Sistema de Gestión y Almacenamiento de Documentos para Universidad

Autor: Galvez Romero Irvin Osvaldo
Asignatura: Gestión de Proyectos
Docente: Castro Modesto Yolanda
Periodo: 2025

Resumen (Abstract)

Este documento describe el diseño e implementación inicial de un sistema web para la gestión y almacenamiento de documentos de alumnos en una universidad. El objetivo es digitalizar trámites (p. ej., Reinscripción, Kárdex, Pago), reducir uso de papel, mejorar trazabilidad y acelerar la revisión por docentes/administrativos. Se detalla el modelo de datos (SQL Server 2019), la API de autenticación (Node/TypeScript), los módulos del MVP, consideraciones de seguridad, y plan de pruebas.

1. Introducción

Las universidades manejan múltiples trámites estudiantiles que requieren entrega de documentos físicos. Este proyecto propone un sistema web centralizado que permita:

- Carga y validación de documentos por los alumnos.
- Revisión y dictamen por docentes/administrativos.
- Auditoría, trazabilidad y reporteo.

2. Objetivos

- Digitalizar el flujo de recepción y revisión de documentos.
- Garantizar seguridad, integridad y trazabilidad de la información.
- Habilitar búsqueda por metadatos y texto (OCR).
- Proveer reportes operativos (SLA, volumetría).

3. Alcance (MVP)

- Portal de alumnos: registro/login, inicio de trámite, subida de documentos, estado y notificaciones.
- Portal de docentes/administrativos: revisión, comentarios, aprobación/rechazo, reasignaciones.
- Administración: catálogos (tipos de documento, requisitos por programa), flujos por trámite, roles y permisos.
- Validaciones automáticas: tipo/MIME, tamaño, duplicados; OCR para texto (índice de texto completo).
- Auditoría y reportes básicos.

No incluye (por ahora): firmas electrónicas avanzadas, integración con pagos en línea o SSO institucional.

4. Arquitectura Técnica (visión general)

- Frontend de prueba: `frontend/auth-test.html` (formulario simple de registro/login).

- Backend: Node.js + TypeScript (Express), JWT para autenticación.
- Base de datos: SQL Server 2019 Developer (instancia local `SARFERT`).
- Almacenamiento de archivos: clave de almacenamiento en BD (preparado para S3/MinIO/FS).
- Búsqueda: Full-Text Index sobre texto OCR.

5. Diseño de Base de Datos

- Motor: SQL Server 2019.
- Script DDL: `sql/univ_docs_mvp.sql` (crea base, tablas, foráneas, índices e índice de texto completo).
- Documentación detallada de tablas: `docs/db.md`.

Entidades núcleo:

- Identidad/RBAC: `usuarios`, `roles`, `permisos`, `usuarios_roles`, `roles_permisos`.
- Académico: `departamentos`, `programas`, `alumnos`, `personal`.
- Trámites: `tipos_tramite`, `tipos_documento`, `requisitos_tramite`, `pasos_flujo`.
- Instancias: `tramites_alumno`, `asignaciones_tramite`, `eventos_tramite`.
- Documentos y versiones: `documentos`, `versiones_documento`, `revisiones_documento`.
- Validaciones: `validaciones_archivo`, `escaneos_antivirus`, `ocr_texto_documento` (FTI).
- Notificaciones y auditoría: `notificaciones`, `bitacora_auditoria`.

Diagrama ER de referencia: ver `docs/casos-uso.md` (secciones de ERD) y el modelo plasmado en el DDL.

5.1. Índices clave

- `tramites_alumno(tipo_tramite_id, estatus, iniciado_en DESC)`.
- `documentos(tramite_id, estatus)` y UQ `(tramite_id, requisito_id)`.
- `versiones_documento(documento_id, version)`, `versiones_documento(sha256)`.
- FTI en `ocr_texto_documento(texto LANGUAGE 3082)` con catálogo `catalogo_ft`.

5.2. Consideraciones

- `NVARCHAR` para soporte Unicode.
- Tiempos `DATETIME2` en UTC.
- Transacciones acotadas; versionado por documento.

6. API de Autenticación (Backend)

- Código: `backend/src/tiers/auth.routes.ts`, servidor en `backend/src/server.ts`.
- Documentación: `docs/api-auth.md`.
- Variables de entorno: `.env` en `backend` (creado).

Endpoints:

- `POST /auth/register`: crea usuario y alumno en transacción, devuelve `token` y `refreshToken`.
- `POST /auth/login`: valida credenciales y devuelve `token` y `refreshToken`.
- `POST /auth/refresh`: emite nuevo access token a partir del `refreshToken`.
- `GET /auth/me`: retorna el perfil del usuario autenticado (JWT).

Pruebas: `frontend/auth-test.html` o cURL/Postman (ejemplos en `backend/README.md`).

7. Casos de Uso

- Documento: `docs/casos-uso.md` con diagramas (Mermaid) para:
  - Actores y casos de uso generales.
  - Portal de alumnos y revisión.
  - Validaciones automáticas y flujo de reinscripción (diagrama de secuencia).
  - Búsqueda y reportes.

8. Seguridad y Cumplimiento

- Autenticación: JWT (access 1h, refresh 7d).
- Contraseñas: `bcrypt` (salt 10).
- Validación de entradas: `zod`.
- SQL parametrizado con `mssql` (prevención de inyección).
- Auditoría: tabla `bitacora_auditoria`.
- Futuros: antivirus (ClamAV), DLP básico, retención legal y WORM en almacenamiento.

9. Plan de Pruebas (resumen)

- Unitarias (backend): validación de payloads y servicios (pendiente de incorporar framework de pruebas).
- Integración: registro, login, refresh, me (escenarios: éxito, credenciales inválidas, duplicados).
- Base de datos: verificación de claves foráneas, UQ y FTI activo.
- Manual: `frontend/auth-test.html` para pruebas end-to-end básicas.

10. Conclusiones y Trabajo Futuro

El sistema sienta las bases para digitalizar trámites y gestionar documentos con trazabilidad. Próximos pasos:

- Implementar portal completo en Next.js (UI/UX).
- Procesos asíncronos (colas) para OCR, antivirus y conversiones.
- Buscador avanzado (metadatos + OCR) y tablero de reportes.
- Integración con SSO institucional y firma electrónica cuando aplique.

11. Guía de Ejecución (resumen)

1. Instalar SQL Server 2019 (motor + Full-Text) y SSMS.
2. Ejecutar `sql/univ_docs_mvp.sql` en SSMS.
3. Backend: `cd backend && npm install && npm run dev`.
4. Probar: `http://localhost:4000/health` y la página `frontend/auth-test.html`.

12. Anexos

- A1. Script DDL: `sql/univ_docs_mvp.sql`.
- A2. API Auth: `docs/api-auth.md` y `backend/README.md`.
- A3. Casos de Uso: `docs/casos-uso.md`.

Exportar a PDF

Opciones:

- VS Code: instalar extensión "Markdown PDF" → abrir `docs/entrega-universidad.md` → clic derecho → `Markdown PDF: Export (pdf)`.
- Pandoc (si está instalado):
  - PowerShell:

```
pandoc -s -o entrega-universidad.pdf docs/entrega-universidad.md
```

- Navegador: abrir el archivo en un visor Markdown compatible → imprimir → "Guardar como PDF".