: Carril de sistema o administración para el flujo principal del proceso.
2. **ONG Originante**: Responsable de la creación, definición y finalización del proyecto.
3. **ONG Colaboradora**: Responsable de ejecutar las tareas comprometidas para cubrir las etapas del proyecto.
4. **Gestión Etapa**: Carril de sistema para representar el sub-proceso del flujo de una etapa.

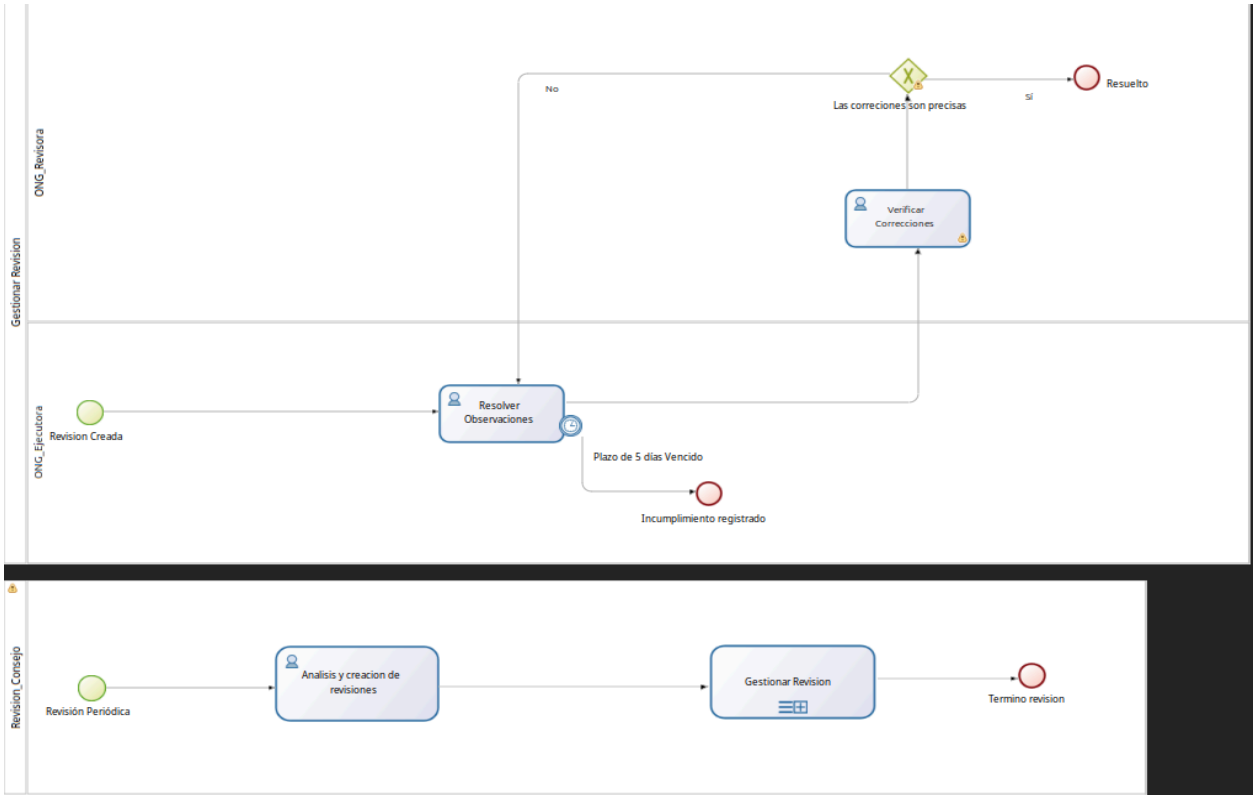
A continuación, se detalla el flujo de actividades:

- **Inicio**: El proceso comienza con la necesidad de una ONG registrada de generar un proyecto y buscar financiamiento.
- **1. Definir Proyecto, Plan Económico y Etapas** (Tarea de Usuario - ONG Originante):
  - La ONG registra el proyecto, presenta un plan de trabajo y un plan económico de financiamiento.
  - Esto implica la creación de las etapas del plan (nombre, fechas) y la especificación del **pedido de cobertura** (económico, materiales, mano de obra, etc.) para cada etapa.
  - **Variables de Negocio**:
    - *proyecto*: Contiene la información del proyecto creado.
    - *etapas*: Lista con los detalles de cada etapa definida.

- **2. Cargar pedidos en cloud** (Tarea de Servicio - Gestión Proyecto):
    - Una vez definidas las etapas y sus pedidos de colaboración, esta tarea representa la interacción con los Servicios Web desarrollados en la plataforma PaaS (cloud). Su objetivo es registrar los pedidos de colaboración en dicho sistema, haciéndolos visibles para el resto de la red de ONGs.
- **3. Gestionar Etapa** (Tarea de Usuario - ONG Originante):
    - Esta actividad permite a la ONG Originante consultar la red de *Project Planning* para ver qué organismos(ONGs Colaboradoras) respondieron a los pedidos de cobertura de cada etapa mediantes “compromisos”.
    - La ONG selecciona la opción que mejor se adecúa de entre los compromisos ofrecidos por los posibles donantes.
    - La tarea finaliza cuando la ONG Originante pudo cubrir todas las etapas del proyecto.
- **4. Ejecutar Tarea Comprometida** (Tarea de Usuario - ONG Colaboradora):
    - Este carril representa la fase de ejecución.
    - Una vez que se selecciona un compromiso de ayuda en la tarea anterior, la ONG Colaboradora ejecuta la tarea o compromiso que asumió.
    - **Variable de Proceso**: cantEtapasEjecutadas (Indica cuántas etapas se llevan ejecutadas hasta el momento).
- **5. Gateway (¿Hay etapas por ejecutar?)**:
    - Este punto de decisión verifica si todas las etapas con compromisos asignados han sido ejecutadas.
    - **"Sí"**: Si aún quedan etapas en fase de ejecución, el flujo regresa a la tarea *Ejecutar Tarea Comprometida*.
    - **"No"**: Si todas las etapas están completadas, el flujo avanza a la tarea de finalización.
- **6. Terminar proyecto** (Tarea de Usuario - ONG Originante):
    - Una vez cubiertas y ejecutadas todas las etapas, la ONG Originante marca el proyecto como completo.
    - Esta tarea permite, además, realizar las consultas de control y seguimiento gerencial requeridas.
- **Terminado**: El proceso finaliza.

Proceso de Gestión de Revisión

Este proceso modela la actividad de control y seguimiento periódico de los proyectos en ejecución por parte del Consejo Directivo, tal como se describe en el enunciado del trabajo.



Flujo Principal:

El proceso se inicia de forma periódica (dos veces por mes) para garantizar el control y seguimiento de los proyectos en ejecución.

- **Revisión Periódica**: El proceso comienza por un evento de tiempo recurrente.
- **1. Análisis y creación de revisiones** (Tarea de Usuario - ONG Consejo):
    - Los miembros del Consejo Directivo realizan una consulta de control y seguimiento sobre los proyectos activos, generando y cargando *observaciones* para los proyectos en ejecución.
    - **Variable de Proceso**: observaciones (Lista de observaciones para el proyecto).
- **2. Gestionar Revisión** (Subproceso Embebido - Revisión Periódica):
    - Esta tarea representa el subproceso detallado a continuación, donde se maneja la resolución y verificación de una observación específica.
- **Finalización**: El flujo principal finaliza una vez completado el subproceso de gestión de la revisión.

Subproceso: Gestionar Revisión

Este subproceso maneja la interacción entre el Consejo Directivo y la ONG ejecutora para resolver una observación específica.

- **Revisión Creada**: Inicia cuando el Consejo crea una observación.
    - **Variable de Proceso Clave**: observación (Detalle de la observación específica a resolver).
- **1. Resolver Observaciones** (Tarea de Usuario - ONG Ejecutora):
    - La ONG ejecutora del proyecto recibe las observaciones.

    - El flujo incluye un *Evento Límite de Tiempo* (Timer) de 5 días asociado a esta tarea.
        - Si la tarea **no se completa** a tiempo, el flujo sigue la línea inferior hacia el evento *Incumplimiento registrado*.
        - Si la tarea **se completa** dentro del plazo, el flujo continúa hacia la *Verificación*.
- **2. Verificar Correcciones** (Tarea de Usuario - ONG Revisora):
    - La ONG Revisora, es decir un miembro del Consejo, examina las correcciones realizadas por la ONG ejecutora.
- **3. Gateway (¿Las correcciones son precisas?)**:
    - Este punto de decisión determina la calidad de la resolución:
        - **"Sí"**: La corrección es satisfactoria. El subproceso finaliza en el estado *Resuelto*.
        - **"No"**: La corrección no es precisa. El flujo regresa a la tarea *Resolver Observaciones* para que la ONG Ejecutora realice los ajustes necesarios.
- **Resuelto**: El subproceso termina de manera exitosa.
- **Incumplimiento registrado**:El subproceso termina si la ONG Ejecutora no resolvió las observaciones en el plazo de 5 días.

Descripción de los Web Services Desarrollados (API REST Cloud)

La capa de servicios web fue desarrollada y desplegada en la URL base `https://dssd2025-cloud.onrender.com/Dssd2025Cloud/doc` como una API RESTful, accesible a través de su documentación `Swagger`. Estos servicios son el núcleo de la interacción de datos entre la aplicación web, el proceso Bonita y la persistencia de la información del negocio, principalmente basada en los pedidos y los compromisos realizados. Todos los servicios de la sección 2 y 3 descritos a continuación, requieren incluir una cabecera Authorization con el token JWT generado a través del servicio de Login.

1. Autenticación y Autorización (JWT)

Servicio	Método	Endpoint	Entradas	Funcionalidad
Generar Token	POST	/api/v1/auth/login	username, password	Autenticación y generación de Token JWT.

2. Servicios de Pedidos

Servicio	Método	Endpoint	Entradas	Funcionalidad

Crear Pedido	POST	/api/v1/pedidos	proyectoId, etapaId, descripcion	Registra un nuevo pedido de colaboración.
Obtener Pedidos	GET	/api/v1/pedidos		Lista todos los pedidos de colaboración existentes en el sistema.
Obtener pedido por ID	GET	/api/v1/pedidos/{pedidoId}	pedidoId	Devuelve los detalles de un pedido específico.
Modificar Pedido	PUT	/api/v1/pedidos/{pedidoId}	id, proyectoId, etapaId, descripcion, estado	Permite actualizar la información y el estado de un pedido específico.
Listar Compromisos de un Pedido	GET	/api/v1/pedidos/{pedidoId}/compromisos	pedidoId (Path)	Permite obtener todos los (compromisos) asociados a un pedido particular.

3. Servicios de Compromisos

Servicio	Método	Endpoint	Entradas	Funcionalidad
Crear Compromiso	POST	/api/v1/compromisos	pedidoId, ongColaboranteId, descripcion	Permite a una ONG Colaboradora registrar formalmente un ofrecimiento de ayuda para cubrir un pedido.
Obtener Compromisos	GET	/api/v1/compromisos	Ninguna	Lista todos los compromisos de colaboración.
Obtener Compromiso por ID	GET	/api/v1/compromisos/{compromisoId}	compromisoId (Path)	Recupera los detalles de un compromiso específico.
Aceptar Compromiso	PUT	/api/v1/compromisos/{compromisoId}/aceptar	compromisoId (Path)	Utilizado por la ONG Originante para aceptar una propuesta, cambiando el estado del compromiso a "ACEPTADO" y el pedido asociado a "CUBIERTO".
Rechazar Compromiso	PUT	/api/v1/compromisos/{compro	compromisoId (Path)	Utilizado por la ONG Originante para rechazar

		misoId}/rechazar		una propuesta, cambiando el estado del compromiso a "RECHAZADO".

Descripción de la aplicación web y funcionalidad

HU-01 - Ingresar a la aplicación

Como ONG registrada
Quiero ingresar al panel de proyectos
Para crear proyectos, hacer seguimiento de pedidos y registrar compromisos de ayuda

HU-02 - Crear un proyecto

Como ONG registrada
Quiero crear un nuevo proyecto
Para iniciar el seguimiento de sus etapas y pedidos de ayuda asociados

HU-03 - Listar proyectos existentes

Como ONG registrada
Quiero visualizar una lista de mis proyectos
Para comprender su estado y progreso

HU-04 - Visualizar detalle de un proyecto

Como ONG registrada
Quiero ver la información completa de un proyecto
Para visualizar compromisos de ayuda registrados, ejecutar etapas y registrar la finalización de un proyecto.

HU-05 - Ver pedidos de ayuda asociados a una etapa

Como ONG registrada
Quiero conocer si una etapa requiere aportes externos
Para registrar compromisos de ayuda

HU-06 - Ver compromisos de aportes

Como ONG registrada
Quiero ver los compromisos registrados para una etapa
Para evaluar el aporte propuesto, y decidir si es aceptado o rechazado

HU-07 - Listar mis proyectos cubiertos

Como ONG registrada
Quiero ver mis proyectos que se encuentren cubiertos

Para evaluar su evolución y decidir el comienzo de la ejecución de sus etapas.

HU-08 - Ejecutar etapa

Como ONG registrada
Quiero ejecutar una etapa de uno de mis proyectos cubiertos
Para comenzar a desarrollar el proyecto

HU-09 - Finalizar un proyecto

Como ONG registrada
Quiero cerrar uno de mis proyectos del cual ya se hayan ejecutado todas sus etapas
Para dar por finalizado el proyecto correctamente.

HU-10 - Ver indicadores

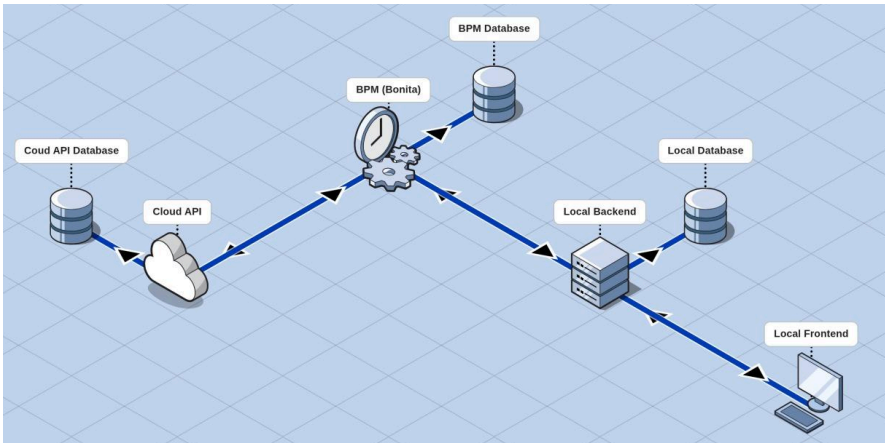
Como directivo
Quiero conocer los indicadores de proyectos y compromisos.
Para entender la evolución de los proyectos y los aportes de las ONGs..

Consultas gerenciales

Se generaron 4 indicadores que involucran información de la base de datos de la aplicación web local, datos de Bonita a través de la REST API y la API cloud.

- Porcentaje de proyectos que terminaron en término y fuera de término: se comparó la fecha de finalización definida en la base de datos local, contra la fecha de finalización del proceso en Bonita.
- ONG más colaboradora: se consultan las etapas que solicitaron ayuda y las ONG que registraron colaboración en el cloud, para contabilizar la que tenga mayor ocurrencia.
- Rubro más solicitado y top 3 colaboradores: se consultan todas las etapas para identificar el rubro más solicitado, y en las ONG que registraron colaboración en el cloud.

Descripción de la arquitectura



- **Aplicación Local**: compuesta por una capa visual en Angular 17, una capa de servicios en Spring Boot JDK 21 y una base de datos PostgreSQL 15.
- **BPMS**: sistema de gestión de procesos implementado en Bonita Studio, para centralizar la evolución de los procesos.
- **Aplicación Cloud:** compuesta por una capa de servicios Spring Boot JDK 21 y una base de datos PostgreSQL 15. Desplegado en Render (PaaS - Platform as a Service).

Extensión de trabajo práctico

Kubernetes

Se desarrolló una aplicación Spring Boot con un endpoint "/token" que devuelve un número único, incremental y globalmente ordenado.
La aplicación utiliza una base de datos en memoria Redis para sincronizar el numerador entre las diferentes réplicas del contenedor.
Se generó el código necesario para desplegar la aplicación web que y en el mismo repositorio se armaron generaron los archivos para el despliegue.
En el carpeta **k8s** se encuentran los archivos para el despliegue de la aplicación utilizando **Microk8s** como implementación de **Kubernetes,** adicionalmente de un archivo que despliega una instancia única de Redis.
El archivo ***2-deployment.yaml*** define el estado deseado del clúster. Se agregó configuración de readiness y liveness prob's, para evitar que los Pods atiendan peticiones cuando aún no están totalmente listos.
Se vincula con Redis a través de variables de entorno y el nombre del servicio definido en el primer despliegue.
El archivo ***3-service.yaml*** define la abstracción de una colección de Pods, para que los clientes puedan interactuar con los contenedores de forma transparente.
En el archivo ***4-ingress.yaml*** está definido el ingreso al servicio por fuera del clúster. Permite el ruteo de tráfico externo a uno o más servicios internos. Se definió el path de coincidencia y el servicio asociado.
Por último en el archivo ***5-hpa.yaml*** está configurado el escalado horizontal del recurso, para dar respuesta a un incremento de la demanda.

Jenkins

En el mismo repositorio, en la carpeta **jenkins** se encuentran los archivos para instalar y levantar el servicio de Jenkins. Además se armó un pipeline que simula las tareas de CI (Continuous Integration) de un repositorio público.
El pipeline se puede vincular con un Trigger "escuchando" los cambios de un rama GIT para disparar la ejecución.
En el pipeline se define la utilización de cualquier **agente** y el **tool** "JDK25" (creado a través de plugin de Jenkins para agregar una JDK más nueva). También define **parámetros** con valores por defecto, donde indica la rama y el commit_id que van a clonarse del repositorio público. Luego **variables de entorno** que indican la ruta del repositorio, la carpeta del build y el nombre del artefacto.
Luego están descritas las etapas a ejecutar:

- Clonado del repositorio

- Compilación del proyecto
- Ejecución de test automatizados
- Armado del paquete de la aplicación
- Publicación del artefacto. En esta etapa a modo de ejemplo, se mueve el JAR generado a una carpeta de Jenkins. Pero en un pipeline productivo podría publicar el artefacto en un repositorio para ser utilizado en despliegues posteriores.

Por último, dependiendo de la ejecución se imprimen los mensajes definidos en la sección “post”.