SISTEMA DE TICKETS CLIENTE 360

Este sistema soluciona una problemática que se esta observando actualmente dentro de la institución al momento de desplegar nuevas características dentro del sistema de Cliente 360.

Problemas:

- Multiples incidencias descentralizadas
- Dificultad para el seguimiento de las incidencias
- Complejidad para recopilar cada una de las mismas
- Duplicacion de incidencias
- Incidencias quedan sin resolver
- Historial de incidencias resultas
- Ausencia de responsables de resolución
- Imposibilidad de medir desempeño

Limitaciones:

- Personal
- Tiempo
- Orden

Solucion:

Se propone el diseño e implementación de un sistema integral de gestión de tickets, orientado a mejorar el seguimiento, control y análisis de incidencias por proyecto, mediante una arquitectura moderna, escalable. La solución estará enfocada en ofrecer una experiencia de usuario ágil, reducir los tiempos de respuesta y facilitar la toma de decisiones a través de información centralizada y métricas claras.

Tecnologias:

- VueJS
- Java SpringBoot
- MinIO
- MongoDB
- Redis
- PostgreSQL
- NGINX

Implementacion:

Realizar una interfaz de usuario moderna utilizando VueJS y Nuxt, asi como estilos utilizando TailwindCSS, protegiendo rutas mediante un sistema de roles y Json Web Tokens (JWT), Asincronia para evitar largas esperas a los usuarios, Un backend en Java utilizando Spring boot para robustes, Seguido de una arquitectura por capas separando controladores, servicios, repositorios, daos y dtos, hacer uso de MinIO para el almacenamiento de archivos en diferentes formatos, uso de PostgreSQL para persistencia de datos que no requieran de alta concurrencia, uso de MongoDB para guardado de logs y auditoria, Redis para mantener data en cache que se consulta con regularidad como lo pueden ser el listado de tickets, Docker para empaquetar cada uno de los componentes para alta portabilidad y versionamiento de imágenes para contenedores.

Impacto:

- Mejor seguimiento de tickets por proyecto
- Centralización de los tickets segmentándolos por proyecto
- Graficas centralizadas para verificar cantidad de tickets en diferentes estados y rendimiento de las personas de soporte
- Reducción de tiempos de respuesta y resolución
- Mejora en la comunicación con los usuarios

Justificacion monorepo

¿Por qué Turborepo?

Turborepo es una herramienta moderna para monorepositorios enfocada en:

- Ejecución paralela y cacheada de tareas
- Alto rendimiento en builds y tests
- Integración natural con npm / pnpm / yarn workspaces
- Simplicidad de configuración

Justificación:
Turborepo permite escalar repositorios con múltiples paquetes compartidos manteniendo **tiempos de build bajos**, **dependencias explícitas** y **alta cohesión entre módulos**, sin imponer una estructura rígida.

```
monorepo/
├── apps/
│   └── web/
│
├── packages/
│   ├── ui/
│   ├── utils/
│   ├── settings/
│   └── interfaces/
│
├── turbo.json
├── package.json
├── tsconfig.base.json
└── .eslintrc.js
```

- apps/ → productos finales
- packages/ → librerías reutilizables y compartidas

packages/interfaces

Centralizar **interfaces y tipos TypeScript compartidos**, garantizando:

- Tipado consistente entre aplicaciones y librerías
- Contratos de datos claros
- Reducción de errores

Estructura

packages/interfaces/

├── src/

│ ├── ticket.interface.ts

│ ├── user.interface.ts

│ └── index.ts

├── package.json

└── tsconfig.json

4.2 packages/utils

Proveer **funciones de utilidad, helpers y hooks reutilizables**:

- Formateo de datos
- Validaciones
- Hooks comunes
- Lógica transversal

packages/utils/

```
├── src/
│   ├── date.utils.ts
│   ├── validation.utils.ts
│   ├── hooks/
│   │   └── useDebounce.ts
│   └── index.ts
```

4.3 packages/settings

Centralizar **configuraciones compartidas y constantes globales**:

- Variables de entorno tipadas
- Constantes de aplicación
- Configuración de ESLint / Prettier / Husky

Estructura

packages/settings/

```
├── src/
│   ├── env.ts
│   ├── constants.ts
│   └── index.ts
├── eslint/
│   │   └── base.eslintrc.js
```

Ejemplo:

```
export const APP_NAME = 'Ticket System';
export const DEFAULT_PAGE_SIZE = 10;
```

Justificación
Evita duplicación y asegura coherencia de configuración en todo el monorepo.

4.4 packages/ui

Propósito

Implementar una **biblioteca de componentes de UI reutilizables**, desacoplada de aplicaciones concretas.

Ejemplos:

- Button
- Modal
- Table
- Componentes de dominio (TicketCard)

Estructura

packages/ui/

```
├── src/
│   ├── components/
│   │   ├── Button/
│   │   ├── TicketCard/
│   │   └── Modal/
│   ├── theme/
│   └── index.ts
```

Justificación
ui se sitúa en la **capa superior** del monorepo, consumiendo tipado, utilidades y configuración compartida.