📄 Documento Funcional – SecureNotes

1. Introducción

SecureNotes es una aplicación web full stack desarrollada como *Proof of Concept (PoC)* con el objetivo de demostrar competencias técnicas asociadas a un perfil **Senior Full Stack Developer**.

La aplicación permite a los usuarios **registrarse, autenticarse y gestionar notas personales**, aplicando **buenas prácticas de seguridad**, arquitectura moderna y tecnologías ampliamente utilizadas en entornos productivos.

El foco principal del proyecto no está en la complejidad funcional, sino en: - Correcta toma de decisiones técnicas - Seguridad de la información - Escalabilidad - Reproducibilidad del entorno mediante contenedores

---

2. Objetivo de la Aplicación

Permitir a los usuarios: - Crear una cuenta segura - Iniciar sesión mediante autenticación basada en tokens - Crear, listar y gestionar notas personales - Elegir si el contenido de cada nota debe almacenarse de forma encriptada

Al mismo tiempo, el sistema demuestra: - Diferencias prácticas entre **hashing** y **encriptación** - Separación clara entre frontend y backend - Uso de **SPA moderna** con manejo de estado global - Uso de **Docker** para estandarizar entornos

---

3. Arquitectura General

La aplicación se compone de tres capas principales:

3.1 Frontend

- Aplicación SPA desarrollada en React (o Next.js)
- Consumo de API REST
- Manejo de estado global
- Renderizado moderno

3.2 Backend

- API REST desarrollada en ASP.NET 8 Web API
- Autenticación basada en JWT
- Lógica de negocio desacoplada
- Acceso a datos mediante Entity Framework Core

3.3 Persistencia

- Base de datos relacional SQL

- Modelo simple y normalizado

Cada componente puede ejecutarse de forma independiente o como parte de un entorno dockerizado.

---

4. Tecnologías Utilizadas

Backend

- **ASP.NET 8 Web API**
- **Entity Framework Core**
- **JWT (JSON Web Tokens)**
- **BCrypt** para hashing de contraseñas
- **AES** para encriptación de datos sensibles
- **Swagger / OpenAPI**

Frontend

- **React** o **Next.js**
- **Zustand** para manejo de estado global
- **Fetch / Axios** para comunicación HTTP

Infraestructura

- **Docker**
- **Docker Compose**
- Variables de entorno para configuración sensible

Base de Datos

- **SQL Server** o **PostgreSQL** (según entorno)

---

5. Funcionalidades Principales

5.1 Autenticación y Autorización (JWT)

- Registro de usuarios
- Login mediante email y contraseña
- Generación de JWT
- Protección de endpoints mediante autorización

Características clave: - Arquitectura stateless - Tokens enviados vía headers HTTP - Ideal para entornos distribuidos y dockerizados

---

5.2 Hashing de Contraseñas

- Las contraseñas **nunca se almacenan en texto plano**
- Se utiliza **BCrypt** para generar hashes irreversibles

Justificación técnica: - El hashing es un proceso unidireccional - Las contraseñas no deben ser recuperables - Protege frente a filtraciones de base de datos

---

5.3 Encriptación de Contenido Sensible

- El contenido de las notas puede almacenarse encriptado
- Se utiliza **AES (Advanced Encryption Standard)**
- La clave de encriptación se gestiona mediante variables de entorno

Diferencia clave con el hashing: - La encriptación es reversible - Permite recuperar la información original - Se utiliza únicamente cuando es funcionalmente necesario

---

5.4 Gestión de Notas

- Creación de notas
- Listado de notas del usuario autenticado
- Opción de almacenar contenido encriptado

Cada nota incluye: - Título - Contenido - Indicador de encriptación - Fecha de creación

---

6. Frontend – Rendering y Estado

6.1 Tipos de Rendering

Dependiendo de la implementación, la aplicación puede demostrar:

- **CSR (Client Side Rendering)**: SPA tradicional en React
- **SSR (Server Side Rendering)**: usando Next.js
- **SSG (Static Site Generation)** para páginas públicas

Esto permite demostrar conocimiento sobre: - Performance - SEO - Experiencia de usuario

---

6.2 Manejo de Estado Global – Zustand

- Manejo centralizado del estado de autenticación
- Manejo del estado de las notas

Ventajas de Zustand: - Menor complejidad que Redux - Poco boilerplate - Ideal para aplicaciones pequeñas y medianas

---

7. Seguridad

La aplicación implementa múltiples capas de seguridad:

- Hashing de contraseñas

- Encriptación de datos sensibles
- Autenticación basada en tokens
- Protección de endpoints
- Separación de secretos mediante variables de entorno

---

8. Contenedores y Entorno

La aplicación está preparada para ejecutarse mediante **Docker**, permitiendo:

- Reproducibilidad del entorno
- Aislamiento de dependencias
- Despliegue simplificado

Se utilizan: - Dockerfile para cada servicio - Docker Compose para orquestación

---

9. Alcance del PoC

Este proyecto tiene como alcance: - Demostración técnica - Evaluación de seniority - Base para futuras extensiones

No está orientado a producción, sino a evidenciar: - Buenas prácticas - Correcta toma de decisiones - Dominio del stack moderno

---

10. Conclusión

SecureNotes es una PoC diseñada para mostrar un enfoque profesional y moderno en el desarrollo de aplicaciones web, cubriendo backend, frontend, seguridad, bases de datos y contenedores, alineándose con las expectativas de un perfil Senior.