Proyecto

Presentado por:

Cristian Medina - crmedinab@unal.edu.co

Cristian Montañez - cmontanez@unal.edu.co

Justin Rodriguez - jusrodriguez@unal.edu.co

Sergio Ruiz – seruizh@unal.edu.co

Profesor:

Oscar Eduardo Alvarez Rodriguez

14/11/2024

Universidad Nacional de Colombia

Facultad de Ingeniería

Ingeniería de Software

Departamento de Ingeniería de Sistemas

Plataforma de Objetos Perdidos y Encontrados

2. Levantamiento de requerimientos

Problemática

La comunidad universitaria, conformada por estudiantes, profesores y personal administrativo de la Universidad Nacional de Colombia, tiene dificultades en la gestión de objetos perdidos. Ya que los métodos actuales son dispersos y lentos, dificultando la búsqueda. Esto genera pérdida de tiempo, frustración y pérdidas económicas para los afectados.

Además, la falta de un sistema centralizado provoca desconfianza en la comunidad y que los objetos sin reclamar se acumulen en diferentes dependencias sin un mecanismo efectivo para su devolución. Esta situación impacta negativamente la experiencia universitaria, generando estrés e incomodidad.

Objetivo

El objetivo principal es desarrollar una plataforma intuitiva y eficiente que permita a los miembros de la comunidad universitaria reportar la pérdida o el hallazgo de objetos dentro del campus. La plataforma debe simplificar el proceso de registro, búsqueda y recuperación de objetos perdidos, mejorando la experiencia de la comunidad universitaria y reduciendo la cantidad de objetos que permanecen sin reclamar.

Origen de la Idea

La idea del proyecto surge a partir de la observación y experiencia directa de la problemática recurrente dentro de la Universidad Nacional de Colombia que es la pérdida y recuperación de objetos en el campus.

Miembros del equipo, que han trabajado como auxiliares estudiantiles en distintos edificios de la universidad, han visto que se encuentran objetos extraviados diariamente, los cuales en muchos casos no logran ser devueltos a sus dueños, lo que genera una acumulación constante de estos objetos en los edificios. Aunque no es una idea revolucionaria, ya que proyectos de este tipo han sido propuestos en semestres anteriores, ninguno ha sido implementado de una manera efectiva.

Elección de la idea

Inicialmente se realizó una reunión de lluvia de ideas en la que se propusieron varias ideas de proyectos, incluyendo la creación de un e-commerce de zapatos, la cual fue elegida. Sin embargo, tras analizar su propósito, se concluyó que dicha idea carecía de impacto significativo. A partir de esto, se estuvo en búsqueda de un proyecto el cual abordar, donde los miembros del equipo compartieron experiencias sobre la problemática de los objetos

perdidos en la universidad y se definió como un proyecto importante para abordar. Como resultado, se decidió desarrollar este proyecto.

Una vez definido el proyecto, se llevó a cabo la asignación de roles dentro del equipo. Donde cada integrante presentó sus habilidades, intereses y experiencia en distintas áreas del desarrollo y a través de un consenso, se estableció la distribución más adecuada de responsabilidades, asegurando que cada miembro pudiera aportar de manera óptima al proyecto.

Integrante	Rol
Daniel Montañez	Desarrollador Full Stack (Backend y Frontend)
Cristian Medina	Desarrollador Backend
Justin Rodriguez	Desarrollador Frontend y Diseñador de Interfaz
Sergio Ruiz	Desarrollador Full Stack (Backend y Frontend)

Expectativas de los Usuarios

Los usuarios esperan que el sistema permita:

- Reportar objetos perdidos o encontrados fácilmente.
- Buscar objetos por categoría, ubicación y fecha.
- Facilitar la comunicación entre el dueño y quien encontró el objeto.

Expectativas del desarrollo del proyecto

Como equipo de desarrollo, esperamos obtener varios aprendizajes al llevar a cabo este proyecto, tanto a nivel académico como profesional. En primer lugar, buscamos aplicar de manera práctica los conocimientos teóricos adquiridos en la materia de Ingeniería de Software. Desarrollar una solución para una problemática real nos permite comprender mejor los retos que implica la implementación de un sistema funcional y eficiente.

Además, este proyecto representa una oportunidad para mejorar nuestras habilidades técnicas en diversas áreas del desarrollo de software. Trabajaremos en el diseño de bases de datos, programación web, gestión de proyectos y pruebas de software, esto preparándonos para desafíos en el ámbito profesional.

El trabajo en equipo es otro aspecto fundamental en este proceso. A través de la colaboración, aprenderemos a comunicarnos de manera efectiva, resolver problemas en conjunto y optimizar nuestro flujo de trabajo. Que es una de las habilidades más esenciales para cualquier entorno de desarrollo profesional.

Por otro lado, nos motiva el impacto que esta plataforma tendrá en la comunidad universitaria. Crear una herramienta que facilite la recuperación de objetos perdidos mejorará la experiencia dentro del campus y brindará una solución a un problema recurrente. Además, promoviendo una cultura colaborativa en la universidad.

Requerimientos Funcionales

1. **Registro de usuarios:**

   Los usuarios deben registrarse utilizando su correo institucional de la universidad.

   Debe enviarse un correo de confirmación para validar la cuenta.

2. **Inicio de sesión:**

   Autenticación mediante correo institucional y contraseña.

3. **Cierre de sesión:**

   Los usuarios deben poder cerrar sesión de manera segura.

4. **Reporte de objetos:**

   Los usuarios deben poder registrar un objeto perdido con los siguientes datos:

   - Nombre del objeto.
   - Categoría (llaves, electrónicos, ropa, etc.).
   - Descripción breve.
   - Estado (perdido,encontrado)
   - Ubicación donde se extravió o se encontró.
   - Fecha del extravío.
   - Imagen opcional.
   - Forma de contacto (correo Institucional, número telefónico)

5. **Editar los reportes:**

   Los usuarios deben poder editar los reportes creados por ellos mismos para actualizar detalles.

6. **Eliminar los reportes.**

   Los usuarios deben poder eliminar reportes creados por ellos mismos

7. **Historial de reportes:**

   Visualización de los objetos registrados (perdidos/encontrados) por el usuario.

   Mostrando el estado de los objetos reportados.

8. **Visualización de Objetos:**

   Mostrar un listado de objetos con información detallada de los objetos perdidos y encontrados, incluyendo imágenes.

9. **Búsqueda por palabras clave (nombre o descripción del objeto).**

   Permite buscar objetos ingresando el nombre o una descripción relacionada.

10. **Búsqueda por Categoría.**

Filtra los resultados según categorías predefinidas para agilizar la búsqueda.

11. Búsqueda por Fecha (rango de fechas).

Ofrece la posibilidad de establecer un rango de fechas para identificar objetos reportados en un período específico.

12. Búsqueda por Ubicación.

Permite delimitar la búsqueda a una ubicación específica, ayudando a localizar objetos según el lugar donde fueron encontrados o perdidos.

13. Notificaciones Automatizadas:

Enviar notificaciones al usuario cuando un objeto encontrado coincida con la descripción de su objeto perdido.

14. Solicitud de Reclamación del Objeto

Si un usuario identifica que un objeto perdido fue encontrado o que un objeto registrado como encontrado le pertenece:

Puede enviar una solicitud de reclamación desde la plataforma.

15. Validación de Reclamos

El usuario que reportó el objeto (ya sea como perdido o encontrado) recibirá una notificación sobre la solicitud de reclamación enviada.

La notificación incluirá información sobre el solicitante, como su nombre y correo electrónico, para facilitar el contacto directo.

16. Visualización de usuarios

El sistema debe mostrar un listado completo de usuarios con su información

17. Búsqueda de usuarios por nombre

Búsqueda de usuarios según su nombre registrado.

18. Búsqueda de usuarios por correo electrónico

Localización de usuarios específicos mediante su dirección de correo.

19. Suspender o reactivar cuentas

Los administradores podrán suspender una cuenta para restringir el acceso del usuario a la plataforma.

La reactivación permitirá restablecer el acceso completo de una cuenta suspendida.

20. Eliminar cuentas

Los administradores tendrán la opción de eliminar permanentemente una cuenta.

Requerimientos No Funcionales

1. **Seguridad**

   Garantizar la protección de datos personales y la autenticidad de los usuarios mediante sistemas de autenticación y encriptación.

2. **Rendimiento**

   La aplicación debe ser capaz de gestionar múltiples registros y búsquedas en tiempo real sin afectar la experiencia del usuario.

3. **Escalabilidad**

   Preparar el sistema para posibles expansiones, como incluir otros campus universitarios o integrar nuevas funcionalidades.

4. **Usabilidad**

   Diseñar una interfaz intuitiva y accesible, asegurando que cualquier usuario pueda interactuar con la plataforma sin dificultad.

5. **Accesibilidad**

   Compatible con dispositivos móviles, tablets y computadoras.

3. Análisis de requerimientos

Priorización

Para el análisis de requerimiento se plantea clasificar inicialmente los requerimientos siguiendo el método MoSCoW, donde se clasifican los elementos según lo esencial que sean para la completitud del proyecto. De esta manera se tiene la siguiente convención:

- Must have (M): Requisitos esenciales para el funcionamiento del sistema.
- Should Have (S): Características importantes, pero que pueden ser pospuestas.
- Could Have (C): Deseables, pero no fundamentales para la primera versión.
- Won't Have (W): Elementos que no se incluirán en este ciclo de desarrollo.

Requerimientos funcionales

- Registro de usuarios (M)
- Inicio de sesión (M)
- Cierre de sesión (M)
- Reporte de objetos (M)
- Editar los reportes (M)
- Eliminar los reportes (M)
- Historial de reportes (M)
- Visualización de Objetos (M)
- Búsqueda por palabras clave (S)
- Búsqueda por Categoría (S)
- Búsqueda por Fecha (rango de fechas) (S)
- Búsqueda por Ubicación (S)
- Solicitud de Reclamación del Objeto (S)
- Validación de Reclamos (S)
- Notificaciones Automatizadas (W)
- Eliminar cuentas (solo administradores) (W)
- Suspender o reactivar cuentas (W)
- Búsqueda de usuarios por nombre (W)
- Búsqueda de usuarios por correo electrónico (W)
- Visualización de usuarios (listado completo de usuarios con información básica) (W)
- Funcionalidades más avanzadas de gestión de usuarios (W)

Requerimientos No Funcionales

- Seguridad (M)
- Usabilidad (M)
- Accesibilidad (M)
- Rendimiento (S)
- Escalabilidad (C)

Estimación de tiempos

Estimación de tiempos y organización de los requerimientos en un orden lógico de implementación:

Must Have (M)

- **Registro de usuarios: 3 días** – Se requiere validar los datos ingresados, y conectarlo con la base de datos MySQL. La autenticación se hará con JWT para garantizar sesiones seguras.
- **Inicio de sesión: 3 días** – Implementación de autenticación con JWT, gestión de sesiones y validaciones de credenciales en el backend. Es crucial para la seguridad del sistema.
- **Cierre de sesión: 1 día** – Sencillo de implementar, implica invalidar el token JWT en el frontend y limpiar las sesiones activas.
- **Reporte de objetos: 2 días** – Creación de formularios en React con validaciones de datos, envío al backend y almacenamiento en MySQL. Se prioriza una interfaz intuitiva.
- **Editar los reportes: 3 días** – Se debe permitir modificar reportes con restricciones de permisos y asegurar la actualización en tiempo real en la base de datos.
- **Eliminar los reportes: 2 días** – Implementación de eliminación lógica con confirmaciones de usuario para evitar la pérdida accidental de datos.
- **Historial de reportes: 3 días** – Implementación de consultas optimizadas para mostrar reportes anteriores con filtros básicos sin afectar el rendimiento.
- **Visualización de objetos: 3 días** – Diseño de la interfaz para mostrar los reportes almacenados con paginación y filtros básicos.
- **Seguridad: 2 días** – Protección contra inyecciones SQL, XSS y validaciones de API para evitar vulnerabilidades.
- **Usabilidad: 5 días** – Se enfocará en la experiencia de usuario con un diseño intuitivo y pruebas de navegación para mejorar la accesibilidad.
- **Accesibilidad: 3 días** – Se implementarán estándares como etiquetas ARIA, contraste adecuado y compatibilidad con navegación por teclado.

Total Must Have: 30 días

Should Have (S)

- **Búsqueda por palabras clave: 3 días** – Se implementará un motor de búsqueda en MySQL con índices optimizados para mejorar la velocidad de consulta.
- **Búsqueda por categoría: 2 días** – Implementación de filtrado de reportes según categoría mediante consultas optimizadas en MySQL.
- **Búsqueda por fecha (rango de fechas): 2 días** – Uso de rangos de fechas en MySQL con índices para optimizar las consultas.
- **Búsqueda por ubicación: 1 día** – Se integrará con ubicaciones predefinidas en la base de datos.
- **Solicitud de reclamación del objeto: 3 días** – Se creará un sistema de reclamación con validaciones y estados del objeto (en revisión, aprobado, rechazado).
- **Validación de reclamos: 3 días** – Implementación de un sistema de aprobación manual o automatizado basado en reglas predefinidas.
- **Rendimiento: 3 días** – Se optimizarán consultas y carga de datos para garantizar tiempos de respuesta rápidos y eficiente uso de los recursos.

Total Should Have: 17 día

Could Have (C)

- **Escalabilidad: 3 días** – Se diseñará una arquitectura flexible y modular para permitir el crecimiento del sistema sin afectar su rendimiento.

Total Could Have: 3 días

Won't Have (W)

- **Notificaciones automatizadas: 5 días** – Se requeriría integración con Firebase o un servicio de correos electrónicos, además de configuración de eventos automáticos.
- **Eliminar cuentas (solo administradores): 3 días** – Implementación de eliminación permanente con validaciones y políticas de recuperación de datos.
- **Suspender o reactivar cuentas: 3 días** – Creación de estados de usuario en la base de datos con restricciones de acceso según reglas establecidas.
- **Búsqueda de usuarios por nombre: 2 días** – Optimización de consultas en MySQL con índices para mejorar la eficiencia.
- **Búsqueda de usuarios por correo electrónico: 2 días** – Implementación similar a la búsqueda por nombre, pero con validaciones adicionales.
- **Visualización de usuarios: 3 días** – Creación de un panel administrativo con información básica de los usuarios de forma segura.
- **Funcionalidades más avanzadas de gestión de usuarios: 8 días** – Desarrollo de permisos personalizados y herramientas de administración avanzadas.

Total Won't Have: 26 días (Fuera del alcance actual)

A partir de este enfoque, se puede elaborar el siguiente cronograma preliminar para iniciar el desarrollo:

Prioridad	Requerimiento	Días
Must Have (M)	Registro de usuarios	3
	Inicio de sesión	3
	Cierre de sesión	1
	Reporte de objetos	2
	Editar los reportes	3
	Eliminar los reportes	2
	Historial de reportes	3
	Visualización de objetos	3
	Seguridad	2
	Usabilidad	5
	Accesibilidad	3
Total (M)		30
Should Have (S)	Búsqueda por palabras clave	3
	Búsqueda por categoría	2
	Búsqueda por fecha (rango de fechas)	2
	Búsqueda por ubicación	1
	Solicitud de reclamación del objeto	3
	Validación de reclamos	3
	Rendimiento	3
Total (S)		17
Could Have (C)	Escalabilidad	3
Total (C)		3
Total Estimacion		40

4: Análisis gestión de software

- **Equipo**: 4 desarrolladores, 2 desarrolladores full stack, 1 desarrollador front y un desarrollador back
- **Duración**: 40 días
- **Tecnologías**:
    - **Frontend**: React
    - **Backend**: Node.js con Express.
    - **Base de datos**: MySql
    - **Notificaciones**: Integración futura con Firebase o un servicio de emails.
    - **Autenticación**: JWT para sesiones seguras.

Estimación del costo del equipo de desarrollo para los 40 días del proyecto:

1. **Desarrolladores Full Stack (2 personas):**
    - Un **Full Stack Developer** Junior gana aproximadamente **2.800.000 COP** al mes.
    - Para 40 días de trabajo (aproximadamente 1.33 meses), cada **Full Stack Developer** costaría **3.724.000 COP**.
    - **Costo total para los dos Full Stack Developers**: **7.448.000 COP**.
2. **Desarrollador Front-End (1 persona):**
    - Un **Front-End Developer** junior gana aproximadamente **2.200.000 COP** al mes.
    - Para 40 días de trabajo, su costo sería **2.926.000 COP**.
3. **Desarrollador Back-End (1 persona):**
    - Un **Back-End Developer** junior gana aproximadamente **2.400.000 COP** al mes.
    - Para 40 días de trabajo, su costo sería **3.192.000 COP**.

Costo total del equipo de desarrollo para los 40 días:

- **Full Stack (2 personas)**: **7.448.000 COP**
- **Front-End (1 persona)**: **2.926.000 COP**
- **Back-End (1 persona)**: **3.192.000 COP**

Total estimado del equipo de desarrollo para 40 días: 13.566.000 COP

Con estos valores, el total estimado para el equipo de desarrollo durante los 40 días de trabajo es **13.566.000 COP**.