Manual Técnico SafeHood360

Edward Alejandro García González

Daniel Fernando Pinto Sabogal

Corporación Universitaria Iberoamericana

Facultad de Ingeniería - Ingeniería de Software

Proyecto de Software

Profesora Tatiana Cabrera

Bogotá, Colombia.

Junio 2025

Tabla de Contenidos

Introducción.....3

Estructura.....4

SafeHood360 Frontend.....4

SafeHood360 Backend.....27

Lista de Figuras

Figura 1 Estructura.....4

Figura 2 Login.....5

Figura 3 Crear usuario.....7

Figura 4 Olvidar contraseña.....10

Figura 5 PQR.....11

Figura 5 Dashboard.....14

Figura 6 Perfil.....17

Figura 7 Reportar.....22

Figura 8 Grupos.....23

Figura 9 Directorio.....25

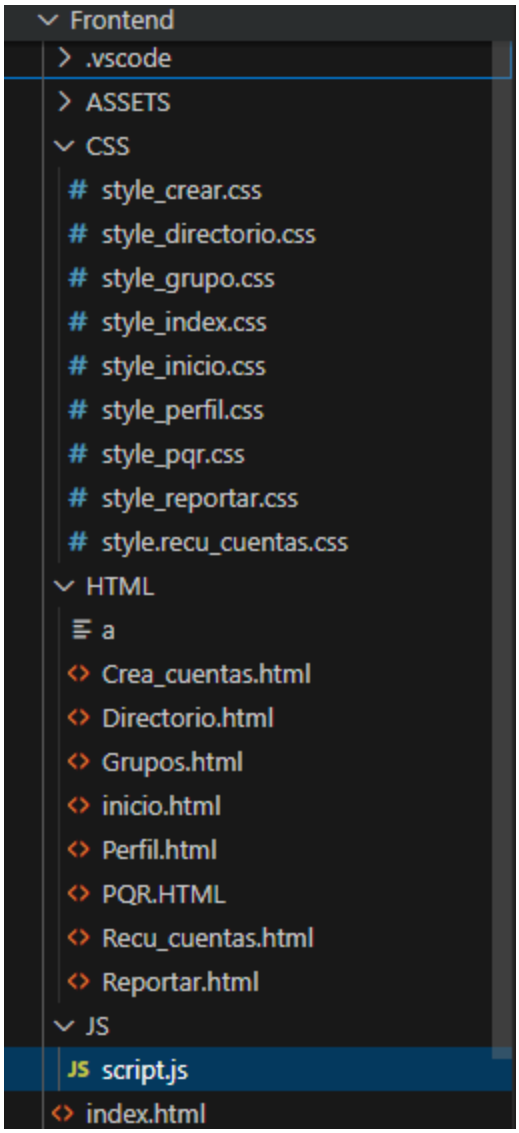
Introducción

El presente Manual Técnico tiene como objetivo proporcionar una guía detallada sobre la arquitectura, componentes, funcionamiento y mantenimiento de la aplicación **SafeHood360**. Esta herramienta ha sido diseñada para fortalecer la seguridad comunitaria mediante una plataforma digital que permite a los usuarios reportar incidentes, recibir alertas en tiempo real y acceder a recursos de ayuda de manera rápida y eficiente.

Manual Técnico - SafeHood360 Frontend

Para la construcción de la estructura del Frontend se emplean principalmente las tecnologías HTML, CSS y JavaScript. Estas herramientas son fundamentales para crear una interfaz que no solo sea funcional, sino que también resulte sencilla de usar, intuitiva en su navegación y visualmente atractiva para el usuario final. HTML se encarga de definir la estructura y el contenido básico de la página web, mientras que CSS permite diseñar y estilizar los elementos para mejorar su apariencia, aportando colores, tipografías, espacios y distribuciones que facilitan la experiencia visual. Por su parte, JavaScript añade dinamismo y funcionalidades interactivas que hacen que la interfaz responda a las acciones del usuario (Pinto, 2025)

Repositorio Frontend: https://github.com/EdwardgG97/SafeHood360-Frontend



Index.html

En esta sección del código del *index* se construye una interfaz dividida en dos partes: a la izquierda se muestra el logotipo, un eslogan y un enlace a la sección de PQRS; mientras que a la derecha se encuentra el formulario de inicio de sesión, con campos para usuario y contraseña, un enlace para recuperar la cuenta, otro para crear una nueva, y un botón para iniciar sesión. (Pinto, 2025)

```
<body>
  <div class="container">
    <div class="left">
      <div class="logo">🔰 <span>SafeHood360</span></div>

      <div class="left-content">
        <h1>Together for a <br><span>safer hood</span></h1>
      </div>

      <div class="pqrs">
        <a href="HTML/PQR.HTML">PQRS</a>
      </div>
    </div>
    <div class="right">
      <div class="login-box">
        <input type="text" placeholder="User">
        <input type="password" placeholder="Pass">
        <a href="HTML/Recu_cuentas.html" class="forgot">¿Ha olvidado su contraseña?</a>
        <button class="login-btn" onclick="window.location.href='HTML/inicio.html'">Iniciar sesión</button>
        <hr>
        <a href="HTML/Crea_cuentas.html" class="register-btn">Nueva cuenta</a>
      </div>
```

Style_index.css

Este archivo CSS da forma y estilo a la página de inicio de sesión de **SafeHood360**, dividiendo la pantalla en dos secciones con diseño responsivo y moderno. La **sección izquierda** presenta el logotipo, un eslogan atractivo y un enlace a PQRS, todo con colores claros y una estructura centrada. Adicional muestra un fondo con un degradado azul y contiene una caja de login con campos para ingresar usuario y contraseña (Pinto, 2025)

```css
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

body, html {
  height: 100%;
  font-family: 'Segoe UI', sans-serif;
}

.container {
  display: flex;
  height: 100vh;
  width: 100%;
}

.left {
  width: 50%;
  background-color: #ffffff;
  padding: 40px 60px;
  display: flex;
  flex-direction: column;
  position: relative;
}

.logo {
  font-size: 24px;
  font-weight: bold;
  color: #120078;
  display: flex;
  align-items: center;
```

Crea_cuentas.html

Presentamos una interfaz clara e intuitiva que permite a los usuarios registrarse proporcionando datos personales como nombre, correo, dirección, número de contacto, y barrios de residencia. También incluye la opción de subir una imagen de perfil. El formulario es accesible y está organizado en filas para facilitar su llenado (Pinto, 2025)

```
<!-- Formulario de registro -->
<main class="form-container">
  <form class="registro-form">
    <h2>Crea una cuenta</h2>

    <!-- Subida de imagen de perfil -->
    <div class="profile-upload">
      <div class="avatar">👤</div>
      <label for="upload" class="upload-label">⬆ Cargar imagen</label>
      <input type="file" id="upload" hidden>
    </div>

    <!-- Campos de entrada -->
    <div class="input-group">
      <div class="form-row">
        <label for="nombres">Nombres:</label>
        <input type="text" id="nombres" required>

        <label for="apellidos">Apellidos:</label>
        <input type="text" id="apellidos" required>
      </div>

      <div class="form-row">
        <label for="correo">Correo:</label>
        <input type="email" id="correo" required>

        <label for="contrasena">Contraseña:</label>
        <input type="password" id="contrasena" required>
      </div>
```

Style_crear.css

Utilizando una estructura moderna, limpia y responsive basada en Flexbox y Grid. Se eliminan los márgenes y rellenos por defecto para garantizar un diseño consistente. El encabezado destaca el logotipo con un diseño llamativo en color azul (Pinto, 2025)

```css
/* CAMPOS DE ENTRADA */
.input-group {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(220px, 1fr));
  gap: 15px;
  margin-bottom: 25px;
}

.input-group label {
  display: block;
  text-align: left;
  margin-bottom: 5px;
  font-weight: 500;
}

.input-group input {
  width: 100%;
  padding: 10px;
  background-color: #eae3eb;
  border: 1px solid #ccc;
  border-radius: 4px;
  font-size: 14px;
}

/* BOTÓN DE ENVÍO */
.submit-btn {
  background-color: #2ecc71;
  color: white;
  padding: 12px 24px;
```

JS

Implementamos una validación simple en el formulario de registro, evitando que se envíe si alguno de los campos de texto, correo electrónico, contraseña, teléfono o fecha está vacío. Al intentar enviar el formulario, se detiene el envío automático y se verifica que todos los campos requeridos estén completos (Pinto, 2025)

```
// REGISTRO
const registroForm = document.querySelector('.registro-form');
if (registroForm) {
  registroForm.addEventListener('submit', (e) => {
    e.preventDefault();
    const inputs = registroForm.querySelectorAll(
      'input[type="text"], input[type="email"], input[type="password"], input[type="tel"], input[type="date"]'
    );
    let camposVacios = false;

    inputs.forEach(input => {
      if (input.value.trim() === '') {
        camposVacios = true;
      }
    });

    if (camposVacios) {
      alert('⚠️ Por favor, completa todos los campos antes de registrarte.');
    } else {
      alert('✅ Registro completo. (Aquí iría la lógica real)');
    }
  });
}
```

Recu_cuentas.html

Se presenta un diseño sencillo y claro, con un encabezado que muestra el logo, un mensaje explicativo, un campo de entrada para el contacto (email o teléfono), un botón para enviar la solicitud y un enlace para regresar a la página principal (Pinto, 2025)

```
<main class="container">
  <div class="recovery-box">
    <h1>Recuperar Cuenta</h1>
    <p>
      Ingresa tu correo electrónico para enviar el enlace de recuperación de contraseña o
      ingresa tu número de celular para enviar el enlace por SMS
    </p>
    <hr>
    <label for="contact">email / phone</label>
    <input type="text" id="contact" placeholder="ej. correo@dominio.com o 3001234567" />
    <button id="sendBtn">Enviar</button>
    <a href="../index.html">Regresar</a>
  </div>
</main>
```

Style.recu_cuentas.css

El cuerpo usa una fuente clara y fondo blanco para una apariencia sencilla. El contenedor principal tiene un ancho limitado y está centrado con un fondo azul claro, bordes redondeados y un padding amplio para destacar el contenido (Pinto, 2025)

```css
input[type="text"], input[type="email"], input[type="tel"] {
  width: 100%;
  padding: 12px;
  margin: 8px 0 20px 0;
  border: 1px solid #ccc;
  border-radius: 4px;
  background-color: #ede6f0;
}

#sendBtn {
  background-color: #2e8b57;
  color: white;
  border: none;
  padding: 12px 24px;
  border-radius: 20px;
  font-size: 16px;
  cursor: pointer;
}

#sendBtn:hover {
  background-color: #246b45;
}
```

JS

Cuando el usuario hace clic, se obtiene el valor del campo de entrada para correo o teléfono, y se verifica si está vacío. Si el campo está vacío, muestra una alerta pidiendo que se ingrese el dato; si está lleno, muestra una alerta confirmando que el enlace de recuperación fue enviado

```
// RECUPERACIÓN DE CUENTA
const sendBtn = document.getElementById('sendBtn');
if (sendBtn) {
  sendBtn.addEventListener('click', () => {
    const input = document.getElementById('contact');
    const value = input?.value.trim() || '';

    if (!value) {
      alert('⚠️ Por favor, ingresa tu correo o número de teléfono.');
    } else {
      alert('✅ Enlace de recuperación enviado (simulado).');
    }
  });
}
```

PQR.html

```
<form id="pqrsForm">
  <input type="email" placeholder="Correo electrónico" required>
  <textarea rows="5" placeholder="Escribe tu mensaje aquí..." required></textarea>

  <div class="success-message" id="successMessage">
    Tu PQRS será gestionado y te daremos respuesta al correo registrado
    <span class="close-btn" onclick="cerrarMensaje()">x</span>
  </div>

  <button type="submit">Enviar</button>
  <button type="button" onclick="window.location.href='../index.html'">Regresar</button>

</form>
</main>
```

Style_pqr.css

Utilizamos un fondo azul claro que brinda sensación de confianza, con un contenedor centrado y un área de formulario blanca y redondeada que mejora la legibilidad. Los encabezados destacan en azul y los textos están bien espaciados. (Pinto, 2025)

```css
input[type="email"],
textarea {
  width: 100%;
  padding: 10px;
  margin: 10px 0;
  border-radius: 6px;
  border: 1px solid #ccc;
  font-size: 14px;
  background-color: #f4f4f4;
}

button {
  background-color: #2c9f5f;
  color: white;
  border: none;
  padding: 12px 30px;
  font-size: 16px;
  border-radius: 30px;
  cursor: pointer;
}

button:hover {
  background-color: #21894e;
}
```

JS

Este fragmento de código JavaScript agrega funcionalidad al formulario PQRS para mostrar un mensaje de éxito al enviarlo (Pinto, 2025)

```
// PQRS FORMULARIO
const pqrsForm = document.getElementById("pqrsForm");
if (pqrsForm) {
  pqrsForm.addEventListener("submit", function(event) {
    event.preventDefault();
    const successMsg = document.getElementById("successMessage");
    if (successMsg) {
      successMsg.style.display = "block";
    }
  });
}
```

Inicio.html

Este código HTML crea una página web tipo "dashboard" para la aplicación SafeHood360, diseñada para mostrar información relacionada con la seguridad de una comunidad. Incluye una barra lateral con un menú de navegación y un botón para cerrar sesión, y en el área principal muestra un título, un gráfico (que se genera con Chart.js), y una tabla con reportes recientes de incidentes, como robos o emergencias médicas, indicando hora, barrio y estado del caso. También incluye enlaces a redes sociales y carga estilos y scripts externos para darle funcionalidad e interactividad. (Pinto, 2025)

```
<div class="container">
  <!-- Sidebar -->
  <div class="sidebar">
    <div class="top-section">
      <div class="logo">
        <span>SafeHood360</span>
      </div>
      <div class="menu">
        <h3>Menú</h3>
        <a href="../HTML/inicio.html">Inicio</a>
        <a href="../HTML/Perfil.html">Perfil</a>
        <a href="../HTML/Reportar.html">Reportar</a>
        <a href="../HTML/Grupos.html">Grupos</a>
        <a href="../HTML/Directorio.html">Directorios</a>
      </div>
    </div>
    <div class="logout">
        <a href="../index.html">Cerrar sesión</a>
    </div>
  </div>
```

```
<main class="dashboard">
  <h1 class="titulo">Dashboard</h1>
  <h3 class="subtitulo">Reporte Anual:</h3>
  <canvas id="grafico" width="400" height="150"></canvas>

  <h3 class="subtitulo">Llamadas del día:</h3>
  <table>
    <thead>
      <tr>
        <th>Hora llamada</th>
        <th>Barrio</th>
        <th>Tipo de incidente</th>
        <th>Estado</th>
      </tr>
    </thead>
    <tbody>
      <tr><td>11:35 a. m.</td><td>Modelia</td><td>Robo</td><td>En curso</td></tr>
      <tr><td>9:15 a. m.</td><td>Hogares Soacha</td><td>Sospechoso</td><td>Resuelto</td></tr>
      <tr><td>6:28 a. m.</td><td>La prosperidad</td><td>Emergencia Médica</td><td>Pendiente</td></tr>
      <tr><td>1:29 p. m.</td><td>La prosperidad</td><td>Ruido</td><td>Enviado al CAI</td></tr>
      <tr><td>6:49 a. m.</td><td>Modelia</td><td>Sospechoso</td><td>Pendiente</td></tr>
      <tr><td>10:28 a. m.</td><td>Hogares Soacha</td><td>Robo</td><td>En curso</td></tr>
    </tbody>
  </table>
```

Style_incio.css

La barra lateral contiene navegación con efectos al pasar el mouse, y un botón de cierre de sesión estilizado. El área principal tiene títulos, una tabla de reportes, botones para acciones (como

reportar o cargar evidencia), y un gráfico. Además, se incluyen estilos para modales, mensajes de éxito, y íconos sociales animados. (Pinto, 2025)

```css
.logout a:hover {
  background-color: #c0392b;
}

/* Contenido principal */
.main-content, .dashboard {
  flex: 1;
  padding: 40px;
  text-align: center;
  background-color: white;
  border-left: 10px solid #d0e1ff;
}

.titulo {
  color: #273c75;
  font-size: 28px;
  margin-bottom: 10px;
}

.subtitulo {
  color: #f39c12;
  margin-top: 25px;
  margin-bottom: 10px;
}
```

JS

Este código busca un elemento con el ID gráfico y, si existe, crea un gráfico de línea en él utilizando la librería Chart.js. El gráfico muestra la cantidad de incidentes registrados por mes, desde enero hasta octubre, usando un conjunto de datos con valores numéricos (Pinto, 2025)

```
const graficoCanvas = document.getElementById('grafico');
if (graficoCanvas) {
  const ctx = graficoCanvas.getContext('2d');
  new Chart(ctx, {
    type: 'line',
    data: {
      labels: ['Enero', 'Febrero', 'Marzo', 'Abril', 'Mayo', 'Junio', 'Julio', 'Agosto', 'Septiembre', 'Octubr
      datasets: [{
        label: 'Incidentes',
        data: [20, 45, 30, 25, 15, 18, 22, 30, 55, 70],
        fill: false,
        borderColor: '#3498db',
        tension: 0.3
      }]
    },
    options: {
      responsive: true,
      plugins: {
        legend: {
          display: false
        }
      }
    }
  });
}
});
```

Perfil.html

Este código HTML representa la página de perfil de usuario de la aplicación SafeHood360. Muestra la información personal del usuario como nombre, dirección, correo, etc.) dentro de una tarjeta de perfil editable, aunque no tiene funcionalidad real de guardado aún el botón "Actualizar" no está vinculado a ningún script. (Pinto, 2025)

```
<div class="profile-fields">
  <div class="field">
    <label>Nombres:</label>
    <input type="text" value="Edward Alejandro">
  </div>
  <div class="field">
    <label>Apellidos:</label>
    <input type="text" value="Garcia Gonzalez">
  </div>
  <div class="field">
    <label>Sexo:</label>
    <input type="text" value="Masculino">
  </div>
  <div class="field">
    <label>Dirección:</label>
    <input type="text" value="Tv 7 # 4 A - 83">
  </div>
  <div class="field">
    <label>Correo:</label>
    <input type="email" value="egarci71@estudiante.ibero">
  </div>
  <div class="field">
    <label>Contraseña:</label>
    <input type="password" value="********">
  </div>
  <div class="field">
    <label>Celular:</label>
```

```
    </div>
    <div class="field">
      <label>Celular:</label>
      <input type="tel" value="3015820854">
    </div>
    <div class="field">
      <label>Fecha Nacimiento:</label>
      <input type="date" value="1997-01-12">
    </div>
  </div>

  <div class="buttons">
    <button class="btn-update">Actualizar</button>
    <button class="btn-update" onclick="window.location.href='../HTML/inicio.html'">Volver</button>
  </div>
 </div>
</main>
</div>
```

```
<!-- Contenido principal -->
<main class="profile-content">
  <div class="profile-card">
    <h2>Perfil</h2>
    <div class="profile-img">
      <div class="avatar">👤</div>
    </div>

    <div class="profile-fields">
      <div class="field">
        <label>Nombres:</label>
        <input type="text" value="Edward Alejandro">
      </div>
      <div class="field">
        <label>Apellidos:</label>
        <input type="text" value="Garcia Gonzalez">
      </div>
      <div class="field">
        <label>Sexo:</label>
        <input type="text" value="Masculino">
      </div>
      <div class="field">
        <label>Dirección:</label>
        <input type="text" value="Tv 7 # 4 A - 83">
      </div>
      <div class="field">
```

```
      <div class="field">
        <label>Correo:</label>
        <input type="email" value="egarci71@estudiante.ibero">
      </div>
      <div class="field">
        <label>Contraseña:</label>
        <input type="password" value="********">
      </div>
      <div class="field">
        <label>Celular:</label>
        <input type="tel" value="3015820854">
      </div>
      <div class="field">
        <label>Fecha Nacimiento:</label>
        <input type="date" value="1997-01-12">
      </div>
    </div>

    <div class="buttons">
      <button class="btn-update">Actualizar</button>
      <button class="btn-update" onclick="window.location.href='../HTML/inicio.html'">Volver</button>
    </div>
  </div>
</main>
</div>
```

Style_perfil.css

La tarjeta incluye campos personales como nombre, correo, dirección, etc., con inputs estilizados, botones verdes para acciones como "Actualizar" o "Volver", y una animación de entrada suave (fadeInUp). También se aplica un diseño responsivo: en pantallas pequeñas, los campos se reorganizan en una sola columna y se ocultan el menú (Pinto, 2025)

```css
.profile-card {
  width: 100%;
  max-width: 750px;
  box-shadow: 0 8px 16px rgba(0, 0, 0, 0.1);
  animation: fadeInUp 0.6s ease;
}

@keyframes fadeInUp {
  from {
    opacity: 0;
    transform: translateY(20px);
  }
  to {
    opacity: 1;
    transform: translateY(0);
  }
}

.profile-card h2 {
  text-align: center;
  color: #120078;
  margin-bottom: 25px;
  font-size: 28px;
  font-weight: 700;
}

/* Imagen de perfil (opcional) */
.profile-img {
  display: flex;
  justify-content: center;
  margin-bottom: 25px;
}
```

Reportar.html

Este archivo HTML representa la **página de reportes de incidentes** de la aplicación **SafeHood360**. Permite a los usuarios ubicar en el mapa (mediante un iframe de Google Maps centrado en Soacha) el lugar del incidente y acceder a un **modal interactivo** para detallar lo ocurrido. Dentro del modal, el usuario puede escribir una descripción del evento, cargar evidencia (como fotos), y enviar el reporte mediante un botón. (Pinto, 2025)

```
      <!-- Aquí puedes insertar el iframe de Google Maps o usar Leaflet.js -->
      <iframe
      src="https://maps.google.com/maps?q=Soacha&t=&z=14&ie=UTF8&iwloc=&output=embed"
      width="100%" height="350" style="border:0;" allowfullscreen="" loading="lazy"></iframe>

    </div>
    <button class="btn-cargar" onclick="abrirModal()">Cargar</button>
  </main>
</div>

<!-- Modal -->
<div id="modal" class="modal hidden">
  <div class="modal-content">
    <span class="cerrar" onclick="cerrarModal()">&times;</span>
    <h3>¡¡Lea atentamente!!</h3>
    <p>Por favor ingrese con el mayor detalle posible. Importar fotos del evento nos ayuda a tener mejor evide
    <label class="cargar-evidencia">📤 Cargar evidencia
      <input type="file" hidden>
    </label>
    <textarea placeholder="¿Qué sucedió?"></textarea>
    <div id="exito" class="exito hidden">✅ ¡Envío exitoso!</div>
    <button class="btn-enviar" onclick="enviarReporte()">Enviar</button>
  </div>
</div>
```

Style_reportar.html

El mapa está embebido con bordes suaves y un diseño centrado. Al hacer clic en "Cargar", se despliega un **modal flotante** donde el usuario puede escribir una descripción del incidente, adjuntar evidencia y enviarla. Se usan colores estratégicos: rojo para acciones urgentes (como reportar) (Pinto, 2025)

```css
.logout a:hover {
  background-color: #c0392b;
}

.main-content {
  flex: 1;
  padding: 40px;
  text-align: center;
}

.titulo {
  color: #c0392b;
  font-size: 26px;
  margin-bottom: 20px;
}

#mapa {
  border: 2px solid #ccc;
  border-radius: 8px;
  overflow: hidden;
  margin-bottom: 20px;
}

.btn-cargar {
  padding: 12px 40px;
  background-color: #c0392b;
  color: white;
```

JS

El código JavaScript permite abrir, cerrar y enviar un formulario de reporte en un modal, validando que el mensaje no esté vacío y mostrando una confirmación temporal de éxito. Además, incluye una función de búsqueda en una lista de contactos que filtra los resultados (Pinto, 2025)

```
function cerrarMensaje() {
  const msg = document.getElementById("successMessage");
  if (msg) {
    msg.style.display = "none";
  }
}

function abrirModal() {
  const modal = document.getElementById('modal');
  if (modal) modal.classList.remove('hidden');
}

function cerrarModal() {
  const modal = document.getElementById('modal');
  const exito = document.getElementById('exito');
  if (modal) modal.classList.add('hidden');
  if (exito) exito.classList.add('hidden');
}
```

Grupos.html

Este archivo HTML representa la página de **Grupos** del sitio *SafeHood360*, mostrando una interfaz con barra lateral de navegación, botones para seleccionar grupos vecinales por barrio y un área donde se cargan dinámicamente mensajes relacionados con cada grupo

```
<main class="main-content">
  <h1 class="titulo">Grupos</h1>
  <div class="botones">
    <button onclick="mostrarMensajes(1)">Barrio 1</
    <button onclick="mostrarMensajes(2)">Barrio 2</
    <button onclick="mostrarMensajes(3)">Barrio 3</button>
  </div>
  <div class="mensajes" id="mensajes">
    <!-- Mensajes del grupo se insertan dinámicamente -->
  </div>
  <div class="redes">
    <a href="#"><img src="https://img.icons8.com/ios-glyphs/30/facebook-new.png" alt="Facebook" /></a>
    <a href="#"><img src="https://img.icons8.com/ios-glyphs/30/twitter.png" alt="Twitter" /></a>
    <a href="#"><img src="https://img.icons8.com/ios-glyphs/30/instagram-new.png" alt="Instagram" /></a>
  </div>
</main>
</div>
```

The button element represents a button labeled by its contents.

Widely available across major browsers (Baseline since 2015)

MDN Reference

Style_grupos.cs

Una área principal clara donde se visualizan los grupos y mensajes. Se emplean botones estilizados para seleccionar barrios, una sección de mensajes con sombra y esquinas redondeadas, y enlaces a redes sociales (Pinto, 2025)

```
.botones button:hover {
  background-color: #2740c6;
}

.mensajes {
  background-color: #f5f5f5;
  border-radius: 10px;
  padding: 20px;
  min-height: 200px;
  box-shadow: 0 2px 6px rgba(0,0,0,0.1);
}

.mensajes p {
  margin-bottom: 15px;
  font-size: 15px;
  line-height: 1.5;
}
```

Directorio.html

La página "Directorio" de SafeHood360 permite a los usuarios acceder de forma rápida y sencilla a contactos clave como emergencias, bomberos, CAIs y otros números comunitarios. Incluye una barra lateral de navegación, un buscador para filtrar contactos y una lista de opciones con botones de selección. (Pinto, 2025)

```
<!-- Contenido principal -->
<div class="main-content">
  <h2 class="titulo">Directorio</h2>

  <div class="buscador">
    <input type="text" id="busqueda" placeholder="Buscar contacto">
    <button onclick="buscarContacto()">🔍</button>
  </div>

  <ul id="lista-contactos" class="lista-contactos">
    <li><input type="radio" name="telefono" value="123"> Emergencias Nacional: 123</li>
    <li><input type="radio" name="telefono" value="3218000593"> Bomberos: 3218000593</li>
    <li><input type="radio" name="telefono" value="3019858216"> CAI Barrio 1: 3019858216</li>
    <li><input type="radio" name="telefono" value="3208557490"> CAI Barrio 2: 3208557490</li>
    <li><input type="radio" name="telefono" value="3108558115"> CAI Barrio 3: 3108558115</li>
    <li><input type="radio" name="telefono" value="3216540874"> Contacto 1: 3216540874</li>
    <li><input type="radio" name="telefono" value="3165489879"> Contacto 2: 3165489879</li>
    <li><input type="radio" name="telefono" value="3018588050"> Contacto 3: 3018588050</li>
  </ul>
```

Style_directorios

La interfaz incluye un buscador de contactos con estilo limpio y botones integrados, una lista ordenada de números telefónicos, un ícono animado de llamada 📞 para simular la acción de llamar, y un mensaje visual de confirmación (Pinto, 2025)

```css
.lista-contactos {
  list-style: none;
  padding: 0;
  margin: 0 auto;
  width: 80%;
}

.lista-contactos li {
  padding: 10px;
  border-bottom: 1px solid #ccc;
}

.icono-llamada {
  font-size: 30px;
  text-align: center;
  margin: 20px;
  cursor: pointer;
  transition: color 0.3s, transform 0.3s;
}

.icono-llamada.llamando {
  color: green;
  animation: parpadeo 0.8s infinite;
  transform: scale(1.2);
}
```

JS

Esta función simularLlamada() permite simular una llamada telefónica desde el directorio de contactos. Primero verifica si el usuario ha seleccionado un número (radio button). Si lo hizo, activa una animación visual en el ícono de llamada y muestra el mensaje

```
function simularLlamada() {
  const seleccionado = document.querySelector('input[name="telefono"]:checked');
  const icono = document.querySelector('.icono-llamada');
  const mensaje = document.getElementById('mensaje-llamada');

  if (seleccionado) {
    const numero = seleccionado.value;

    // Animar ícono y mostrar mensaje
    icono.classList.add('llamando');
    mensaje.classList.remove('oculto');

    // Esperar 2 segundos y hacer la llamada
    setTimeout(() => {
      icono.classList.remove('llamando');
      mensaje.classList.add('oculto');
      window.location.href = `tel:${numero}`;
    }, 2000);
  } else {
    alert("⚠️ Por favor, selecciona un número para llamar.");
  }
}
```

Manual Técnico - SafeHood360 Backend

El proyecto Backend se desarrollo en Java 17 con framework SpringBoot, se debe solicitar acceso al repositorio del Backend, haciendo uso de herramienta de versiones Git. Se sugiere usar IntelliJ IDEA para el desarrollo local.

Repositorio Backend: https://github.com/EdwardgG97/SafeHood360-Backend

1. Tecnologías Principales

- **Framework**: Spring Boot 3.5.0
- **Lenguaje**: Java 17
- **Base de Datos**: MySQL 8
- **ORM**: JPA/Hibernate

- **Desarrollo**: Lombok para reducción de boilerplate code

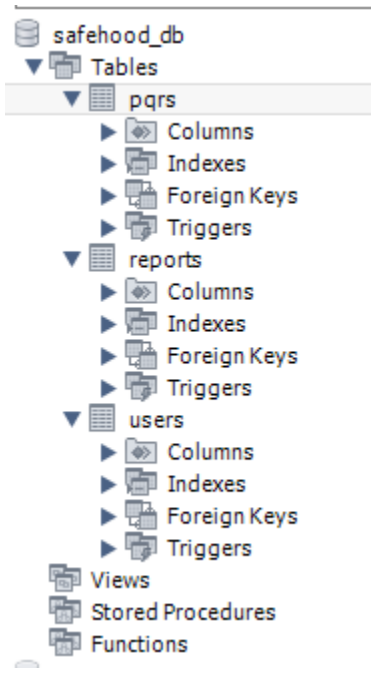
2. Arquitectura

2.1 Estructura del Proyecto

- **Spring Boot Starter**:
    - spring-boot-starter-web: Para REST APIs
    - spring-boot-starter-data-jpa: Para persistencia de datos
    - spring-boot-starter-test: Para pruebas unitarias

2.2 Configuración del Servidor

- **Puerto**: 8080
- **Base de Datos**:



    - URL: jdbc:mysql://127.0.0.1:3306/safehood_db

- Usuario: root
- Contraseña: admin
- Dialecto: MySQL8Dialect

3. Características Técnicas

3.1 Persistencia de Datos

- JPA/Hibernate con configuración automática (ddl-auto=update)
- Inicialización de datos a través de data.sql
- Logging detallado de SQL (DEBUG/TRACE)

3.2 Seguridad

- SSL deshabilitado para la conexión a la base de datos
- Zona horaria UTC para consistencia

3.3 Logging

- Niveles de logging configurados para:
  - SQL: DEBUG
  - Hibernate: TRACE
  - JDBC: DEBUG

4. Requisitos del Sistema

4.1 Dependencias

- Java 17 JDK
- MySQL 8
- Maven 3.6.3 o superior

4.2 Configuración del Entorno

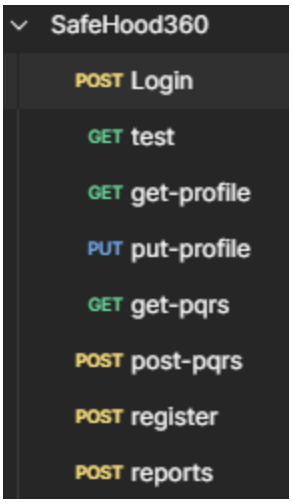
- Puerto 8080 disponible
- Base de datos MySQL configurada
- Variables de entorno para credenciales de la base de datos

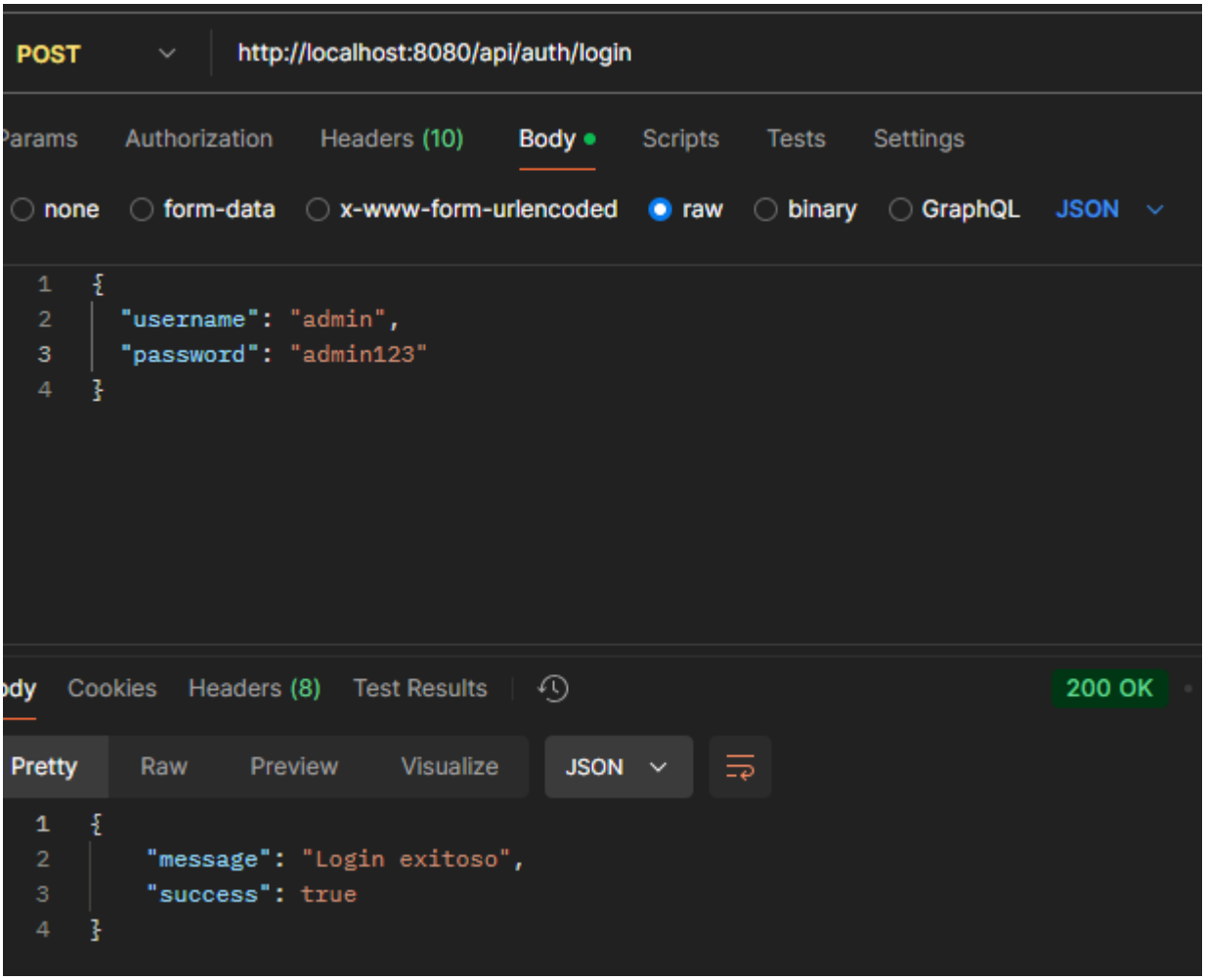
5. Estructura del Código

El proyecto sigue la estructura estándar de Spring Boot con:

- Paquete principal: com.safehood
- Recursos en: src/main/resources
- Configuración en: application.properties

Dando un vistazo a la estructura del proyecto encontraremos un archivo **“SafeHood360.postman_collection.json”** donde podremos importar este archivo a la aplicación Postman, donde tendremos unos request a la API en localhost, mostrando su correcta funcionalidad.


SafeHood360
POST Login
GET test
GET get-profile
PUT put-profile
GET get-pqrs
POST post-pqrs
POST register
POST reports


POST
http://localhost:8080/api/auth/login
Params
Authorization
Headers (10)
Body
Scripts
Tests
Settings
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
{
"username": "admin",
"password": "admin123"
}
ody
Cookies
Headers (8)
Test Results
200 OK
Pretty
Raw
Preview
Visualize
JSON
{
"message": "Login exitoso",
"success": true
}

El proyecto Backend Se utiliza una arquitectura **Hexagonal** (también conocida como Ports and Adapters) con una clara separación de responsabilidades. Vamos a examinar algunos componentes clave para confirmarlo:

1. **Capa de Controlador (Controller)**:

- Maneja las peticiones HTTP
- Actúa como adaptador hacia el exterior

2. **Capa de Servicio (Service)**:

- Contiene la lógica de negocio
- Implementa los casos de uso del sistema

3. **Capa de Modelo (Model)**:

- Entidades del dominio
- Mapeo objeto-relacional (ORM)

4. **Capa de Repositorio (Repository)**:

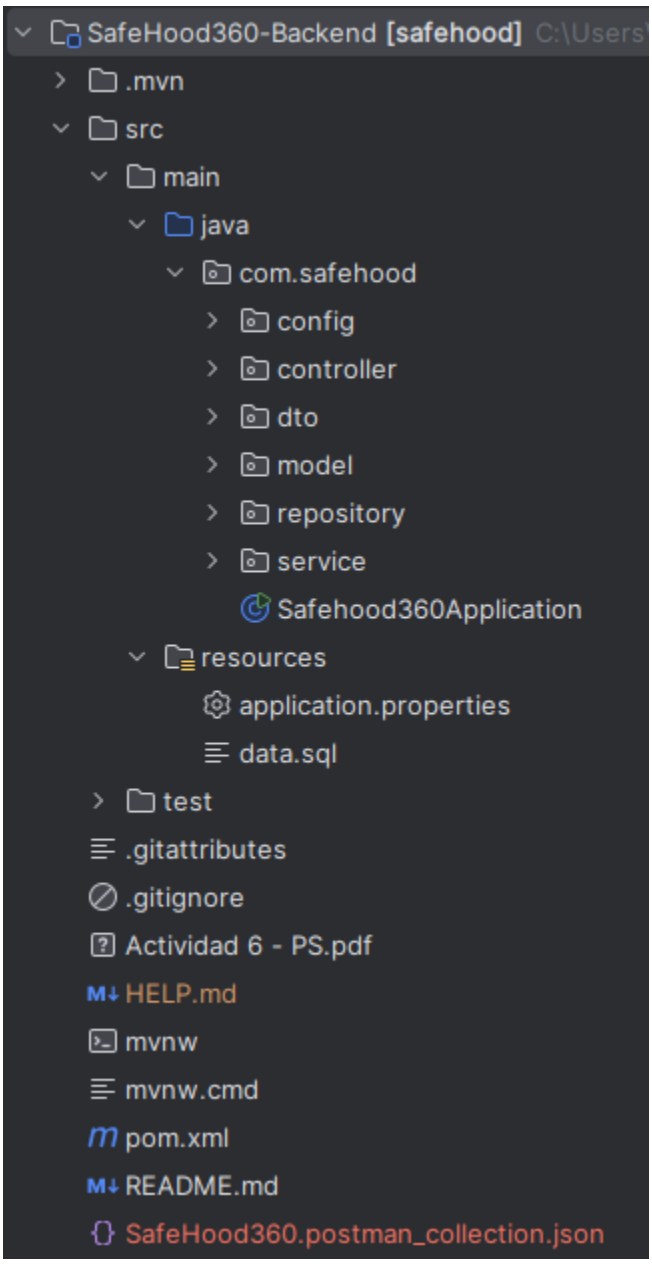
- Acceso a datos
- Interfaz con la base de datos

5. **DTOs (Data Transfer Objects)**:

- Transferencia de datos entre capas
- Separación de la representación interna y externa

Esta estructura sigue los principios del **Hexagonal Architecture**:

- **Puertos**: Interfaces que definen cómo el sistema interactúa con el exterior
- **Adaptadores**: Implementaciones concretas que conectan los puertos con el mundo exterior
- **Núcleo del Dominio**: Lógica de negocio centralizada en los servicios y modelos



Links a Consultar

https://spring.io/projects/spring-boot

https://maven.apache.org/

https://www.postman.com/

https://developer.mozilla.org/es/docs/Web/HTTP/Reference/Status

https://www.ibm.com/es-es/think/topics/rest-apis

https://developer.mozilla.org/es/docs/Web/HTML

https://developer.mozilla.org/es/docs/Learn_web_development/Core/Scripting/What_is_JavaScript

https://www.w3schools.com/Css/

https://www.mysql.com/