Universidad del Quindío
Programa de Ingeniería de Sistemas y Computación
Programación Avanzada - Proyecto Final
Febrero de 2025

Proyecto Final del espacio académico

La seguridad ciudadana y el bienestar comunitario son preocupaciones crecientes en muchas ciudades del mundo. Para abordar estas problemáticas, se necesita desarrollar una plataforma que permita a los ciudadanos reportar de manera sencilla y eficiente situaciones de riesgo o emergencia en su entorno.

Para utilizar esta plataforma, los usuarios deben registrarse proporcionando información básica como su nombre completo, ciudad de residencia, dirección, correo electrónico y contraseña. Una vez registrados, los usuarios pueden iniciar sesión para acceder al sistema y reportar incidentes en su entorno. Cada reporte debe tener un título, categoría, descripción, ubicación (latitud y longitud) y al menos una imagen. Los reportes están categorizados por: seguridad (robos, actividades delictivas), emergencias médicas (accidentes de tránsito, desmayos, etc), infraestructura (calles en mal estado, problemas de alumbrado público), mascotas (mascotas perdidas, encontradas), comunidad (contaminación, basuras, etc).

Dentro de la plataforma, los usuarios pueden visualizar en tiempo real los reportes realizados por otros miembros de la comunidad, comentar en ellos para aportar información adicional y recibir notificaciones sobre incidentes cercanos. Además, el sistema permite a los moderadores verificar y gestionar los reportes, asegurando su relevancia y calidad para fomentar un ambiente seguro y colaborativo.

En esta aplicación se desea contar con dos tipos de usuarios:

Cliente:

- Registro. Cuando el usuario se registra no puede iniciar sesión inmediatamente. Una vez el usuario se registra se le envía un correo electrónico con un código. Y cuando ingrese la primera vez se le pedirá este código para activar su cuenta. Este código debe tener una validez de 15 minutos.
- Login. Solo pueden iniciar sesión los usuarios que ya tienen su cuenta activada.
- Crear reportes, editarlos o borrarlos. El reporte se publica pero sin estar verificado.
- Cambiar el estado de un reporte a Resuelto, cuando el cliente lo considere.
- Botón de "Es importante" para priorizar un reporte. Los usuarios pueden “calificar” como importante cada reporte y con base en esto determinar su severidad.
- Agregar comentarios en los reportes publicados para aportar información adicional.
- Notificaciones en tiempo real: Uso de websockets o Firebase Cloud Messaging (https://firebase.google.com/docs/cloud-messaging) para alertar de reportes nuevos en la zona del usuario. Debe haber un apartado de notificaciones (en forma de lista). También deben haber notificaciones vía correo electrónico.
- Ver el detalle de cada alerta (con los comentarios y la información completa del mismo).
- Editar sus datos personales y eliminar su cuenta.

Administrador o moderador:

- Loguearse.
- Gestionar categorías de reportes (mascotas, delitos, etc).
- Gestionar reportes de usuarios (Verificando, Rechazando, Eliminando).
- Si un moderador rechaza un reporte debe escribir un motivo por el que lo hizo y el cliente que creó el reporte tiene 5 días para hacer las modificaciones pertinentes y enviarlo de nuevo a revisión.
- Marcar cualquier reporte como resuelto.
- Gestionar su propia cuenta (editar, eliminar).
- Generar informes a partir de los reportes de incidentes por sectores donde se generan y/o su categoría en un marco de tiempo dado. Los reportes se deben mostrar en la web y en PDF.

Para tener en cuenta:

- Diseñar diagrama de clases para representar cada entidad del dominio de los datos del sistema.
- El proyecto se debe implementar usando Spring Boot en el backend y Angular en el frontend. Así como MongoDB para la gestión de los datos.
- La dirección de residencia del usuario y la ubicación de los reportes deben obtenerse a través de un mapa, y en la base de datos se debe almacenar la latitud y la longitud correspondientes.
- Para la parte de los mapas se recomienda usar Mapbox (https://www.mapbox.com/).
- Se debe validar que no hayan registros repetidos.
- Cuando un reporte cambia de estado, se debe registrar en un historial de estados, incluyendo la fecha del cambio. Es fundamental garantizar que la trazabilidad de los estados no se pierda en ningún momento.
- Tanto el cliente como el moderador pueden recuperar la contraseña si la olvidan. Al olvidar su contraseña pueden cambiarla por medio de un código enviado a su correo electrónico. Este código debe tener una validez de 15 minutos.
- Cuando un cliente hace un comentario en un reporte, se debe enviar un email (al usuario que creó dicho reporte) con lo que escribió la persona.
- Es necesario implementar un mecanismo que asegure que las notificaciones enviadas a los usuarios sean relevantes y estén relacionadas con su ubicación de residencia. Para ello, se debe definir un rango en kilómetros a la redonda que delimite la relevancia geográfica de las notificaciones.
- Para el manejo de imágenes se debe hacer uso de un servicio externo, puede ser Cloudinary, Flickr, AWS S3, Google Cloud Storage, etc.
- El código fuente del proyecto debe estar en un repositorio de Github. Todos los integrantes del grupo deben contribuir en el desarrollo del proyecto.
- La aplicación web debe contar con una página de inicio diferente para los clientes y para los administradores.
- Los administradores están precargados en la base de datos.
- Las eliminaciones de registros se deben hacer por medio de un campo que indique si están activos o inactivos.

IMPORTANTE:

- Piense en al menos dos necesidades asociadas a este sistema para que las incluya dentro de su modelo. Estas necesidades no pueden ser triviales sino que deben representar algo relevante en el sistema (tanto backend como frontend).
- El proyecto puede realizarse en grupos máximo de 3 personas.

Fases del proyecto final

Fase 1: (25%)

Para la primera fase deben diseñar los mockups (prototipos) para todas las pantallas que tendrá el proyecto final, además, deben realizar un documento (en PDF) con la especificación de los requerimientos del proyecto (Debe seguir los lineamientos de Open API), indicando los datos de entrada y las respuestas esperadas (respuestas con todo tipo de información). Para esta fase se recomiendo visitar las siguientes páginas:: https://editor.swagger.io/ y https://www.openapis.org/

Fase 2 (25%):

- El backend debe implementar completamente la capa de persistencia (modelo de datos) y la capa de negocio (servicios de negocio, autorización, autenticación, API y pruebas).
- Cada requisito del proyecto debe contar con un método en la capa de negocio que lo resuelva, incluyendo las validaciones pertinentes, y estar respaldado por su respectiva prueba unitaria desarrollada con JUnit.
- Los métodos de negocio necesarios en el frontend deben estar expuestos a través de la API RESTful, con la correspondiente validación de roles.
- Para las pruebas unitarias, se debe crear un dataset que contenga al menos cinco registros por cada colección de la base de datos.

Fase 3 (25%):

- Para cada requisito definido en el proyecto, debe implementarse una funcionalidad correspondiente en la aplicación web, garantizando la integración de mecanismos de autenticación y autorización.
- Tanto el backend como el frontend deben estar desplegados en la nube.