1. Introducción y Objetivos del Proyecto

- **Descripción:**
  Se desarrollará una aplicación web utilizando Node.js, React y MongoDB, que se alinee con el Manifiesto Anthem y aproveche el dataset provisto. La aplicación pretende mejorar la forma en que las personas viven y trabajan en la ciudad, optimizando la movilidad, la sostenibilidad y la seguridad a través del análisis y visualización de datos urbanos (tráfico, calidad del aire, contaminación acústica, etc.).
- **Objetivos:**
  - Mejorar la movilidad inteligente y la gestión urbana.
  - Monitorear en tiempo real datos ambientales y de tráfico.
  - Proveer alertas y análisis predictivo para la toma de decisiones.
  - Integrar múltiples fuentes de datos para generar insights útiles para ciudadanos, administradores y analistas.

---

2. Alcance del Proyecto

- **Funcionalidades Principales:**
  - **Gestión de Usuarios:** Registro, login y roles (administrador, analista y ciudadano).
  - **Dashboards Interactivos:** Visualización de indicadores clave (gráficos, estadísticas) filtrables por fecha y sensor.
  - **Mapas Interactivos:** Geolocalización de sensores y puntos de interés, con información detallada al hacer clic.
  - **Análisis Predictivo y Alertas:** Modelos básicos para identificar patrones anómalos en tráfico, calidad del aire y otros parámetros.
  - **API REST:** Endpoints para la consulta, actualización y procesamiento de datos en MongoDB.

---

3. Requerimientos del Sistema

- **Requerimientos Funcionales:**
  - Autenticación y gestión de usuarios.
  - Visualización y filtrado de datos en dashboards.

- Módulo de mapas interactivos con filtros por tipo de sensor.
- Generación y gestión de alertas en tiempo real.
- API para el manejo y consulta de datos (sensores, alertas, usuarios).

- **Requerimientos No Funcionales:**
  - Seguridad en la autenticación y en la transmisión de datos.
  - Alta disponibilidad y rendimiento, considerando el volumen de datos.
  - Escalabilidad para incorporar nuevos sensores o funcionalidades.
  - Interfaz intuitiva y responsiva para dispositivos móviles y de escritorio.

---

4. Casos de Uso y Historias de Usuario

- **Caso de Uso 1: Autenticación de Usuario**
  - **Actor:** Usuario (Ciudadano, Administrador, Analista)
  - **Precondición:** El usuario debe estar registrado en el sistema.
  - **Flujo Principal:**
    1. El usuario ingresa sus credenciales en la pantalla de login.
    2. El sistema valida las credenciales y, en caso de éxito, redirige al dashboard principal.
  - **Flujo Alternativo:**
    - Credenciales incorrectas → Mensaje de error y opción para reintentar.

- **Historia de Usuario 1:**
  Como ciudadano, quiero iniciar sesión en la aplicación para acceder a la información en tiempo real sobre la movilidad y el estado de la ciudad.
  Criterios de Aceptación:
  - La pantalla de login es clara y permite ingresar usuario y contraseña.
  - Un mensaje de error se muestra en caso de credenciales incorrectas.
  - Al ingresar datos válidos, se redirige correctamente al dashboard.

- **Caso de Uso 2: Consulta de Dashboard**
    - **Actor:** Usuario (Ciudadano, Analista)
    - **Precondición:** Usuario autenticado.
    - **Flujo Principal:**
        1. El usuario ingresa al dashboard desde el menú principal.
        2. Se muestran gráficos y estadísticas interactivas (por ejemplo, datos de tráfico y calidad del aire).
        3. El usuario aplica filtros (por fecha, tipo de sensor, etc.) para personalizar la visualización.
    - **Flujo Alternativo:**
        - El usuario modifica los filtros y el sistema actualiza los gráficos en tiempo real.
- **Historia de Usuario 2:**
  Como analista, quiero visualizar en un dashboard los indicadores de tráfico y calidad del aire, para detectar patrones y tomar decisiones basadas en datos.
  Criterios de Aceptación:
    - El dashboard muestra datos actualizados y permite aplicar filtros por fecha y sensor.
    - Los gráficos son interactivos y permiten hacer zoom o seleccionar rangos específicos.

- **Caso de Uso 3: Visualización de Datos Geolocalizados**
    - **Actor:** Usuario (Ciudadano)
    - **Precondición:** Usuario autenticado.
    - **Flujo Principal:**
        1. El usuario accede a la vista de mapa interactivo.
        2. El sistema muestra un mapa con marcadores correspondientes a diferentes sensores (tráfico, contaminación, etc.).
        3. Al hacer clic en un marcador, se muestra un panel con detalles del sensor y datos históricos.

- **Flujo Alternativo:**
    - El usuario filtra los sensores por tipo para ver únicamente aquellos de interés.

- **Historia de Usuario 3:**
  Como ciudadano, quiero ver en un mapa interactivo la ubicación de sensores urbanos para comprender la situación de mi entorno en tiempo real.
  Criterios de Aceptación:
    - El mapa muestra marcadores con distintos íconos según el tipo de sensor.
    - Al seleccionar un marcador, se despliega información detallada del sensor.

- **Caso de Uso 4: Gestión de Alertas y Notificaciones**
    - **Actor:** Usuario (Administrador, Analista)
    - **Precondición:** El sistema ha detectado una anomalía o patrón fuera de lo normal.
    - **Flujo Principal:**
        1. El backend procesa los datos y genera alertas según reglas predefinidas.
        2. El sistema notifica a los usuarios pertinentes (por ejemplo, al administrador) mediante la interfaz o correos electrónicos.
        3. El usuario revisa y gestiona la alerta, pudiendo marcarla como "en proceso" o "resuelta".
    - **Flujo Alternativo:**
        - Notificación vía SMS o integración con otras plataformas de mensajería.

- **Historia de Usuario 4:**
  Como administrador, quiero recibir alertas en tiempo real cuando se detecten anomalías (por ejemplo, congestión extrema en el tráfico) para poder actuar de inmediato.
  Criterios de Aceptación:

- Las alertas se generan automáticamente y se muestran en una lista actualizable en el dashboard.
- El administrador puede cambiar el estado de la alerta y añadir comentarios.

- **Caso de Uso 5: Gestión y Consulta de Datos**
  - **Actor:** Administrador, Analista
  - **Precondición:** Datos cargados en la base de datos.
  - **Flujo Principal:**
    1. El usuario solicita datos históricos o en tiempo real mediante la API.
    2. El sistema consulta la base de datos MongoDB y retorna los datos solicitados.
    3. El usuario utiliza los datos para análisis o visualización.
  - **Flujo Alternativo:**
    - Aplicación de filtros avanzados para la consulta.
- **Historia de Usuario 5:**
  Como analista, quiero acceder a datos históricos para realizar estudios predictivos y tendencias que me permitan anticipar incidencias en la ciudad.
  Criterios de Aceptación:
  - La API permite consultar datos mediante filtros (fecha, tipo de sensor, ubicación).
  - La interfaz muestra la información de manera ordenada y permite exportar informes.

---

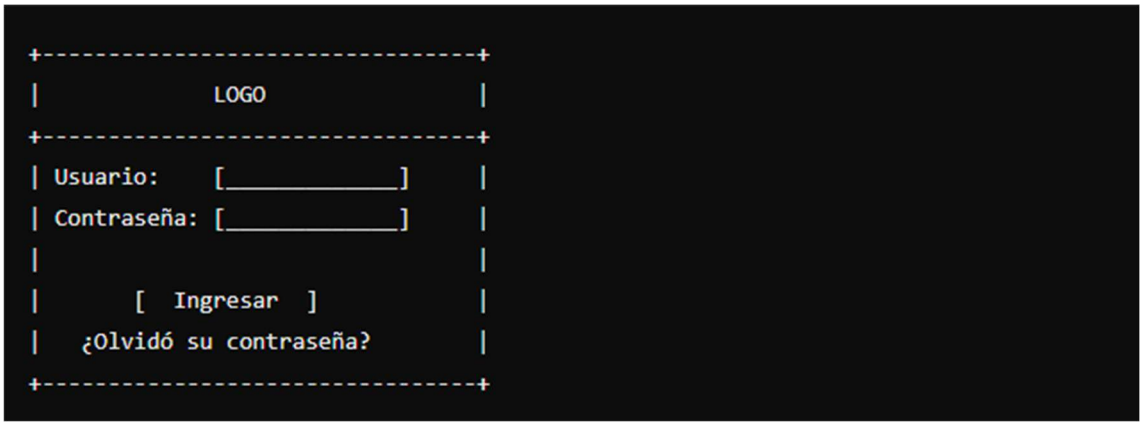
5. Bocetos y Wireframes de la Interfaz

A continuación se describen y ejemplifican los bocetos de las pantallas principales:

- **Pantalla de Login:**
  - **Elementos:**
    - Logo o nombre de la aplicación.

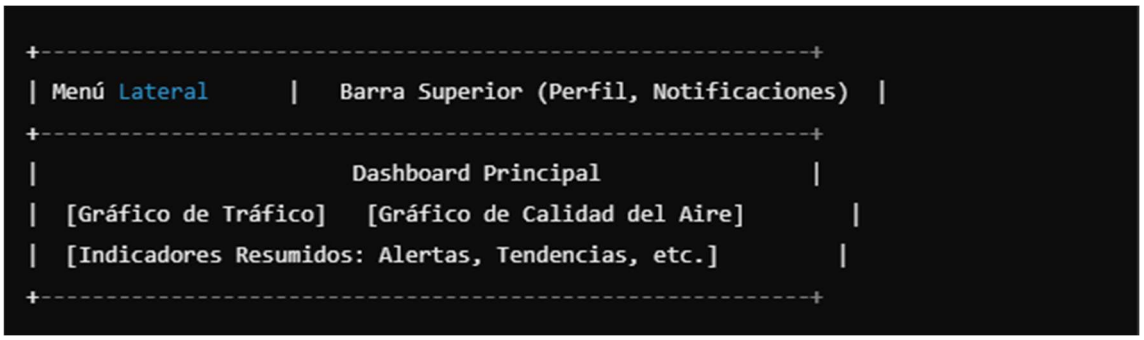
- Campos de entrada para usuario y contraseña.
- Botón de "Ingresar".
- Enlace para recuperación de contraseña.

- **Boceto:**



Dashboard Principal:

- **Elementos:**
  - Menú lateral con navegación (Dashboard, Mapa, Alertas, Configuración).
  - Barra superior con perfil de usuario, notificaciones y opciones de búsqueda.
  - Sección central con gráficos interactivos (líneas, barras, pie charts) y tarjetas de indicadores.
- **Boceto:**

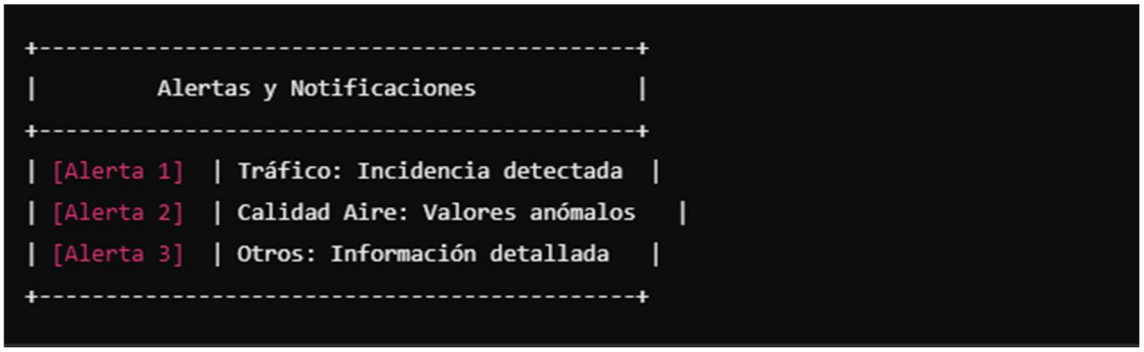


Vista de Mapa Interactivo:

- **Elementos:**
    - Mapa central con marcadores personalizados según el tipo de sensor.
    - Panel lateral o superior para aplicar filtros (tipo de sensor, fecha, etc.).
    - Panel emergente al seleccionar un marcador con detalles y gráficos de historial.
- **Boceto:**

Vista de Alertas y Notificaciones:

- **Elementos:**
    - Lista de alertas con estado (nueva, en proceso, resuelta).
    - Filtros por tipo de alerta y fecha.
    - Opción para ver detalles y actualizar el estado de la alerta.
- **Boceto:**



---

6. Arquitectura del Sistema y Modelo de Datos

- **Arquitectura General:**
    - **Frontend:**
        - Aplicación en React, responsable de la interfaz y la visualización interactiva.
        - Comunicación con el backend a través de API REST.
    - **Backend:**
        - Servidor en Node.js con Express para la lógica de negocio y endpoints REST.
        - Procesamiento de datos, generación de alertas y comunicación con la base de datos.
    - **Base de Datos:**
        - MongoDB para el almacenamiento de datos (usuarios, sensores, datos históricos, alertas).
- **Diagrama de Componentes:**

- **Modelo de Datos (Colecciones Sugeridas):**
    - **Usuarios:**
      { _id: ObjectId, nombre: String, email: String, rol: String, // (administrador, analista, ciudadano) contraseña: String, fecha_registro: Date }
    - **Sensores:**
      { _id: ObjectId, tipo_sensor: String, // Ej. "Tráfico", "Calidad del Aire"

ubicacion: { lat: Number, lng: Number }, descripción: String, datos: [{ timestamp: Date, valor: Number, metadata: Object }] }

- **Alertas:**
  { _id: ObjectId, tipo_alerta: String, sensor_id: ObjectId, descripción: String, estado: String, // ("nueva", "en proceso", "resuelta") timestamp: Date }

---

7. Diseño de la API REST

- **Endpoints Principales:**
  - **Autenticación y Usuarios:**
    - POST /api/auth/login → Validación de credenciales.
    - POST /api/auth/register → Registro de nuevos usuarios.
    - GET /api/users → Listado de usuarios (acceso restringido para administradores).
  - **Sensores y Datos:**
    - GET /api/sensores → Recuperar la lista de sensores y sus metadatos.
    - GET /api/sensores/:id → Detalle de un sensor específico.
    - GET /api/datos?sensor=<tipo>&fecha_inicio=<date>&fecha_fin=<date> → Consulta de datos filtrados.
  - **Alertas:**
    - GET /api/alertas → Listado de alertas.
    - POST /api/alertas → Creación manual o automática de alertas.
    - PUT /api/alertas/:id → Actualización del estado de una alerta.

---

8. Cronograma y Roles para la Fase de Análisis y Diseño (Fase 1)

- **Duración:** 1 semana.
- **Actividades Diarias:**
  - **Día 1:**

        - Reunión de inicio (kickoff) para repasar el Manifiesto Anthem y el dataset.
        - Asignación de roles y establecimiento de herramientas colaborativas.
    - **Día 2:**
        - Definición y documentación de los requerimientos funcionales y no funcionales.
        - Elaboración de los casos de uso y redacción de las historias de usuario.
    - **Día 3:**
        - Diseño y bocetos de la interfaz: Pantalla de login, dashboard, mapa y alertas.
        - Revisión y feedback interno sobre los wireframes.
    - **Día 4:**
        - Diseño de la arquitectura del sistema y el modelo de datos.
        - Creación de diagramas de flujo y componentes.
    - **Día 5:**
        - Especificación del diseño de la API REST y definición de endpoints.
        - Consolidación de la documentación técnica.
    - **Día 6:**
        - Revisión conjunta y ajustes en la documentación.
        - Preparación para la transición a la fase de desarrollo.
    - **Día 7:**
        - Revisión final y aprobación de la documentación completa.
- **Roles del Equipo (Equipo de Dos Personas):**
    - **Miembro A:**
        - Responsable de la definición técnica: arquitectura, modelo de datos, endpoints de la API REST.
        - Documentación técnica y diagramas (arquitectura, flujo de datos).

- **Miembro B:**
  - Responsable de la definición de la interfaz: bocetos, wireframes, casos de uso y redacción de historias de usuario.
  - Coordinación de la documentación de requerimientos y validación de la experiencia de usuario.

---

9. Herramientas y Recursos para la Documentación

- **Comunicación y Gestión de Proyectos:**
  - Slack, Microsoft Teams o similar.
  - Trello, Jira o GitHub Projects para seguimiento de tareas.
- **Documentación y Diagramación:**
  - Google Docs o Confluence para la documentación colaborativa.
  - Draw.io o Lucidchart para diagramas de arquitectura y flujo de datos.
  - Figma, Sketch o Adobe XD para los wireframes y prototipos de la interfaz.