Módulo 1: Configuración del Sistema (Admin)

HU 1: Gestión de Sectores (Departamentos)

Como: Administrador del Sistema

Quiero: Poder crear, editar y desactivar Sectores (ej. "Contabilidad", "Desarrollo", "RRHH")

Para: Mantener actualizada la estructura organizativa de la empresa.

- **Criterios de Aceptación (Frontend):**
  - Debe existir una página en "Configuración -> Sectores".
  - La página debe mostrar una tabla con los sectores existentes (Nombre, Estado).
  - Debe haber un botón "Nuevo Sector" que abra un formulario/modal.
  - El formulario debe tener un campo "Nombre" (requerido, validado en el cliente).
  - La tabla debe tener botones de "Editar" y "Desactivar" por cada fila.
  - Al "Desactivar", debe mostrar un modal de confirmación.

- **Criterios de Aceptación (Backend - API):**
  - GET /api/sectores: Retorna un JSON listando todos los sectores.
  - POST /api/sectores: Recibe un JSON {"nombre": "Ventas"}. Debe validar que el nombre no esté duplicado. Retorna 201 Created si es exitoso o 400 Bad Request si el nombre ya existe.
  - PUT /api/sectores/{id}: Recibe un JSON {"nombre": "Ventas y Marketing"} para actualizar el nombre.
  - DELETE /api/sectores/{id}: Realiza una **baja lógica** (soft delete) del sector (ej. EstaActivo = false). Retorna 204 No Content.

HU 2: Gestión de Roles

Como: Administrador del Sistema

Quiero: Poder crear, editar y definir los Roles de los empleados (ej. "Analista Jr.", "Supervisor", "Gerente")

Para: Estandarizar las posiciones dentro de la compañía.

- **Criterios de Aceptación (Frontend):**
  - Similar a HU 1, pero en "Configuración -> Roles".
  - El formulario de "Nuevo Rol" debe pedir "Nombre" (requerido) y "Descripción" (opcional).
  - La tabla debe mostrar los roles y sus descripciones.

- **Criterios de Aceptación (Backend - API):**
  - GET /api/roles: Retorna un JSON listando todos los roles.
  - POST /api/roles: Recibe {"nombre": "Supervisor", "descripcion": "..."}. Valida duplicados. Retorna 201 Created o 400 Bad Request.
  - PUT /api/roles/{id}: Actualiza el rol.
  - DELETE /api/roles/{id}: Realiza una baja lógica del rol.

Módulo 2: Gestión de Empleados (RRHH)

HU 3: Alta de Nuevo Empleado (Cubre R1, R2, R3)

Como: Administrador de RRHH

Quiero: Registrar un nuevo empleado con sus datos personales, sector, rol, sueldo y nivel de estudio.

Para: Incorporarlo formalmente al sistema.

- **Criterios de Aceptación (Frontend):**
  - Debe existir un formulario de "Alta de Empleado".
  - El formulario debe incluir campos como Nombre, Apellido, DNI/Legajo, Fecha de Ingreso, Email, Sueldo (con validación de moneda).
  - Debe haber tres campos <select> (desplegables): "Sector", "Rol" y “Nivel de Estudio”.
  - Estos desplegables deben cargarse dinámicamente al cargar la página.
  - Debe haber validación de cliente (campos requeridos, formato de email, etc.).
  - Al guardar, debe redirigir al listado de empleados o al perfil del nuevo empleado.

- **Criterios de Aceptación (Backend - API):**
  - El FE necesitará poblar los desplegables: GET /api/sectores?estado=activo, GET /api/roles?estado=activo y GET /api/niveles-estudio.
  - POST /api/empleados: Recibe un JSON con todos los datos del empleado, incluyendo sectorId, rolId y nivelesEstudioId.
    - Ejemplo: {”nombre”: “pepe”, “legajo”: 1234, “sectorId”: 1, “roldId”: 2, “sueldo”: 1800000, “nivelEstudioId”: 4}.
  - La API **debe** validar del lado del servidor que el DNI/Legajo no esté duplicado.
  - La API debe validar que “sueldo” sea un valor positivo y que nivelEstudioId (si no es nulo) exista en la tabla NivelesEstudio.
  - Retorna 201 Created con el objeto del empleado creado, o 400 Bad Request con los errores de validación.

HU 4: Modificación de Datos del Empleado (Cubre R1, R2, R3)

Como: Administrador de RRHH

Quiero: Modificar la información de un empleado existente, incluyendo su rol, sector, sueldo o nivel de estudio.

Para: Mantener los datos actualizados.

- **Criterios de Aceptación (Frontend):**
  - Al seleccionar un empleado, se debe navegar a una página de "Editar Empleado".
  - El formulario debe ser el mismo que el de Alta (HU 3), pero pre-cargado con los datos del empleado.
  - Los desplegables de Sector, Rol y Niveles de Estudio deben mostrar el valor actualmente asignado.
  - Al presionar "Guardar", se debe mostrar una notificación de "Cambios guardados".

- **Criterios de Aceptación (Backend - API):**
  - GET /api/empleados/{id}: Retorna el JSON con todos los datos de un empleado específico para pre-cargar el formulario.
  - PUT /api/empleados/{id}: Recibe el JSON completo del empleado con los datos modificados. La API aplica los cambios en la base de datos (SQL Server). Retorna 200 OK.

HU 5: Desvinculación de Empleado (Cubre R1)

Como: Administrador de RRHH

Quiero: Dar de baja (desvincular) a un empleado que deja la compañía

Para: Reflejar que ya no forma parte de la plantilla activa.

- **Criterios de Aceptación (Frontend):**
  - En la página de "Editar Empleado" o en el listado, debe existir un botón "Desvincular".
  - Al presionarlo, debe aparecer un modal de confirmación pidiendo la "Fecha de Egreso".
  - Una vez confirmado, el empleado ya no debe aparecer en el listado principal (que solo muestra activos).

- **Criterios de Aceptación (Backend - API):**
  - POST /api/empleados/{id}/desvincular: (Usamos POST en lugar de DELETE para poder enviar la fecha en el body).
  - Recibe un JSON: {"fechaEgreso": "2025-10-24"}.
  - La API realiza la **baja lógica**: actualiza el campo FechaEgreso y/o EstaActivo = false en la DB. **No debe** hacer un DELETE de la fila.
  - Retorna 200 OK.

HU 6: Asignación de Supervisores (Cubre R4)

Como: Administrador de RRHH (o Gerente de Sector)

Quiero: Asignar un empleado (que tenga rol "Supervisor" o "Gerente") como responsable de otro empleado

Para: Definir la estructura jerárquica.

- **Criterios de Aceptación (Frontend):**
  - En el formulario de "Editar Empleado" (HU 4), debe existir un campo "Supervisor Directo".
  - Este campo debe ser un desplegable o un buscador (autocomplete) que permita seleccionar a *otro* empleado.
  - Idealmente, este buscador solo debería sugerir empleados que tengan roles de "Supervisor" o "Gerente".

- **Criterios de Aceptación (Backend - API):**
  - El FE necesita un endpoint para ese buscador: GET /api/empleados?esSupervisor=true (o GET /api/empleados/supervisores).
  - El PUT /api/empleados/{id} (de HU 4) ahora aceptará un campo adicional en el JSON: supervisorId: 123.

- La API es responsable de guardar esta clave foránea (SupervisorID) en el registro del empleado.

HU 7: Visualización y Filtro de Empleados

Como: Administrador de RRHH o Supervisor

Quiero: Ver un listado de todos los empleados y poder filtrarlos por sector o nombre

Para: Encontrar rápidamente información.

- **Criterios de Aceptación (Frontend):**
  - Debe existir una página "Empleados" que muestre una tabla/listado paginado de empleados **activos**.
  - Columnas: Nombre, Legajo, Rol, Sector, Supervisor (si tiene).
  - Debe haber una barra de búsqueda (para filtrar por Nombre/Apellido/Legajo).
  - Debe haber un desplegable "Sector" para filtrar por sector.
  - Cada vez que se cambia un filtro, el frontend debe volver a llamar a la API.

- **Criterios de Aceptación (Backend - API):**
  - GET /api/empleados: Este endpoint debe ser flexible y aceptar *query parameters*.
  - Ejemplos:
    - GET /api/empleados?pagina=1&itemsPorPagina=20 (Paginación)
    - GET /api/empleados?search=Juan (Busca por nombre/apellido)
    - GET /api/empleados?sectorId=3 (Filtra por sector)
    - GET /api/empleados?search=Juan&sectorId=3 (Combinado)
  - La API debe construir la consulta SQL (LINQ to SQL) dinámicamente.
  - Debe retornar un JSON con el listado filtrado y la información de paginación (ej. totalItems).

Módulo 3: Seguridad y Acceso

HU 8: Autenticación de Usuarios

Como: Cualquier usuario

Quiero: Iniciar sesión en el sistema con mi usuario y contraseña

Para: Acceder a la información que me corresponde.

- **Criterios de Aceptación (Frontend):**
  - Debe existir una página de Login (/login).
  - Debe pedir "Usuario" (o email) y "Contraseña".
  - Al presionar "Ingresar", debe llamar al endpoint de login.
  - Si tiene éxito, debe guardar el Token (JWT) de forma segura (ej. localStorage) y redirigir al Dashboard.
  - Debe enviar este Token en el *Header* (Authorization: Bearer <token>) de todas las futuras peticiones a la API.
  - Si falla, debe mostrar un mensaje de error "Credenciales incorrectas".

- **Criterios de Aceptación (Backend - API):**
  - POST /api/auth/login: Recibe {"username": "...", "password": "..."}.
  - La API debe verificar el usuario y *hashear* la contraseña recibida para compararla con la almacenada en SQL Server (usar .NET Identity).
  - Si es válido, genera un JWT (JSON Web Token) que incluya el UserID y el Rol del usuario.
  - Retorna 200 OK con el token: {"token": "eyJ..."}.
  - Si no es válido, retorna 401 Unauthorized.

HU 9: Control de Acceso Basado en Roles (RBAC)

Como: Administrador del Sistema

Quiero: Que el sistema restrinja las acciones según el rol del usuario autenticado

Para: Asegurar que un empleado regular no pueda modificar datos.

- **Criterios de Aceptación (Frontend):**
  - El FE debe (al iniciar sesión) decodificar el JWT o llamar a un endpoint /api/auth/me para saber el rol del usuario.
  - Debe *ocultar* elementos de la UI según el rol.
  - Ej: Si el rol es "Empleado", ocultar el menú "Configuración" y los botones "Editar" y "Desvincular" (HU 4, HU 5).
  - Ej: Si el rol es "Supervisor", solo debe ver a su equipo en el listado (HU 7).

- **Criterios de Aceptación (Backend - API):**
  - Esta es la seguridad real. **Todos** los endpoints (excepto /login) deben estar protegidos.
  - La API debe validar el JWT en cada petición.
  - Debe usar *decoradores* o *middleware* de autorización. Ej: [Authorize(Roles = "Admin, RRHH")] para el POST /api/empleados.
  - Si un usuario "Empleado" intenta llamar a POST /api/empleados (aunque el botón estuviera oculto), la API **debe** rechazarlo con 403 Forbidden.

Módulo 4: Reportes (Requerimiento Ejecutivo)

HU 10: Reporte Ejecutivo de Dotación

Como: Ejecutivo (Director o Gerente General)

Quiero: Generar un reporte (Dashboard) que muestre la distribución actual del personal, incluyendo análisis de salarios, antigüedad y nivel de académico.

Para: Tomar decisiones estratégicas sobre contratación, presupuesto, retención y capacitación.

- **Criterios de Aceptación (Frontend):**
  - Debe existir una página "Dashboard" o "Reportes".
  - Debe mostrar varios *widgets* o *tarjetas* :
    - (KPI) **Masa Salarial Total:** Suma de todos los “Sueldo” de empleados activos.
    - (KPI) **Sueldo Promedio:** Promedio de “Sueldo”.

        - (KPI) **Antigüedad Promedio:** Promedio de años/meses de servicio.
        - (Gráfico de torta) **“Distribución por Nivel de Estudio”** (ej.: 40% Universitario, 30% Terciario, etc.).
        - (Gráfico de barras) **“Sueldo Promedio por Sector”.**
        - (Gráfico de barras) **“Antigüedad Promedio por Sector”.**
        - (Gráfico de torta) **“Empleados por Sector o por Rol”.**
    - Debe haber un botón "Exportar a PDF".
    - Debe tener filtros por fecha para ver "Altas" y "Bajas" en un período.

- **Criterios de Aceptación (Backend - API):**
    - GET /api/reportes/dotacion: Este endpoint debe hacer cálculos más complejos.
    - Debe calcular y retornar las estadísticas en el JSON.
    - Debe retornar un JSON complejo que el FE pueda usar para construir los gráficos.
    - Ej: {"totalActivos": 150, "distribucionSector": [...], "masaSalarial": 75000000, "antiguedadPromedio": 5.2, "distribucionEstudio": [{"nivel": "Universitario", "cantidad": 60}, ...]}.
    - GET /api/reportes/dotacion/pdf?rango=...: (Opcional) Un endpoint que genera el PDF en el servidor y devuelve el archivo.