Índice

[Índice 1](#_Toc139379178)

[1 Empresas 2](#_Toc139379179)

[2 Tennants 4](#_Toc139379180)

[2.1 TennatsEmpresas 6](#_Toc139379181)

[3 Tipos de Documentos 8](#_Toc139379182)

[4 Sindicatos 10](#_Toc139379183)

[5 Obras Sociales 12](#_Toc139379184)

[6 Entidades bancarias 14](#_Toc139379185)

[7 Profesiones 16](#_Toc139379186)

[8 Modalidades de Contrato 18](#_Toc139379187)

[9 Etiquetas 20](#_Toc139379188)

[10 Tipo Emails 22](#_Toc139379189)

[11 Tipo Telefonos 24](#_Toc139379190)

[12 Tipo Cuentas Bancarias 26](#_Toc139379191)

[13 Provincias 28](#_Toc139379192)

[14 Códigos Postales 30](#_Toc139379193)

[15 Estado Colaboradores 32](#_Toc139379194)

[16 Áreas de trabajo 34](#_Toc139379195)

[17 Tipos Domicilios 36](#_Toc139379196)

[18 Tipos Colaboradores 38](#_Toc139379197)

[19 Paises 40](#_Toc139379198)

[20 Localidades 42](#_Toc139379199)

[21 Estados civiles 44](#_Toc139379200)

[22 Niveles de formación 46](#_Toc139379201)

[23 Colaboradores 48](#_Toc139379202)

[23.1 Colaborador Etiquetas 51](#_Toc139379203)

[23.2 Colaborador Sindicato 53](#_Toc139379204)

[23.3 Colaborador ObrasSociales 55](#_Toc139379205)

[23.4 Colaborador Teléfonos 57](#_Toc139379206)

[23.5 Colaborador Emails 59](#_Toc139379207)

[23.6 Colaborador Domicilios 61](#_Toc139379208)

[23.7 Colaborador Entidades Bancarias 63](#_Toc139379209)

[24 Notificaciones 65](#_Toc139379210)

[25 Estados Novedades 67](#_Toc139379211)

[26 Tipos de Novedades 69](#_Toc139379212)

[26.1 TipoNovedadEstados 70](#_Toc139379213)

[27 Novedades 72](#_Toc139379214)

[27.1 Novedad Estados 74](#_Toc139379215)

# Empresas

|  |  |
| --- | --- |
| **Gestión de Empresas** | |
| **Nombre** | CRUD Empresas |
| **Descripción** | Este caso de uso describe el proceso de gestión de empresas en el sistema. |
| **Actores** | Administrador del sistema |
| **Precondición** | El usuario debe tener permisos necesarios para gestionar las empresas. |
| **Flujo Normal** | 1. El usuario accede a la sección de gestión de empresas del sistema. 2. El sistema muestra la lista de empresas existentes. 3. El usuario puede realizar las siguientes acciones:    1. Crear una nueva empresa:       1. El usuario selecciona la opción de crear una nueva empresa.       2. El sistema muestra un formulario para ingresar los detalles de la nueva empresa.       3. El usuario completa los campos requeridos, como la descripción de la empresa.       4. El usuario guarda la nueva empresa.    2. Editar una empresa existente:       1. El usuario selecciona la opción de editar una empresa existente.       2. El sistema muestra un formulario con los detalles de la empresa seleccionada.       3. El usuario modifica los campos necesarios, como la descripción de la empresa.       4. El usuario guarda los cambios realizados en la empresa.    3. Eliminar una empresa existente:       1. El usuario selecciona la opción de eliminar una empresa existente.       2. El sistema muestra una confirmación para asegurarse de la eliminación de la empresa.       3. El usuario confirma la eliminación de la empresa. 4. El sistema actualiza la lista de empresas después de realizar cualquier acción. |
| **Pos condición** | Se han creado, editado o eliminado empresas según las acciones realizadas por el usuario. |
| **Permisos** | Parámetros: Empresas |
| **Tareas Necesarias** | 1 - Generar una clase capaz de almacenar los siguientes atributos:   |  |  |  |  | | --- | --- | --- | --- | | **Empresas** | | | | |  | Tipo | Obli | Observaciones | | IdEmpresa | Int32 | SI | Identificar único de la clase | | Descripcion | String | SI | Indica el Nombre de la empresa | | Baja | Bool | SI | Valor por defecto **false** | | Observaciones | Text | NO | Campo de texto |   2 – Generar una migración para la clase anterior  3 – Generar el CRUD para la clase definida en la tarea 1 |
| **Observaciones** | El campo observaciones deberá ser multiline (Rows = 5) |

# Tennants

|  |  |
| --- | --- |
| **Gestión de Tennants** | |
| **Nombre** | CRUD Tennants |
| **Descripción** | Este caso de uso describe el proceso de gestión de Tennants en el sistema. |
| **Actores** | Administrador del sistema |
| **Precondición** | El usuario debe tener permisos necesarios para gestionar los Tennants. |
| **Flujo Normal** | 1. El usuario accede a la sección de gestión de tennants del sistema. 2. El sistema muestra la lista de tennants existentes. 3. El usuario puede realizar las siguientes acciones:    1. Crear un nuevo tennant:       1. El usuario selecciona la opción de crear un nuevo tennant.       2. El sistema muestra un formulario para ingresar los detalles del nuevo tennant.       3. El usuario completa los campos requeridos, como la descripción del tennant.       4. El usuario guarda el nuevo tennant.    2. Editar un tennant existente:       1. El usuario selecciona la opción de editar un tennant existente.       2. El sistema muestra un formulario con los detalles del tennant seleccionada.       3. El usuario modifica los campos necesarios, como la descripción del tennant.       4. El usuario guarda los cambios realizados en el tennant.    3. Eliminar un tennant existente:       1. El usuario selecciona la opción de eliminar un tennant existente.       2. El sistema muestra una confirmación para asegurarse de la eliminación del tennant.       3. El usuario confirma la eliminación del tennant. 4. El sistema actualiza la lista de tennants después de realizar cualquier acción. |
| **Pos condición** | Se han creado, editado o eliminado tennants según las acciones realizadas por el usuario. |
| **Permisos** | Parámetros: Tennant |
| **Tareas Necesarias** | 1 - Generar una clase capaz de almacenar los siguientes atributos:   |  |  |  |  | | --- | --- | --- | --- | | **Tennants** | | | | |  | Tipo | Obli | Observaciones | | IdTennant | Int32 | SI | Identificar único de la clase | | descripción | String | SI | Indica el Nombre del tennant | | Baja | Bool | SI | Valor por defecto **false** | | Observaciones | Text | NO | Campo de texto |   2 – Generar una migración para la clase anterior  3 – Generar el CRUD para la clase definida en la tarea 1 |
| **Observaciones** | El campo observaciones deberá ser multiline (Rows = 5) |

## TennatsEmpresas

|  |  |
| --- | --- |
| **Gestión de TennantsEmpresas** | |
| **Nombre** | CRUD TennantsEmpresas |
| **Descripción** | Este caso de uso describe el proceso de gestión de la relación entre tennants y empresas en el sistema. |
| **Actores** | Administrador del sistema / Acceso a Tennants |
| **Precondición** | El usuario debe tener permisos necesarios para gestionar los Tennants. |
| **Flujo Normal** | 1. El usuario accede a la sección de gestión de Tennants del sistema. 2. El sistema muestra la lista de tennants existentes. 3. El usuario puede realizar las siguientes acciones:    1. Agregar una nueva empresa al tennant:       1. El usuario selecciona la opción editar un tennant y luego la opción agregar una nueva empresa al tennant.       2. El sistema muestra una lista desplegable con las empresas disponibles.       3. El usuario selecciona una empresa de la lista.       4. El usuario guarda la nueva empresa del tennant.    2. Editar una empresa del tennant existente:       1. El usuario selecciona la opción de editar una empresa existente.       2. El sistema muestra una lista desplegable con la empresa actual seleccionada y las demás empresas.       3. El usuario selecciona otra empresa.       4. El usuario guarda la nueva empresa del tennant.    3. Eliminar una empresa existente:       1. El usuario selecciona la opción de eliminar una empresa existente.       2. El sistema muestra una confirmación para asegurarse de la eliminación de la empresa.       3. El usuario confirma la eliminación de la empresa. 4. El sistema actualiza la lista de empresas del tennant después de realizar cualquier acción. |
| **Pos condición** | Se han creado, editado o eliminado empresas del tennant según las acciones realizadas por el usuario. |
| **Permisos** | Parámetros: Tennant |
| **Tareas Necesarias** | 1 - Generar una clase capaz de almacenar los siguientes atributos:   |  |  |  |  | | --- | --- | --- | --- | | **TennantEmpresas** | | | | |  | Tipo | Obli | Observaciones | | IdTennantEmpresa | Int32 | SI | Identificar único de la clase | | IdTennant | Int32 | SI | Clave foránea referenciando al tennant. | | IdEmpresa | Int32 | SI | Clave foránea referenciando a la empresa. |   2 – Generar una migración para la clase anterior  3 – Generar el CRUD para la clase definida en la tarea 1 |
| **Observaciones** | El campo observaciones deberá ser multiline (Rows = 5) |

# Tipos de Documentos

|  |  |
| --- | --- |
| **Gestión de Tipos de Documentos** | |
| **Nombre** | CRUD Tipos de Documentos |
| **Descripción** | Este caso de uso describe el proceso de gestión de Tipos de Documentos en el sistema. Los tipos de documentos se utilizan para gestionar atributos en los colaboradores. |
| **Actores** | Administrador del sistema / Acceso a Tipos de Documentos |
| **Precondición** | El usuario debe tener permisos necesarios para gestionar los tipos de documentos. |
| **Flujo Normal** | 1. El usuario accede a la sección de gestión de Tipos de Documentos del sistema. 2. El sistema muestra la lista de Tipos de Documentos existentes. 3. El usuario puede realizar las siguientes acciones:    1. Crear un nuevo Tipo de Documento:       1. El usuario selecciona la opción de crear un nuevo Tipo de Documento.       2. El sistema muestra un formulario para ingresar los detalles del nuevo Tipo de Documento.       3. El usuario completa los campos requeridos, como el nombre del Tipo de Documento.       4. El usuario guarda el nuevo Tipo de Documento.    2. Editar un Tipo de Documento existente:       1. El usuario selecciona la opción de editar un Tipo de Documento existente.       2. El sistema muestra un formulario con los detalles del Tipo de Documento seleccionado.       3. El usuario modifica los campos necesarios, como la Sigla del Tipo de Documento.       4. El usuario guarda los cambios realizados en el Tipo de Documento.    3. Eliminar un Tipo de Documento existente:       1. El usuario selecciona la opción de eliminar un Tipo de Documento existente.       2. El sistema muestra una confirmación para asegurarse de la eliminación del Tipo de Documento.       3. El usuario confirma la eliminación del Tipo de Documento. 4. El sistema actualiza la lista de Tipos de Documentos después de realizar cualquier acción. |
| **Pos condición** | Se han creado, editado o eliminado el Tipo de Documento según las acciones realizadas por el usuario. |
| **Permisos** | Parámetros: AministradorSuperUsuario |
| **Tareas Necesarias** | 1 - Generar una clase capaz de almacenar los siguientes atributos:   |  |  |  |  | | --- | --- | --- | --- | | **TiposDocumentos** | | | | |  | Tipo | Obli | Observaciones | | IdTpoDocumento | Int32 | SI | Identificador único de la clase | | NombreTipoDocumento | String | SI | Indica el Nombre del Tipo de Documento | | Baja | Bool | SI | Valor por defecto **false** | | Sigla | String | SI | Siglas del nombre del tipo de documento |   2 – Generar una migración para la clase anterior  3 – Generar el CRUD para la clase definida en la tarea 1 |
| **Excepciones** |  |
| **Observaciones** | Las siglas deben ser almacenadas en Mayúsculas,  [Documento Nacional de Identidad (D.N.I.), Cédula de Identidad (C.I.), Pasaporte (PAS), Otro (OTR.)] |

# Sindicatos

|  |  |
| --- | --- |
| **Gestión de Sindicatos** | |
| **Nombre** | CRUD Sindicatos |
| **Descripción** | Este caso de uso describe el proceso de gestión de sindicatos en el sistema. |
| **Actores** | Administrador del sistema |
| **Precondición** | El usuario debe tener permisos necesarios para gestionar los sindicatos. |
| **Flujo Normal** | 1. El usuario accede a la sección de gestión de sindicatos del sistema. 2. El sistema muestra la lista de sindicatos existentes. 3. El usuario puede realizar las siguientes acciones:    1. Crear un nuevo sindicato:       1. El usuario selecciona la opción de crear un nuevo sindicato.       2. El sistema muestra un formulario para ingresar los detalles del nuevo sindicato.       3. El usuario completa los campos requeridos.       4. El usuario guarda el nuevo sindicato.    2. Editar un sindicato existente:       1. El usuario selecciona la opción de editar un sindicato existente.       2. El sistema muestra un formulario con los detalles del sindicato seleccionado.       3. El usuario modifica los campos necesarios, como el nombre del sindicato.       4. El usuario guarda los cambios realizados en el sindicato.    3. Eliminar un sindicato existente:       1. El usuario selecciona la opción de eliminar un sindicato existente.       2. El sistema muestra una confirmación para asegurarse de la eliminación del sindicato.       3. El usuario confirma la eliminación del sindicato. 4. El sistema actualiza la lista de sindicatos después de realizar cualquier acción. |
| **Pos condición** | Se han creado, editado o eliminado sindicatos según las acciones realizadas por el usuario. |
| **Permisos** | Parámetros: Sindicatos. |
| **Tareas Necesarias** | 1 - Generar una clase capaz de almacenar los siguientes atributos:   |  |  |  |  | | --- | --- | --- | --- | | **Sindicatos** | | | | |  | Tipo | Obli | Observaciones | | IdSindicato | Int32 | SI | Identificador único de la clase. | | NombreSindicato | String | SI | Indica el Nombre del sindicato. | | IdCodPostal | Int32 | SI | Clave foránea referenciando al código postal. | | Mail | String | SI | Correo electrónico del sindicato. | | IdTennant | Int32 | SI | Indica al tennat que pertenece. | | NombreContacto | String | NO | Indica el nombre de la persona a contactar del sindicato. | | Teléfono | String | NO | Teléfono del Sindicato. | | Calle | String | SI | Nombre de la calle de la dirección del sindicato. | | Altura | String | SI | Altura de la dirección del sindicato. | | Piso | String | NO | Piso de la dirección del sindicato. | | Dpto | String | NO | Número del departamento del sindicato. | | Latitud | Decimal | NO | Número de la latitud de la dirección del sindicato. | | Longitud | Decimal | NO | Número de la longitud de la dirección del sindicato. | | OtraReferencia | String | NO | Otro dato relevante sobre la dirección del sindicato. | | Baja | Bool | NO | Estado del sindicato en el sistema. | | Observaciones | Text | NO | Datos relevantes sobre el sindicato a tener en cuenta. | | Logo | String | NO | Dirección URL del logo del sindicato. |   2 – Generar una migración para la clase anterior  3 – Generar el CRUD para la clase definida en la tarea 1 |
| **Excepciones** |  |
| **Observaciones** | El campo observaciones deberá ser multiline (Rows = 5).  Si está cargada la longitud y la latitud debemos colocar un botón para ir a la ubicación marcada. |

# Obras Sociales

|  |  |
| --- | --- |
| **Gestión de Obras Sociales** | |
| **Nombre** | CRUD Obras Sociales |
| **Descripción** | Este caso de uso describe el proceso de gestión de Obras Sociales en el sistema. Las Obras Sociales se utilizan para gestionar atributos en los colaboradores. |
| **Actores** | Administrador del sistema / Accesos a Obras Sociales |
| **Precondición** | El usuario debe tener permisos necesarios para gestionar las Obras Sociales. |
| **Flujo Normal** | 1. El usuario accede a la sección de gestión de Obras Sociales del sistema. 2. El sistema muestra la lista de obras sociales existentes. 3. El usuario puede realizar las siguientes acciones:    1. Crear una nueva obra social:       1. El usuario selecciona la opción de crear una nueva obra social.       2. El sistema muestra un formulario para ingresar los detalles de la nueva obra social.       3. El usuario completa los campos requeridos, como el nombre de la obra social.       4. El usuario guarda la nueva obra social.    2. Editar una obra social existente:       1. El usuario selecciona la opción de editar una obra social existente.       2. El sistema muestra un formulario con los detalles de la obra social seleccionada.       3. El usuario modifica los campos necesarios, como el nombre de la Obra Social.       4. El usuario guarda los cambios realizados en la obra social.    3. Eliminar una obra social existente:       1. El usuario selecciona la opción de eliminar una obra social existente.       2. El sistema muestra una confirmación para asegurarse de la eliminación de la obra social.       3. El usuario confirma la eliminación de la obra social. 4. El sistema actualiza la lista de Obras Sociales después de realizar cualquier acción. |
| **Pos condición** | Se han creado, editado o eliminado obras sociales según las acciones realizadas por el usuario. |
| **Permisos** | Parametros: ObrasSociales |
| **Tareas Necesarias** | 1 - Generar una clase capaz de almacenar los siguientes atributos:   |  |  |  |  | | --- | --- | --- | --- | | **ObrasSociales** | | | | |  | Tipo | Obli | Observaciones | | IdObraSocial | Int32 | SI | Identificar único de la clase. | | NombreObraSocial | String | SI | Indica el Nombre de la obra social. | | Baja | Bool | SI | Valor por defectofalse. | | IdCodigoPostal | Int32 | SI | Clave foránea referenciando a código postal | | Mail | String | SI | Indica el Mail de la obra social. | | NombreContacto | String | SI | Indica el nombre de contacto de la obra social. | | Telefono | String | SI | Indica el telefono de contacto de la obra social. | | Calle | String | SI | Indica la calle en la que se ubica la obra social. | | Altura | String | SI | Indica la altura a la que se encuentra la obra social. | | Piso | String | NO | Indica el piso donde se encuentra la obra social. | | Dpto | String | NO | Indica el departamento donde se encuentra la obra social. | | Latitud | Decimal | NO | Numero de latitud de la dirección de la obra social. | | Longitud | Decimal | NO | Numero de Longitud de la dirección de la obra social. | | OtraReferenciaLocacion | String | NO | Indica otra referencia de locación. | | Observaciones | Text | NO | Indica observaciones de la obra social. | | Logo | String | SI | Imagen del logo de la obra social. | | IdTennant | Int32 | SI | Indica al tennat que pertenece. |   2 – Generar una migración para la clase anterior  3 – Generar el CRUD para la clase definida en la tarea 1 |
| **Excepciones** |  |
| **Observaciones** |  |

# Entidades bancarias

|  |  |
| --- | --- |
| **Gestión de Entidades bancarias** | |
| **Nombre** | CRUD Entidades Bancarias |
| **Descripción** | Este caso de uso describe el proceso de gestión de entidades bancarias en el sistema. Las entidades bancarias representan a los diferentes bancos cargados en el sistema de por un tennat especifico. |
| **Actores** | Administrador del sistema / Acceso a entidades bancarias. |
| **Precondición** | El usuario debe tener permisos necesarios para gestionar las diferentes entidades bancarias. |
| **Flujo Normal** | 1. El usuario accede a la sección de gestión de entidades bancarias del sistema. 2. El sistema muestra la lista de entidades bancarias existentes. 3. El usuario puede realizar las siguientes acciones:    1. Crear una nueva entidad bancaria:       1. El usuario selecciona la opción de crear una nueva entidad bancaria.       2. El sistema muestra un formulario para ingresar los detalles de la nueva entidad bancaria.       3. El usuario completa los campos requeridos, como nombre de la entidad bancaria.       4. El usuario guarda la nueva entidad bancaria.    2. Editar una entidad bancaria existente:       1. El usuario selecciona la opción de editar una entidad bancaria existente.       2. El sistema muestra un formulario con los detalles de la entidad bancaria seleccionada.       3. El usuario modifica los campos necesarios, como el nombre de la entidad bancaria.       4. El usuario guarda los cambios realizados en la entidad bancaria.    3. Eliminar una entidad bancaria existente:       1. El usuario selecciona la opción de eliminar una entidad bancaria existente.       2. El sistema muestra una confirmación para asegurarse de la eliminación de la entidad bancaria.       3. El usuario confirma la eliminación de la entidad bancaria. 4. El sistema actualiza la lista de entidad bancaria después de realizar cualquier acción. |
| **Pos condición** | Se han creado, editado o eliminado entidad bancaria según las acciones realizadas por el usuario. |
| **Permisos** | Parámetros: EntidadesBancarias |
| **Tareas Necesarias** | 1 - Generar una clase capaz de almacenar los siguientes atributos:   |  |  |  |  | | --- | --- | --- | --- | | **EntidadesBancarias** | | | | |  | Tipo | Obli | Observaciones | | IdEntidadBancaria | Int32 | SI | Identificar único de la clase. | | NombreBanco | String | SI | Indica el Nombre de la entidad bancaria. | | IdCodPostal | Int32 | SI | Clave foránea referenciando al código postal. | | Mail | String | NO | Correo electrónico de la entidad bancaria. | | NombreContacto | String | NO | Indica el nombre de la persona a contactar de la entidad bancaria. | | Telefono | String | NO | Indica el número de teléfono de la entidad bancaria. | | Logo | String | NO | Indica la dirección URL del logo de la entidad bancaria. | | Baja | Bool | SI | Default false, indica el estado de la entidad bancaria. | | IdTennant | Int32 | SI | Indica al tennat que pertenece |   2 – Generar una migración para la clase anterior  3 – Generar el CRUD para la clase definida en la tarea 1 |
| **Excepciones** |  |
| **Observaciones** |  |

# Profesiones

|  |  |
| --- | --- |
| **Gestión de Profesiones** | |
| **Nombre** | CRUD Profesiones |
| **Descripción** | Este caso de uso describe el proceso de gestión de profesiones en el sistema. |
| **Actores** | Administrador del sistema / Accesos a Profesiones |
| **Precondición** | El usuario debe tener permisos necesarios para gestionar las profesiones. |
| **Flujo Normal** | 1. El usuario accede a la sección de gestión de profesiones del sistema. 2. El sistema muestra la lista de profesiones existentes. 3. El usuario puede realizar las siguientes acciones:    1. Crear una nueva profesión:       1. El usuario selecciona la opción de crear una nueva profesión.       2. El sistema muestra un formulario para ingresar los detalles de la nueva profesión.       3. El usuario completa los campos requeridos, como la descripción de la profesión.       4. El usuario guarda la nueva profesión.    2. Editar una profesión existente:       1. El usuario selecciona la opción de editar una profesión existente.       2. El sistema muestra un formulario con los detalles de la profesión seleccionada.       3. El usuario modifica los campos necesarios, como la descripción de la profesión.       4. El usuario guarda los cambios realizados en la profesión.    3. Eliminar una profesión existente:       1. El usuario selecciona la opción de eliminar una profesión existente.       2. El sistema muestra una confirmación para asegurarse de la eliminación de la profesión.       3. El usuario confirma la eliminación de la profesión. 4. El sistema actualiza la lista de profesiones después de realizar cualquier acción. |
| **Pos condición** | Se han creado, editado o eliminado profesiones según las acciones realizadas por el usuario. |
| **Permisos** | Parámetros: Profesiones |
| **Tareas Necesarias** | 1 - Generar una clase capaz de almacenar los siguientes atributos:   |  |  |  |  | | --- | --- | --- | --- | | **Profesiones** | | | | |  | Tipo | Obli | Observaciones | | IdProfesión | Int32 | SI | Identificar único de la clase. | | Descripción | String | SI | Indica el Nombre de la profesión. | | Baja | Bool | SI | Valor por defecto **false.** |   2 – Generar una migración para la clase anterior  3 – Generar el CRUD para la clase definida en la tarea 1 |
| **Excepciones** |  |

# Modalidades de Contrato

|  |  |
| --- | --- |
| **Gestión de Modalidades de Contrato** | |
| **Nombre** | CRUD Modalidades de Contrato |
| **Descripción** | Este caso de uso describe el proceso de gestión de modalidades de contrato en el sistema. Las modalidades de contrato se utilizan para gestionar atributos en los colaboradores. |
| **Actores** | Administrador del sistema / Accesos a modalidades de contrato |
| **Precondición** | El usuario debe tener permisos necesarios para gestionar las modalidades de contrato. |
| **Flujo Normal** | 1. El usuario accede a la sección de gestión de modalidades de contrato del sistema. 2. El sistema muestra la lista de modalidades de contrato existentes. 3. El usuario puede realizar las siguientes acciones:    1. Crear una nueva modalidad de contrato:       1. El usuario selecciona la opción de crear una nueva modalidad de contrato.       2. El sistema muestra un formulario para ingresar los detalles de la nueva modalidad de contrato.       3. El usuario completa los campos requeridos, como la descripción de la modalidad de contrato.       4. El usuario guarda la nueva modalidad de contrato.    2. Editar una modalidad de contrato existente:       1. El usuario selecciona la opción de editar una modalidad de contrato existente.       2. El sistema muestra un formulario con los detalles de la modalidad de contrato seleccionada.       3. El usuario modifica los campos necesarios, como la descripción de la modalidad de contrato.       4. El usuario guarda los cambios realizados en la modalidad de contrato.    3. Eliminar una modalidad de contrato existente:       1. El usuario selecciona la opción de eliminar una modalidad de contrato existente.       2. El sistema muestra una confirmación para asegurarse de la eliminación de la modalidad de contrato.       3. El usuario confirma la eliminación de la modalidad de contrato. 4. El sistema actualiza la lista de modalidades de contrato después de realizar cualquier acción. |
| **Pos condición** | Se han creado, editado o eliminado modalidades de contrato según las acciones realizadas por el usuario. |
| **Permisos** | Parametros:ModalidadesContrato |
| **Tareas Necesarias** | 1 - Generar una clase capaz de almacenar los siguientes atributos:   |  |  |  |  | | --- | --- | --- | --- | | **ModalidadesContrato** | | | | |  | Tipo | Obli | Observaciones | | IdModalidadContrato | Int32 | SI | Identificar único de la clase | | Descripción | String | SI | Indica el Nombre de la etiqueta | | Baja | Bool | SI | Valor por defecto **false** | | IdEmpresa | Int32 | SI | Indica la empresa a la que pertenece. |   2 – Generar una migración para la clase anterior  3 – Generar el CRUD para la clase definida en la tarea 1 |
| **Excepciones** |  |
| **Observaciones** | [CONTRATO PERIODO INDEFINIDO MEDIA JORNADA, CONTRATO PERIODO INDEFINIDO JORNADA COMPLETA, CONTRATO PERIODO DEFINIDO, CONSULTOR EXTERNO, OTRO] |

# Etiquetas

|  |  |
| --- | --- |
| **Gestión de Etiquetas** | |
| **Nombre** | CRUD Etiquetas |
| **Descripción** | Este caso de uso describe el proceso de gestión de etiquetas en el sistema. Las etiquetas se utilizan para gestionar atributos dinámicos en los colaboradores. |
| **Actores** | Administrador del sistema / Accesos a Etiquetas |
| **Precondición** | El usuario debe tener permisos necesarios para gestionar las etiquetas. |
| **Flujo Normal** | 1. El usuario accede a la sección de gestión de etiquetas del sistema. 2. El sistema muestra la lista de etiquetas existentes. 3. El usuario puede realizar las siguientes acciones: 4. Crear una nueva etiqueta:    * 1. El usuario selecciona la opción de crear una nueva etiqueta.      2. El sistema muestra un formulario para ingresar los detalles de la nueva etiqueta.      3. El usuario completa los campos requeridos, como la descripción de la etiqueta.      4. El usuario guarda la nueva etiqueta. 5. Editar una etiqueta existente:    * 1. El usuario selecciona la opción de editar una etiqueta existente.      2. El sistema muestra un formulario con los detalles de la etiqueta seleccionada.      3. El usuario modifica los campos necesarios, como la descripción de la etiqueta.      4. El usuario guarda los cambios realizados en la etiqueta. 6. Eliminar una etiqueta existente:    * 1. El usuario selecciona la opción de eliminar una etiqueta existente.      2. El sistema muestra una confirmación para asegurarse de la eliminación de la etiqueta.      3. El usuario confirma la eliminación de la etiqueta. 7. El sistema actualiza la lista de etiquetas después de realizar cualquier acción. |
| **Pos condición** | Se han creado, editado o eliminado etiquetas según las acciones realizadas por el usuario. |
| **Permisos** | Parametros:Etiquetas |
| **Tareas Necesarias** | 1 - Generar una clase capaz de almacenar los siguientes atributos:     |  |  |  |  | | --- | --- | --- | --- | | **Etiquetas** | | | | |  | Tipo | Obli | Observaciones | | IdEtiqueda | Int32 | SI | Identificar único de la clase | | Descripcion | String | SI | Indica el Nombre de la etiqueta | | Baja | Bool | SI | Valor por defecto **false** | | Observaciones | Text | NO | Campo de texto | | IdTennant | Int32 | SI | Indica al tennat que pertenece |       2 – Generar una migración para la clase anterior  3 – Generar el CRUD para la clase definida en la tarea 1 |
| **Excepciones** |  |
| **Observaciones** | El campo observaciones deberá ser multiline (Rows = 5) |

# Tipo Emails

|  |  |
| --- | --- |
| **Gestión de tipos de emails** | |
| **Nombre** | CRUD tipos de emails |
| **Descripción** | Este caso de uso describe el proceso de gestión de los diferentes tipos de emails en el sistema. |
| **Actores** | Administrador del sistema / SuperAdmin |
| **Precondición** | Solo el superAdmin podrá acceder a este CRUD. |
| **Flujo Normal** | 1. El usuario accede a la sección de gestión de tipos de emails del sistema. 2. El sistema muestra la lista de tipos de emails existentes. 3. El usuario puede realizar las siguientes acciones:    1. Crear un nuevo tipo de email:       1. El usuario selecciona la opción de crear un nuevo tipo de email.       2. El sistema muestra un formulario para ingresar los detalles del nuevo tipo de email.       3. El usuario completa los campos requeridos, como nombre del tipo de email.       4. El usuario guarda el nuevo tipo de email.    2. Editar un tipo de email existente:       1. El usuario selecciona la opción de editar un tipo de email existente.       2. El sistema muestra un formulario con los detalles del tipo de email seleccionada.       3. El usuario modifica los campos necesarios, como el nombre del tipo de email.       4. El usuario guarda los cambios realizados en el tipo de email.    3. Eliminar un tipo de email existente:       1. El usuario selecciona la opción de eliminar un tipo de email existente.       2. El sistema muestra una confirmación para asegurarse de la eliminación del tipo de email.       3. El usuario confirma la eliminación del tipo de email. 4. El sistema actualiza la lista de los tipos de emails después de realizar cualquier acción. |
| **Pos condición** | Se han creado, editado o eliminado tipos de email según las acciones realizadas por el usuario. |
| **Permisos** | Parámetros: SuperAdmin |
| **Tareas Necesarias** | 1 - Generar una clase capaz de almacenar los siguientes atributos:   |  |  |  |  | | --- | --- | --- | --- | | **TipoEmail** | | | | |  | Tipo | Obli | Observaciones | | IdTipoEmail | Int32 | SI | Identificar único de la clase. | | NombreTipoEmail | String | SI | Indica el Nombre del tipo de email. |   2 – Generar una migración para la clase anterior  3 – Generar el CRUD para la clase definida en la tarea 1 |
| **Excepciones** |  |
| **Observaciones** | Cargar estos datos por defecto [PERSONAL, CORPORATIVO, OTRO] |

# Tipo Telefonos

|  |  |
| --- | --- |
| **Gestión de tipos de teléfonos** | |
| **Nombre** | CRUD tipos de teléfonos |
| **Descripción** | Este caso de uso describe el proceso de gestión de los tipos de teléfonos en el sistema. |
| **Actores** | Administrador del sistema / Accesos a tipos de teléfonos |
| **Precondición** | El usuario debe tener permisos necesarios para gestionar los tipos de teléfonos. |
| **Flujo Normal** | 1. El usuario accede a la sección de gestión de tipos de teléfonos del sistema. 2. El sistema muestra la lista de tipos de teléfonos existentes. 3. El usuario puede realizar las siguientes acciones:    1. Crear un nuevo tipo de teléfono:       1. El usuario selecciona la opción de crear un nuevo tipo de teléfono.       2. El sistema muestra un formulario para ingresar los detalles del nuevo tipo de teléfono.       3. El usuario completa el campo requerido.       4. El usuario guarda el nuevo tipo de teléfono.    2. Editar un área de trabajo existente:       1. El usuario selecciona la opción de editar un tipo de teléfono existente.       2. El sistema muestra un formulario con los detalles del tipo de teléfono seleccionado.       3. El usuario modifica los campos necesarios.       4. El usuario guarda los cambios realizados en el tipo de teléfono    3. Eliminar un tipo de teléfono existente:       1. El usuario selecciona la opción de eliminar un tipo de teléfono existente.       2. El sistema muestra una confirmación para asegurarse de la eliminación del tipo de teléfono.       3. El usuario confirma la eliminación del tipo de teléfono. 4. El sistema actualiza la lista de los tipos de teléfono después de realizar cualquier acción. |
| **Pos condición** | Se han creado, editado o eliminado tipos de teléfono según las acciones realizadas por el usuario. |
| **Permisos** | Parámetros:TiposTeléfono |
| **Tareas Necesarias** | 1 - Generar una clase capaz de almacenar los siguientes atributos:   |  |  |  |  | | --- | --- | --- | --- | | **TiposTelefonos** | | | | |  | Tipo | Obli | Observaciones | | IdTiposTelefonos | Int32 | SI | Identificar único de la clase. | | NombreTipoTelefono | String | SI | Indica el Nombre del Tipo de Telefono. |   2 – Generar una migración para la clase anterior  3 – Generar el CRUD para la clase definida en la tarea 1 |
| **Observaciones** | Datos por defecto: [CELULAR PERSONAL, CELULAR CORPORATIVO, FIJO CASA, FIJO OFICINA, OTRO] |

# Tipo Cuentas Bancarias

|  |  |
| --- | --- |
| **Gestión de tipos de cuentas bancarias** | |
| **Nombre** | CRUD tipos de cuentas bancarias |
| **Descripción** | Este caso de uso describe el proceso de gestión de los diferentes tipos de cuentas bancarias en el sistema. |
| **Actores** | Administrador del sistema / SuperAdmin |
| **Precondición** | Solo el superAdmin podrá acceder a este CRUD. |
| **Flujo Normal** | 1. El usuario accede a la sección de gestión de tipos de cuentas bancarias del sistema. 2. El sistema muestra la lista de tipos de cuentas bancarias existentes. 3. El usuario puede realizar las siguientes acciones:    1. Crear un nuevo tipo de cuenta bancaria:       1. El usuario selecciona la opción de crear un nuevo tipo de cuenta bancaria.       2. El sistema muestra un formulario para ingresar los detalles del nuevo tipo de cuenta bancaria.       3. El usuario completa los campos requeridos, como nombre del tipo de cuenta bancaria.       4. El usuario guarda el nuevo tipo de cuenta bancaria.    2. Editar un tipo de cuenta bancaria existente:       1. El usuario selecciona la opción de editar un tipo de cuenta bancaria existente.       2. El sistema muestra un formulario con los detalles del tipo de cuenta bancaria seleccionada.       3. El usuario modifica los campos necesarios, como la descripción del tipo de cuenta bancaria.       4. El usuario guarda los cambios realizados en el tipo de cuenta bancaria.    3. Eliminar un tipo de cuenta bancaria existente:       1. El usuario selecciona la opción de eliminar un tipo de cuenta bancaria existente.       2. El sistema muestra una confirmación para asegurarse de la eliminación del tipo de cuenta bancaria.       3. El usuario confirma la eliminación del tipo de cuenta bancaria. 4. El sistema actualiza la lista de los tipos de cuentas bancarias después de realizar cualquier acción. |
| **Pos condición** | Se han creado, editado o eliminado tipos de cuentas bancarias según las acciones realizadas por el usuario. |
| **Permisos** | Parámetros: SuperAdmin |
| **Tareas Necesarias** | 1 - Generar una clase capaz de almacenar los siguientes atributos:   |  |  |  |  | | --- | --- | --- | --- | | **TiposCuentasBancarias** | | | | |  | Tipo | Obli | Observaciones | | IdTipoCuentaBancaria | Int32 | SI | Identificar único de la clase. | | Descripción | String | SI | Indica la descripción del tipo de cuenta bancaria. | | Baja | Bool | SI | Valor default False | | Observaciones | String | NO | Indica las observaciones del tipo de cuenta bancaria |   2 – Generar una migración para la clase anterior  3 – Generar el CRUD para la clase definida en la tarea 1 |
| **Excepciones** |  |
| **Observaciones** | Cargar estos datos por defecto [CAJA DE AHORRO EN MONEDA LOCAL, CAJA DE AHORRO EN MONEDA EXTRANJERA, CTA.CTE] |

# Provincias

|  |  |
| --- | --- |
| **Gestión de Provincias** | |
| **Nombre** | CRUD de provincias |
| **Descripción** | Este caso de uso describe el proceso de gestión de las provincias en el sistema. |
| **Actores** | Administrador del sistema / SuperAdmin |
| **Precondición** | Solo el superAdmin podrá acceder a este CRUD. |
| **Flujo Normal** | 1. El usuario accede a la sección de gestión de provincias del sistema. 2. El sistema muestra la lista de provincias existentes. 3. El usuario puede realizar las siguientes acciones:    1. Crear una nueva provincia:       1. El usuario selecciona la opción de crear una nueva provincia.       2. El sistema muestra un formulario para ingresar los detalles de la nueva provincia.       3. El usuario completa los campos requeridos, como nombre de la provincia.       4. El usuario guarda la nueva provincia.    2. Editar una provincia existente:       1. El usuario selecciona la opción de editar una provincia existente.       2. El sistema muestra un formulario con los detalles de la provincia seleccionada.       3. El usuario modifica los campos necesarios, como el nombre de la provincia.       4. El usuario guarda los cambios realizados en la provincia.    3. Eliminar una provincia existente:       1. El usuario selecciona la opción de eliminar una provincia existente.       2. El sistema muestra una confirmación para asegurarse de la eliminación de la provincia.       3. El usuario confirma la eliminación de la provincia. 4. El sistema actualiza la lista de las provincias después de realizar cualquier acción. |
| **Pos condición** | Se han creado, editado o eliminado provincia según las acciones realizadas por el usuario. |
| **Permisos** | Parámetros: SuperAdmin |
| **Tareas Necesarias** | 1 - Generar una clase capaz de almacenar los siguientes atributos:   |  |  |  |  | | --- | --- | --- | --- | | **Provincias** | | | | |  | Tipo | Obli | Observaciones | | IdProvincia | Int32 | SI | Identificar único de la clase. | | Nombre | String | SI | Indica el nombre de la provincia. | | Baja | Bool | SI | Valor default False | | IdPais | Int32 | SI | Indica el país al que pertenece la provincia |   2 – Generar una migración para la clase anterior  3 – Generar el CRUD para la clase definida en la tarea 1 |
| **Excepciones** |  |
| **Observaciones** |  |

# Códigos Postales

|  |  |
| --- | --- |
| **Gestión de Códigos Postales** | |
| **Nombre** | CRUD Codigos Postales |
| **Descripción** | Este caso de uso describe el proceso de gestión de códigos postales en el sistema. |
| **Actores** | Administrador del sistema |
| **Precondición** | Solo el superAdmin podrá acceder a este CRUD. |
| **Flujo Normal** | 1. El usuario accede a la sección de gestión de códigos postales del sistema. 2. El sistema muestra la lista de códigos postales existentes. 3. El usuario puede realizar las siguientes acciones:    1. Crear un nuevo código postal:       1. El usuario selecciona la opción de crear un nuevo código postal.       2. El sistema muestra un formulario para ingresar los detalles del nuevo código postal.       3. El usuario completa los campos requeridos.       4. El usuario guarda el nuevo código postal.    2. Editar un código postal existente:       1. El usuario selecciona la opción de editar un código postal existente.       2. El sistema muestra un formulario con los detalles del código postal seleccionado.       3. El usuario modifica los campos necesarios, como el nombre del código postal.       4. El usuario guarda los cambios realizados en el código postal.    3. Eliminar un código postal existente:       1. El usuario selecciona la opción de eliminar un código postal existente.       2. El sistema muestra una confirmación para asegurarse de la eliminación del código postal.       3. El usuario confirma la eliminación del código postal. 4. El sistema actualiza la lista de códigos postales después de realizar cualquier acción. |
| **Pos condición** | Se han creado, editado o eliminado códigos postales según las acciones realizadas por el usuario. |
| **Permisos** | Parametros: Codigospostales. |
| **Tareas Necesarias** | 1 - Generar una clase capaz de almacenar los siguientes atributos:   |  |  |  |  | | --- | --- | --- | --- | | **CodigosPostales** | | | | |  | Tipo | Obli | Observaciones | | IdCodPostal | Int32 | SI | Identificador único de la clase. | | IdLocalidad | int32 | SI | Clave foránea referenciando a la localidad | | CodPostal | String | SI | Indica el número de código postal | | Baja | Bool | SI | Valor por defecto FALSE |   2 – Generar una migración para la clase anterior  3 – Generar el CRUD para la clase definida en la tarea 1 |
| **Excepciones** |  |
| **Observaciones** |  |

# Estado Colaboradores

|  |  |
| --- | --- |
| **Gestión de Estados de Colaboradores** | |
| **Nombre** | CRUD Estado Colaboradores |
| **Descripción** | Este caso de uso describe el proceso de gestión de estados de colaboradores en el sistema. |
| **Actores** | Administrador del sistema |
| **Precondición** | Solo el superAdmin podrá acceder a este CRUD. |
| **Flujo Normal** | 1. El usuario accede a la sección de gestión de estados de colaboradores del sistema. 2. El sistema muestra la lista de estados de colaboradores existentes. 3. El usuario puede realizar las siguientes acciones:    1. Crear un nuevo estado de colaborador:       1. El usuario selecciona la opción de crear un nuevo estado de colaborador.       2. El sistema muestra un formulario para ingresar los detalles del nuevo estado de colaborador.       3. El usuario completa los campos requeridos.       4. El usuario guarda el nuevo estado de colaborador.    2. Editar un estado de colaborador existente:       1. El usuario selecciona la opción de editar un estado de colaborador existente.       2. El sistema muestra un formulario con los detalles del estado de colaborador seleccionado.       3. El usuario modifica los campos necesarios, como la descripción del estado de colaborador.       4. El usuario guarda los cambios realizados en el estado de colaborador.    3. Eliminar un estado de colaborador existente:       1. El usuario selecciona la opción de eliminar un estado de colaborador existente.       2. El sistema muestra una confirmación para asegurarse de la eliminación del estado de colaborador.       3. El usuario confirma la eliminación del estado de colaborador. 4. El sistema actualiza la lista de estados de colaboradores después de realizar cualquier acción. |
| **Pos condición** | Se han creado, editado o eliminado estados de colaboradores según las acciones realizadas por el usuario. |
| **Permisos** | Parametros: EstadoColaboradores. |
| **Tareas Necesarias** | 1 - Generar una clase capaz de almacenar los siguientes atributos:   |  |  |  |  | | --- | --- | --- | --- | | **EstadoColaboradores** | | | | |  | Tipo | Obli | Observaciones | | IdEstadoColaborador | Int32 | SI | Identificador único de la clase. | | Descripción | String | SI | Indica la descripción del estado | | IdTennant | Int32 | SI | Indica el tennat al que pertenece | | Baja | Bool | SI | Valor por defecto FALSE |   2 – Generar una migración para la clase anterior  3 – Generar el CRUD para la clase definida en la tarea 1 |
| **Excepciones** |  |
| **Observaciones** |  |

# Áreas de trabajo

|  |  |
| --- | --- |
| **Gestión de áreas de trabajo** | |
| **Nombre** | CRUD Áreas de trabajo |
| **Descripción** | Este caso de uso describe el proceso de gestión de las diferentes áreas de trabajo en una determinada empresa en el sistema. |
| **Actores** | Administrador del sistema / Accesos a Áreas de trabajo |
| **Precondición** | El usuario debe tener permisos necesarios para gestionar las áreas de trabajo. |
| **Flujo Normal** | 1. El usuario accede a la sección de gestión de áreas de trabajo del sistema. 2. El sistema muestra la lista de áreas de trabajo existentes. 3. El usuario puede realizar las siguientes acciones:    1. Crear una nueva área de trabajo:       1. El usuario selecciona la opción de crear una nueva área de trabajo.       2. El sistema muestra un formulario para ingresar los detalles de la nueva área de trabajo.       3. El usuario completa los campos requeridos, como la descripción de la etiqueta.       4. El usuario guarda la nueva área de trabajo.    2. Editar un área de trabajo existente:       1. El usuario selecciona la opción de editar un área de trabajo existente.       2. El sistema muestra un formulario con los detalles del área de trabajo seleccionada.       3. El usuario modifica los campos necesarios, como la descripción del área de trabajo.       4. El usuario guarda los cambios realizados en el área de trabajo.    3. Eliminar un área de trabajo existente:       1. El usuario selecciona la opción de eliminar un área de trabajo existente.       2. El sistema muestra una confirmación para asegurarse de la eliminación del área de trabajo.       3. El usuario confirma la eliminación del área de trabajo. 4. El sistema actualiza la lista de las áreas de trabajo después de realizar cualquier acción. |
| **Pos condición** | Se han creado, editado o eliminado áreas de trabajo según las acciones realizadas por el usuario. |
| **Permisos** | Parámetros: Áreas de trabajo |
| **Tareas Necesarias** | 1 - Generar una clase capaz de almacenar los siguientes atributos:   |  |  |  |  | | --- | --- | --- | --- | | **AreasTrabajo** | | | | |  | Tipo | Obli | Observaciones | | IdAreaTrabajo | Int32 | SI | Identificar único de la clase. | | Descripción | String | SI | Indica el Nombre del área de trabajo. | | Baja | Bool | SI | Valor por defecto **false** | | IdEmpresa | Int32 | SI | Indica la empresa a la que pertenece |   2 – Generar una migración para la clase anterior  3 – Generar el CRUD para la clase definida en la tarea 1 |
| **Excepciones** |  |
| **Observaciones** |  |

# Tipos Domicilios

|  |  |
| --- | --- |
| **Gestión de tipos de domicilios** | |
| **Nombre** | CRUD tipos de domicilios |
| **Descripción** | Este caso de uso describe el proceso de gestión de los diferentes tipos de domicilios en el sistema. |
| **Actores** | Administrador del sistema / SuperAdmin |
| **Precondición** | Solo el superAdmin podrá acceder a este CRUD. |
| **Flujo Normal** | 1. El usuario accede a la sección de gestión de tipos de domicilios del sistema. 2. El sistema muestra la lista de tipos de domicilios existentes. 3. El usuario puede realizar las siguientes acciones:    1. Crear un nuevo tipo de domicilio:       1. El usuario selecciona la opción de crear un nuevo tipo de domicilio.       2. El sistema muestra un formulario para ingresar los detalles del nuevo tipo de domicilio.       3. El usuario completa los campos requeridos, como nombre del tipo de domicilio.       4. El usuario guarda el nuevo tipo de domicilio.    2. Editar un tipo de domicilio existente:       1. El usuario selecciona la opción de editar un tipo de domicilio existente.       2. El sistema muestra un formulario con los detalles del tipo de domicilio seleccionada.       3. El usuario modifica los campos necesarios, como el nombre del tipo de domicilio.       4. El usuario guarda los cambios realizados en el tipo de domicilio.    3. Eliminar un tipo de domicilio existente:       1. El usuario selecciona la opción de eliminar un tipo de domicilio existente.       2. El sistema muestra una confirmación para asegurarse de la eliminación del tipo de domicilio.       3. El usuario confirma la eliminación del tipo de domicilio. 4. El sistema actualiza la lista de los tipos de emails después de realizar cualquier acción. |
| **Pos condición** | Se han creado, editado o eliminado tipos de domicilios según las acciones realizadas por el usuario. |
| **Permisos** | Parámetros: SuperAdmin |
| **Tareas Necesarias** | 1 - Generar una clase capaz de almacenar los siguientes atributos:   |  |  |  |  | | --- | --- | --- | --- | | **TipoDomicilio** | | | | |  | Tipo | Obli | Observaciones | | IdTipoDomicilio | Int32 | SI | Identificar único de la clase. | | NombreTipoDomicilio | String | SI | Indica el Nombre del tipo de domicilio. |   2 – Generar una migración para la clase anterior  3 – Generar el CRUD para la clase definida en la tarea 1 |
| **Excepciones** |  |
| **Observaciones** | Cargar estos datos por defecto [LABORAL, PERSONAL PRINCIPAL, PERSONAL SECUNDARIA, ENVIO] |

# Tipos Colaboradores

|  |  |
| --- | --- |
| **Gestión de tipos de colaboradores** | |
| **Nombre** | CRUD tipos de colaboradores |
| **Descripción** | Este caso de uso describe el proceso de gestión de los diferentes tipos de colaboradores en el sistema. |
| **Actores** | Administrador del sistema / SuperAdmin |
| **Precondición** | Solo el superAdmin podrá acceder a este CRUD. |
| **Flujo Normal** | 1. El usuario accede a la sección de gestión de tipos de colaboradores del sistema. 2. El sistema muestra la lista de tipos de colaboradores existentes. 3. El usuario puede realizar las siguientes acciones:    1. Crear un nuevo tipo de colaborador:       1. El usuario selecciona la opción de crear un nuevo tipo de colaborador.       2. El sistema muestra un formulario para ingresar los detalles del nuevo tipo de colaborador.       3. El usuario completa los campos requeridos, como nombre del tipo de colaborador.       4. El usuario guarda el nuevo tipo de colaborador.    2. Editar un tipo de colaborador existente:       1. El usuario selecciona la opción de editar un tipo de colaborador existente.       2. El sistema muestra un formulario con los detalles del tipo de colaborador seleccionada.       3. El usuario modifica los campos necesarios, como el nombre del tipo de colaborador.       4. El usuario guarda los cambios realizados en el tipo de colaborador.    3. Eliminar un tipo de colaborador existente:       1. El usuario selecciona la opción de eliminar un tipo de colaborador existente.       2. El sistema muestra una confirmación para asegurarse de la eliminación del tipo de colaborador.       3. El usuario confirma la eliminación del tipo de colaborador. 4. El sistema actualiza la lista de los tipos de emails después de realizar cualquier acción. |
| **Pos condición** | Se han creado, editado o eliminado tipos de colaboradores según las acciones realizadas por el usuario. |
| **Permisos** | Parámetros: SuperAdmin |
| **Tareas Necesarias** | 1 - Generar una clase capaz de almacenar los siguientes atributos:   |  |  |  |  | | --- | --- | --- | --- | | **TipoColaboradores** | | | | |  | Tipo | Obli | Observaciones | | IdTipoColaborador | Int32 | SI | Identificar único de la clase. | | NombreTipoColaborador | String | SI | Indica el Nombre del tipo de colaborador. | | Baja | Bool | SI | Por defecto false. | | IdEmpresa | Int32 | SI | Empresa a la que pertenece el tipo de colaborador |   2 – Generar una migración para la clase anterior  3 – Generar el CRUD para la clase definida en la tarea 1 |
| **Excepciones** |  |
| **Observaciones** |  |

# Paises

|  |  |
| --- | --- |
| **Gestión de países** | |
| **Nombre** | CRUD países |
| **Descripción** | Este caso de uso describe el proceso de gestión de los diferentes países en el sistema. |
| **Actores** | Administrador del sistema / SuperAdmin |
| **Precondición** | Solo el superAdmin podrá acceder a este CRUD. |
| **Flujo Normal** | 1. El usuario accede a la sección de gestión de países del sistema. 2. El sistema muestra la lista de países existentes. 3. El usuario puede realizar las siguientes acciones:    1. Crear un nuevo país:       1. El usuario selecciona la opción de crear un nuevo país.       2. El sistema muestra un formulario para ingresar los detalles del nuevo país.       3. El usuario completa los campos requeridos, como nombre del país.       4. El usuario guarda el nuevo país.    2. Editar un país existente:       1. El usuario selecciona la opción de editar un país existente.       2. El sistema muestra un formulario con los detalles del país seleccionada.       3. El usuario modifica los campos necesarios, como el nombre del país.       4. El usuario guarda los cambios realizados en el país.    3. Eliminar un país existente:       1. El usuario selecciona la opción de eliminar un país existente.       2. El sistema muestra una confirmación para asegurarse de la eliminación del país.       3. El usuario confirma la eliminación del país. 4. El sistema actualiza la lista de los países después de realizar cualquier acción. |
| **Pos condición** | Se han creado, editado o eliminado países según las acciones realizadas por el usuario. |
| **Permisos** | Parámetros: SuperAdmin |
| **Tareas Necesarias** | 1 - Generar una clase capaz de almacenar los siguientes atributos:   |  |  |  |  | | --- | --- | --- | --- | | **Países** | | | | |  | Tipo | Obli | Observaciones | | IdPaís | Int32 | SI | Identificar único de la clase. | | NombrePais | String | SI | Indica el Nombre del país. | | Baja | Bool | SI | Por defecto false. |   2 – Generar una migración para la clase anterior  3 – Generar el CRUD para la clase definida en la tarea 1 |
| **Excepciones** |  |
| **Observaciones** |  |

# Localidades

|  |  |
| --- | --- |
| **Gestión de localidades** | |
| **Nombre** | CRUD localidades |
| **Descripción** | Este caso de uso describe el proceso de gestión de las diferentes localidades en el sistema. |
| **Actores** | Administrador del sistema / SuperAdmin |
| **Precondición** | Solo el superAdmin podrá acceder a este CRUD. |
| **Flujo Normal** | 1. El usuario accede a la sección de gestión de las localidades del sistema. 2. El sistema muestra la lista de las localidades existentes. 3. El usuario puede realizar las siguientes acciones:    1. Crear una nueva localidad:       1. El usuario selecciona la opción de crear una nueva localidad.       2. El sistema muestra un formulario para ingresar los detalles de la nueva localidad.       3. El usuario completa los campos requeridos, como nombre de la localidad.       4. El usuario guarda la nueva localidad.    2. Editar una localidad existente:       1. El usuario selecciona la opción de editar una localidad existente.       2. El sistema muestra un formulario con los detalles de la localidad seleccionada.       3. El usuario modifica los campos necesarios, como el nombre de la localidad.       4. El usuario guarda los cambios realizados en la localidad.    3. Eliminar una localidad existente:       1. El usuario selecciona la opción de eliminar una localidad existente.       2. El sistema muestra una confirmación para asegurarse de la eliminación de la localidad.       3. El usuario confirma la eliminación de la localidad. 4. El sistema actualiza la lista de las localidades después de realizar cualquier acción. |
| **Pos condición** | Se han creado, editado o eliminado localidades según las acciones realizadas por el usuario. |
| **Permisos** | Parámetros: SuperAdmin |
| **Tareas Necesarias** | 1 - Generar una clase capaz de almacenar los siguientes atributos:   |  |  |  |  | | --- | --- | --- | --- | | **Localidades** | | | | |  | Tipo | Obli | Observaciones | | IdLocalidad | Int32 | SI | Identificar único de la clase. | | NombreLocalidad | String | SI | Indica el Nombre de la localidad. | | Baja | Bool | SI | Por defecto false. | | IdProvincia | Int32 | SI | Clave foránea a provincia. |   2 – Generar una migración para la clase anterior  3 – Generar el CRUD para la clase definida en la tarea 1 |
| **Excepciones** |  |
| **Observaciones** |  |

# Estados civiles

|  |  |
| --- | --- |
| **Gestión de Estados civiles** | |
| **Nombre** | CRUD Estados civiles |
| **Descripción** | Este caso de uso describe el proceso de gestión de los diferentes Estados civiles en el sistema. |
| **Actores** | Administrador del sistema / SuperAdmin |
| **Precondición** | Solo el superAdmin podrá acceder a este CRUD. |
| **Flujo Normal** | 1. El usuario accede a la sección de gestión de los Estados civiles del sistema. 2. El sistema muestra la lista de los Estados civiles existentes. 3. El usuario puede realizar las siguientes acciones:    1. Crear un nuevo estado civil:       1. El usuario selecciona la opción de crear un nuevo estado civil.       2. El sistema muestra un formulario para ingresar los detalles del nuevo estado civil.       3. El usuario completa los campos requeridos, como nombre de estado civil.       4. El usuario guarda la nueva estado civil.    2. Editar un estado civil existente:       1. El usuario selecciona la opción de editar un estado civil existente.       2. El sistema muestra un formulario con los detalles del estado civil seleccionada.       3. El usuario modifica los campos necesarios, como el nombre del estado civil.       4. El usuario guarda los cambios realizados en el estado civil.    3. Eliminar un estado civil existente:       1. El usuario selecciona la opción de eliminar un estado civil existente.       2. El sistema muestra una confirmación para asegurarse de la eliminación del estado civil.       3. El usuario confirma la eliminación del estado civil. 4. El sistema actualiza la lista de los estados civiles después de realizar cualquier acción. |
| **Pos condición** | Se han creado, editado o eliminado estados civiles según las acciones realizadas por el usuario. |
| **Permisos** | Parámetros: SuperAdmin |
| **Tareas Necesarias** | 1 - Generar una clase capaz de almacenar los siguientes atributos:   |  |  |  |  | | --- | --- | --- | --- | | **Estados civiles** | | | | |  | Tipo | Obli | Observaciones | | IdEstadoCivil | Int32 | SI | Identificar único de la clase. | | DescripcionEstadoCivil | String | SI | Indica el Nombre del estado civil. | | Baja | Bool | SI | Por defecto false. |   2 – Generar una migración para la clase anterior  3 – Generar el CRUD para la clase definida en la tarea 1 |
| **Excepciones** |  |
| **Observaciones** | Insertar estos datos por defecto: [SOLTERO/A, CASADO/A, DIVORCIADO/A, VIUDO/A,] |

# Niveles de formación

|  |  |
| --- | --- |
| **Gestión de Niveles de formación** | |
| **Nombre** | CRUD Niveles de formación |
| **Descripción** | Este caso de uso describe el proceso de gestión de los diferentes niveles de formación en el sistema. |
| **Actores** | Administrador del sistema / SuperAdmin |
| **Precondición** | Solo el superAdmin podrá acceder a este CRUD. |
| **Flujo Normal** | 1. El usuario accede a la sección de gestión de los niveles de formación del sistema. 2. El sistema muestra la lista de los niveles de formación existentes. 3. El usuario puede realizar las siguientes acciones:    1. Crear un nuevo nivel de formación:       1. El usuario selecciona la opción de crear un nuevo nivel de formación.       2. El sistema muestra un formulario para ingresar los detalles del nuevo nivel de formación.       3. El usuario completa los campos requeridos, como nombre de nivel de formación.       4. El usuario guarda un nuevo nivel de formación.    2. Editar un nivel de formación existente:       1. El usuario selecciona la opción de editar un nivel de formación existente.       2. El sistema muestra un formulario con los detalles del nivel de formación seleccionada.       3. El usuario modifica los campos necesarios, como el nombre del nivel de formación.       4. El usuario guarda los cambios realizados en el nivel de formación.    3. Eliminar un nivel de formación existente:       1. El usuario selecciona la opción de eliminar un nivel de formación existente.       2. El sistema muestra una confirmación para asegurarse de la eliminación del nivel de formación.       3. El usuario confirma la eliminación del nivel de formación. 4. El sistema actualiza la lista de los Niveles de formación después de realizar cualquier acción. |
| **Pos condición** | Se han creado, editado o eliminado Niveles de formación según las acciones realizadas por el usuario. |
| **Permisos** | Parámetros: SuperAdmin |
| **Tareas Necesarias** | 1 - Generar una clase capaz de almacenar los siguientes atributos:   |  |  |  |  | | --- | --- | --- | --- | | **Niveles de formación** | | | | |  | Tipo | Obli | Observaciones | | IdNivelFomacion | Int32 | SI | Identificar único de la clase. | | DescripcionNivelFormacion | String | SI | Indica el Nombre del nivel de formación. | | Baja | Bool | SI | Por defecto false. |   2 – Generar una migración para la clase anterior  3 – Generar el CRUD para la clase definida en la tarea 1 |
| **Excepciones** |  |
| **Observaciones** | Insertar estos datos por defecto: [PRIMARIO INCOMPLETO, PRIMARIO COMPLETO, SECUNDARIO INCOMPLETO, SECUNDARIO COMPLETO, UNIVERSITARIO INCOMPLETO, UNIVERSITARIO COMPLETO, POSTGRADO - ESPCIALIZACION, POSTGRADO - MAESTRIA - POSTGRADO DOCTORADO - POSTGRADOO PhD] |

# Colaboradores

|  |  |
| --- | --- |
| **Gestión de Colaboradores** | |
| **Nombre** | CRUD Colaboradores |
| **Descripción** | Este caso de uso describe el proceso de gestión de colaboradores en el sistema. |
| **Actores** | Administrador del sistema / Accesos a colaboradores |
| **Precondición** | El usuario debe tener permisos necesarios para gestionar los colaboradores. |
| **Flujo Normal** | 1. El usuario accede a la sección de gestión de colaboradores del sistema. 2. El sistema muestra la lista de colaboradores existentes. 3. El usuario puede realizar las siguientes acciones:    1. Crear un nuevo colaborador:       1. El usuario selecciona la opción de crear un nuevo colaborador.       2. El sistema muestra un formulario para ingresar los detalles del nuevo colaborador.       3. El usuario completa los campos requeridos, como la descripción del colaborador.       4. El usuario guarda el nuevo colaborador.    2. Editar un colaborador existente:       1. El usuario selecciona la opción de editar un colaborador existente.       2. El sistema muestra un formulario con los detalles del colaborador seleccionado.       3. El usuario modifica los campos necesarios, como la descripción del colaborador.       4. El usuario guarda los cambios realizados en el colaborador.    3. Eliminar un colaborador existente:       1. El usuario selecciona la opción de eliminar un colaborador existente.       2. El sistema muestra una confirmación para asegurarse de la eliminación del colaborador.       3. El usuario confirma la eliminación del colaborador. 4. El sistema actualiza la lista de colaboradores después de realizar cualquier acción. |
| **Pos condición** | Se han creado, editado o eliminado colaboradores según las acciones realizadas por el usuario. |
| **Permisos** | Parámetros: Etiquetas |
| **Tareas Necesarias** | 1 - Generar una clase capaz de almacenar los siguientes atributos:   |  |  |  |  | | --- | --- | --- | --- | | **Colaboradores** | | | | |  | Tipo | Obli | Observaciones | | IdColaborador | Int32 | SI | Identificar único de la clase | | IdTipoDocumento | Int32 | NO | Clave foránea referenciando a tipo de documento. | | Baja | Bool | SI | Valor por defecto **false** | | IdTipoColaborador | Int32 | NO | clave foránea referenciando a tipo de colaborador. | | IdEmpresa | Int32 | SI | Indica la empresa a la que pertenece | | NroLegajo | String | NO | Numero de legajo del colaborador. | | Nombre | String | SI | Nombre del colaborador. | | Apellido | String | SI | Apellido del colaborador. | | NroDocumento | String | SI | Numero de documento del colaborador. | | Cuil | String | NO | Cuil del colaborador. | | AutoridadDocumento | String | NO | Autoridad del documento del colaborador. | | IdPaisNacimiento | Int32 | NO | clave foránea referenciando al país. | | FechaNacimiento | Date | NO | Fecha de nacimiento del colaborador. | | FechaAltaTemprana | Date | NO | Fecha Alta temprana del colaborador. | | FechaInicio | Date | SI | Fecha de inicio del colaborador. | | FechaEgreso | Date | NO | Fecha de egreso del colaborador. | | FechaFallecimiento | Date | NO | Fecha fallecimiento del colaborador. | | IdEstadoCivil | Int32 | Si | Clave foránea al estado civil. | | NroNupcia | Int | NO | Número del nupcial del colaborador. | | IdNivelFormacion | Int32 | NO | Clave foránea al nivel de formación. | | TieneDiscapacidad | Bool | SI | Default False. | | DescripcionDiscapacidad | String | NO | Detalles sobre la discapacidad del colaborador. | | CertificadoMedico | Bool | Si | Default false. | | IdProfesión | Int32 | NO | Clave foránea referenciando a Profesión | | OtraProfesión | String | NO | Alguna otra habilidad del colaborador. | | Web | String | NO | Dirección URL de la página web del colaborador. | | IdEstadoColaborador | Int32 | NO | Clave foránea referenciando al estado del colaborador. | | IdModalidadContrato | Int32 | NO | Clave foránea referenciando a la modalidad de contrato. | | IdAreaTrabajo | Int32 | NO | Clave foránea referenciando área de trabajo. | | Foto | String | NO | Dirección URL a la foto del colaborador. | | DocumentacionPersonal | String(Max) | NO | Ruta al archivo. | | DocumentacionFomacion | String(Max) | NO | Ruta al archivo. | | DocumentacionSeguimientoInterna | String(Max) | NO | Ruta al archivo. | | GaleriaImagenes | String(Max) | NO | Ruta al archivo. |   2 – Generar una migración para la clase anterior  3 – Generar el CRUD para la clase definida en la tarea 1 |
| **Excepciones** |  |
| **Observaciones** | El campo observaciones deberá ser multiline (Rows = 5) |

## Colaborador Etiquetas

|  |  |
| --- | --- |
| **Gestión de Colaboradores Etiquetas** | |
| **Nombre** | CRUD Colaboradores Etiquetas |
| **Descripción** | Este caso de uso describe el proceso de gestión de la relación entre colaboradores y etiquetas en el sistema. |
| **Actores** | Administrador del sistema / Accesos a Colaboradores |
| **Precondición** | El usuario debe tener permisos necesarios para gestionar los Colaboradores. |
| **Flujo Normal** | 1. El usuario accede a la sección de gestión de colaboradores del sistema. 2. El sistema muestra la lista de colaboradores existentes. 3. El usuario puede realizar las siguientes acciones:    1. Agregar una nueva etiqueta al colaborador:       1. El usuario selecciona la opción editar un colaborador y luego añadir una etiqueta.       2. El sistema muestra una lista desplegable con todas las etiquetas.       3. El usuario selecciona una de las opciones de la lista.       4. El usuario guarda la nueva etiqueta del colaborador.    2. Editar una etiqueta del colaborador existente:       1. El usuario selecciona la opción de editar una etiqueta existente.       2. El sistema muestra una lista desplegable con la etiqueta actual seleccionada y las demás etiquetas.       3. El usuario selecciona otra etiqueta.       4. El usuario guarda la nueva etiqueta del colaborador.    3. Eliminar una etiqueta existente:       1. El usuario selecciona la opción de eliminar una etiqueta existente.       2. El sistema muestra una confirmación para asegurarse de la eliminación de la etiqueta.       3. El usuario confirma la eliminación de la etiqueta. 4. El sistema actualiza la lista de etiquetas del colaborador después de realizar cualquier acción. |
| **Pos condición** | Se han creado, editado o eliminado etiquetas del colaborador según las acciones realizadas por el usuario. |
| **Permisos** | Colaboradores |
| **Tareas Necesarias** | 1 - Generar una clase capaz de almacenar los siguientes atributos:   |  |  |  |  | | --- | --- | --- | --- | | **ColaboradoresEtiquetas** | | | | |  | Tipo | Obli | Observaciones | | IdColaboradoresEtiquetas | Int32 | SI | Identificar único de la clase. | | IdColaborador | Int32 | SI | Clave foránea referenciando al colaborador. | | IdEtiqueta | Int32 | SI | Clave foránea referenciando a la etiqueta. | | Observaciones | Text | NO | Campo de texto. | | Valor | String | SI | Indica el valor de la etiqueta. |   2 – Generar una migración para la clase anterior  3 – Generar el CRUD para la clase definida en la tarea 1 |
| **Excepciones** |  |
| **Observaciones** | El campo observaciones deberá ser multiline (Rows = 5) |

## Colaborador Sindicato

|  |  |
| --- | --- |
| **Gestión de Colaborador Sindicato** | |
| **Nombre** | CRUD Colaborador Sindicato |
| **Descripción** | Este caso de uso describe el proceso de gestión de la relación entre colaborador y sindicatos en el sistema. |
| **Actores** | Administrador del sistema / Accesos a Colaboradores |
| **Precondición** | El usuario debe tener permisos necesarios para gestionar los Colaboradores y además debe estar localizado en el administrador de colaboradores. |
| **Flujo Normal** | 1. El usuario accede a la sección de gestión de los sindicatos de un colaborador en específico. 2. El sistema muestra la lista de sindicatos del colaborador. 3. El usuario puede realizar las siguientes acciones:    1. Agregar un nuevo sindicato al colaborador:       1. El usuario selecciona la opción de añadir un sindicato al colaborador.       2. El sistema muestra una lista desplegable con todos los sindicatos.       3. El usuario selecciona una de las opciones de la lista.       4. El usuario guarda el sindicato para el colaborador.    2. Editar la lista de sindicatos existente:       1. El usuario selecciona la opción de editar la lista de sindicatos existente.       2. El sistema muestra una lista desplegable con los diferentes sindicatos.       3. El usuario selecciona otro sindicato.       4. El usuario guarda los cambios realizados.    3. Eliminar un sindicato de la lista existente:       1. El usuario selecciona la opción de eliminar un sindicato de la lista existente.       2. El sistema muestra una confirmación para asegurarse de la eliminación.       3. El usuario confirma la eliminación del sindicato de la lista. 4. El sistema actualiza la lista de sindicatos del colaborador después de realizar cualquier acción. |
| **Pos condición** | Se han creado, editado o eliminado sindicatos del colaborador según las acciones realizadas por el usuario. |
| **Permisos** | Colaboradores |
| **Tareas Necesarias** | 1 - Generar una clase capaz de almacenar los siguientes atributos:   |  |  |  |  | | --- | --- | --- | --- | | **ColaboradorSindicatos** | | | | |  | Tipo | Obli | Observaciones | | IdColaboradorSindicatos | Int32 | SI | Identificar único de la clase | | IdColaborador | Int32 | SI | Clave foránea referenciando al colaborador. | | IdSindicato | Int32 | SI | Clave foránea referenciando al sindicato. | | FechaAlta | Date | NO | Fecha en la que al colaborador se asoció al sindicato. | | FechaBaja | Date | NO | Fecha en la que el colaborador dejo de ser socio del sindicato. |   2 – Generar una migración para la clase anterior  3 – Generar el CRUD para la clase definida en la tarea 1 |
| **Excepciones** |  |
| **Observaciones** | El campo fechaAlta por default es getDate(). |

## Colaborador ObrasSociales

|  |  |
| --- | --- |
| **Gestión de Colaborador Obras Sociales** | |
| **Nombre** | CRUD Colaborador Obras Sociales |
| **Descripción** | Este caso de uso describe el proceso de gestión de la relación entre colaborador y las Obras Sociales en el sistema. |
| **Actores** | Administrador del sistema / Accesos a Colaboradores |
| **Precondición** | El usuario debe tener permisos necesarios para gestionar los Colaboradores y además debe estar localizado en el administrador de colaboradores. |
| **Flujo Normal** | 1. El usuario accede a la sección de gestión de las obras sociales de un colaborador en específico. 2. El sistema muestra la lista de obras sociales del colaborador. 3. El usuario puede realizar las siguientes acciones:    1. Agregar una nueva obra social al colaborador:       1. El usuario selecciona la opción de añadir una obra social al colaborador.       2. El sistema muestra una lista desplegable con todas las obras sociales.       3. El usuario selecciona una de las opciones de la lista.       4. El usuario guarda la obra social para el colaborador.    2. Editar la lista de obras sociales existente:       1. El usuario selecciona la opción de editar la lista de obras sociales existente.       2. El sistema muestra una lista desplegable con las diferentes obras sociales.       3. El usuario selecciona otra obra social.       4. El usuario guarda los cambios realizados.    3. Eliminar una obra social de la lista existente:       1. El usuario selecciona la opción de eliminar una obra social de la lista existente.       2. El sistema muestra una confirmación para asegurarse de la eliminación.       3. El usuario confirma la eliminación de la obra social de la lista. 4. El sistema actualiza la lista de obras sociales del colaborador después de realizar cualquier acción. |
| **Pos condición** | Se han creado, editado o eliminado obras sociales del colaborador según las acciones realizadas por el usuario. |
| **Permisos** | Colaboradores |
| **Tareas Necesarias** | 1 - Generar una clase capaz de almacenar los siguientes atributos:   |  |  |  |  | | --- | --- | --- | --- | | **ColaboradorObrasSociales** | | | | |  | Tipo | Obli | Observaciones | | IdColaboradorObrasSociales | Int32 | SI | Identificar único de la clase | | IdColaborador | Int32 | SI | Clave foránea referenciando al colaborador. | | IdObraSocial | Int32 | SI | Clave foránea referenciando a la obra social. | | FechaAlta | Date | NO | Fecha en la que al colaborador se asoció a la obra social. | | FechaBaja | Date | NO | Fecha en la que el colaborador dejo de ser socio de la obra social. |   2 – Generar una migración para la clase anterior  3 – Generar el CRUD para la clase definida en la tarea 1 |
| **Excepciones** |  |
| **Observaciones** | El campo fechaAlta por default es getDate(). |

## Colaborador Teléfonos

|  |  |
| --- | --- |
| **Gestión de Colaborador Teléfonos** | |
| **Nombre** | CRUD Colaborador Teléfonos |
| **Descripción** | Este caso de uso describe el proceso de gestión de la relación entre colaborador y sus respectivos teléfonos en el sistema. |
| **Actores** | Administrador del sistema / Accesos a Colaboradores |
| **Precondición** | El usuario debe tener permisos necesarios para gestionar los Colaboradores y además debe estar localizado en el administrador de colaboradores. |
| **Flujo Normal** | 1. El usuario accede a la sección de gestión de los teléfonos de un colaborador en específico. 2. El sistema muestra la lista de teléfonos del colaborador. 3. El usuario puede realizar las siguientes acciones:    1. Agregar un nuevo teléfono al colaborador:       1. El usuario selecciona la opción de añadir un teléfono al colaborador       2. El usuario completa los datos requeridos como número de teléfono.       3. El sistema muestra una lista desplegable con todos los tipos de teléfonos.       4. El usuario selecciona una de las opciones de la lista.       5. El usuario guarda el teléfono para el colaborador.    2. Editar la lista de teléfonos existente:       1. El usuario selecciona la opción de editar la lista de teléfonos existente.       2. El usuario modifica los campos como el número de teléfono.       3. El usuario guarda los cambios realizados.    3. Eliminar un teléfono de la lista existente:       1. El usuario selecciona la opción de eliminar un teléfono de la lista existente.       2. El sistema muestra una confirmación para asegurarse de la eliminación.       3. El usuario confirma la eliminación del teléfono de la lista. 4. El sistema actualiza la lista de teléfonos del colaborador después de realizar cualquier acción. |
| **Pos condición** | Se han creado, editado o eliminado teléfonos del colaborador según las acciones realizadas por el usuario. |
| **Permisos** | Colaboradores |
| **Tareas Necesarias** | 1 - Generar una clase capaz de almacenar los siguientes atributos:   |  |  |  |  | | --- | --- | --- | --- | | **ColaboradorTeléfonos** | | | | |  | Tipo | Obli | Observaciones | | IdColaboradorTeléfonos | Int32 | SI | Identificar único de la clase | | IdColaborador | Int32 | SI | Clave foránea referenciando al colaborador. | | IdTipoTeléfono | Int32 | SI | Clave foránea referenciando al tipo de teléfono. | | Teléfono | String | SI | Número de teléfono del colaborador | | Observaciones | Text | NO | Detalles a tener en cuenta |   2 – Generar una migración para la clase anterior  3 – Generar el CRUD para la clase definida en la tarea 1 |
| **Excepciones** |  |
| **Observaciones** |  |

## Colaborador Emails

|  |  |
| --- | --- |
| **Gestión de Colaborador Emails** | |
| **Nombre** | CRUD Colaborador Emails |
| **Descripción** | Este caso de uso describe el proceso de gestión de la relación entre colaborador y sus respectivos emails en el sistema. |
| **Actores** | Administrador del sistema / Accesos a Colaboradores |
| **Precondición** | El usuario debe tener permisos necesarios para gestionar los Colaboradores y además debe estar localizado en el administrador de colaboradores. |
| **Flujo Normal** | 1. El usuario accede a la sección de gestión de los emails de un colaborador en específico. 2. El sistema muestra la lista de emails del colaborador. 3. El usuario puede realizar las siguientes acciones:    1. Agregar un nuevo email al colaborador:       1. El usuario selecciona la opción de añadir un email al colaborador       2. El usuario completa los datos requeridos como la dirección del email.       3. El sistema muestra una lista desplegable con todos los tipos de emails.       4. El usuario selecciona una de las opciones de la lista.       5. El usuario guarda el email para el colaborador.    2. Editar la lista de emails existente:       1. El usuario selecciona la opción de editar la lista de emails existente.       2. El usuario modifica los campos como la dirección de email.       3. El usuario guarda los cambios realizados.    3. Eliminar un email de la lista existente:       1. El usuario selecciona la opción de eliminar un email de la lista existente.       2. El sistema muestra una confirmación para asegurarse de la eliminación.       3. El usuario confirma la eliminación del email de la lista. 4. El sistema actualiza la lista de emails del colaborador después de realizar cualquier acción. |
| **Pos condición** | Se han creado, editado o eliminado emails del colaborador según las acciones realizadas por el usuario. |
| **Permisos** | Colaboradores |
| **Tareas Necesarias** | 1 - Generar una clase capaz de almacenar los siguientes atributos:   |  |  |  |  | | --- | --- | --- | --- | | **ColaboradorEmails** | | | | |  | Tipo | Obli | Observaciones | | IdColaboradorEmails | Int32 | SI | Identificar único de la clase | | IdColaborador | Int32 | SI | Clave foránea referenciando al colaborador. | | IdTipoEmail | Int32 | SI | Clave foránea referenciando al tipo de email. | | Email | String(255) | SI | Dirección de email del colaborador | | Observaciones | Text | NO | Detalles a tener en cuenta |   2 – Generar una migración para la clase anterior  3 – Generar el CRUD para la clase definida en la tarea 1 |
| **Excepciones** |  |
| **Observaciones** |  |

## Colaborador Domicilios

|  |  |
| --- | --- |
| **Gestión de Colaborador Domicilios** | |
| **Nombre** | CRUD Colaborador Domicilios |
| **Descripción** | Este caso de uso describe el proceso de gestión de la relación entre colaborador y sus respectivos domicilios en el sistema. |
| **Actores** | Administrador del sistema / Accesos a Colaboradores |
| **Precondición** | El usuario debe tener permisos necesarios para gestionar los Colaboradores y además debe estar localizado en el administrador de colaboradores. |
| **Flujo Normal** | 1. El usuario accede a la sección de gestión de los domicilios de un colaborador en específico. 2. El sistema muestra la lista de domicilios del colaborador. 3. El usuario puede realizar las siguientes acciones:    1. Agregar un nuevo domicilio al colaborador:       1. El usuario selecciona la opción de añadir un domicilio al colaborador       2. El usuario completa los datos requeridos como la calle del domicilio.       3. El sistema muestra una lista desplegable con todos los tipos de domicilios.       4. El usuario selecciona una de las opciones de la lista.       5. El usuario guarda el domicilio para el colaborador.    2. Editar la lista de domicilios existente:       1. El usuario selecciona la opción de editar la lista de domicilios existente.       2. El usuario modifica los campos como la calle del domicilio.       3. El usuario guarda los cambios realizados.    3. Eliminar un domicilio de la lista existente:       1. El usuario selecciona la opción de eliminar un domicilio de la lista existente.       2. El sistema muestra una confirmación para asegurarse de la eliminación.       3. El usuario confirma la eliminación del domicilio de la lista. 4. El sistema actualiza la lista de domicilios del colaborador después de realizar cualquier acción. |
| **Pos condición** | Se han creado, editado o eliminado domicilios del colaborador según las acciones realizadas por el usuario. |
| **Permisos** | Colaboradores |
| **Tareas Necesarias** | 1 - Generar una clase capaz de almacenar los siguientes atributos:   |  |  |  |  | | --- | --- | --- | --- | | **ColaboradorDomicilios** | | | | |  | Tipo | Obli | Observaciones | | IdColaboradorDomicilios | Int32 | SI | Identificar único de la clase | | IdColaborador | Int32 | SI | Clave foránea referenciando al colaborador. | | IdTipoDomicilio | Int32 | SI | Clave foránea referenciando al tipo de domicilio. | | IdCodPostal | Int32 | SI | Clave foránea referenciando al código postal. | | Calle | String | SI | Nombre de la calle. | | Altura | String | SI | Altura del domicilio. | | Piso | String | NO | Piso del domicilio. | | Dpto | String | NO | Departamento del domicilio. | | Latitud | String | NO | Latitud del domicilio. | | Longitud | String | NO | Longitud del domicilio. | | OtraReferencia | String | NO | Otro dato relevante sobre la referencia a la ubicación del domicilio. | | Activo | Bool | SI | Por defecto verdadero. | | Observaciones | Text | NO | Datos relevantes. |   2 – Generar una migración para la clase anterior  3 – Generar el CRUD para la clase definida en la tarea 1 |
| **Excepciones** |  |
| **Observaciones** |  |

## Colaborador Entidades Bancarias

|  |  |
| --- | --- |
| **Gestión de Colaborador Entidades Bancarias** | |
| **Nombre** | CRUD Colaborador Entidades Bancarias |
| **Descripción** | Este caso de uso describe el proceso de gestión de la relación entre colaborador y sus respectivas entidades bancarias en el sistema. |
| **Actores** | Administrador del sistema / Accesos a Colaboradores |
| **Precondición** | El usuario debe tener permisos necesarios para gestionar los Colaboradores y además debe estar localizado en el administrador de colaboradores. |
| **Flujo Normal** | 1. El usuario accede a la sección de gestión de las entidades bancarias de un colaborador en específico. 2. El sistema muestra la lista de entidades bancarias del colaborador. 3. El usuario puede realizar las siguientes acciones:    1. Agregar una nueva entidad bancaria al colaborador:       1. El usuario selecciona la opción de añadir una entidad bancaria al colaborador       2. El sistema muestra una lista desplegable con las entidades bancarias cargadas en el sistema.       3. El usuario selecciona una de las opciones de la lista.       4. El usuario carga los datos requeridos como el CBU.       5. El usuario guarda la entidad bancaria para el colaborador.    2. Editar la lista de entidad bancarias existente:       1. El usuario selecciona la opción de editar una de las entidades bancarias del colaborador.       2. El usuario modifica los campos como el CBU.       3. El usuario guarda los cambios realizados.    3. Eliminar una entidad bancaria de la lista existente:       1. El usuario selecciona la opción de eliminar una entidad bancaria de la lista existente.       2. El sistema muestra una confirmación para asegurarse de la eliminación.       3. El usuario confirma la eliminación de la entidad bancaria de la lista. 4. El sistema actualiza la lista de entidad bancarias del colaborador después de realizar cualquier acción. |
| **Pos condición** | Se han creado, editado o eliminado entidades bancarias del colaborador según las acciones realizadas por el usuario. |
| **Permisos** | Colaboradores |
| **Tareas Necesarias** | 1 - Generar una clase capaz de almacenar los siguientes atributos:   |  |  |  |  | | --- | --- | --- | --- | | **ColaboradorEntidadesBancarias** | | | | |  | Tipo | Obli | Observaciones | | IdColaboradorEntidadesBancarias | Int32 | SI | Identificar único de la clase | | IdColaborador | Int32 | SI | Clave foránea referenciando al colaborador. | | IdTipoCuentaBancaria | Int32 | SI | Clave foránea referenciando al tipo de cuenta bancaria. | | IdEntidadBancaria | Int32 | SI | Clave foránea referenciando a la entidad bancaria. | | NumeroCuenta | String | SI | Número de la cuenta. | | CBU | String | SI | Número de CBU de la cuenta. | | Activo | Bool | SI | Por defecto verdadero. | | Observaciones | Text | NO | Datos relevantes. | | Alias | String | SI | Alias de la cuenta bancaria. | | Moneda | String | SI | Por defecto ARS. |   2 – Generar una migración para la clase anterior  3 – Generar el CRUD para la clase definida en la tarea 1 |
| **Excepciones** |  |
| **Observaciones** |  |

# Notificaciones

|  |  |
| --- | --- |
| **Gestión de Notificaciones** | |
| **Nombre** | CRUD Notificaciones |
| **Descripción** | Este caso de uso describe el proceso de gestión de las notificaciones del sistema. |
| **Actores** | Administrador del sistema |
| **Precondición** |  |
| **Flujo Normal** | 1. El usuario accede a la sección de gestión de las notificaciones. 2. El sistema muestra la lista de notificaciones. 3. El usuario puede realizar las siguientes acciones:    1. Agregar una nueva notificación para un colaborador, área de trabajo o empresa:       1. El usuario selecciona la opción de añadir una notificación.       2. El usuario completa los campos requeridos como mensaje.       3. El usuario guarda la nueva notificación.    2. Editar una notificación existente:       1. El usuario selecciona la notificación a editar.       2. El usuario modifica los campos como el mensaje.       3. El usuario guarda los cambios realizados.    3. Eliminar una entidad notificación de la lista existente:       1. El usuario selecciona la opción de eliminar una notificación de la lista existente.       2. El sistema muestra una confirmación para asegurarse de la eliminación.       3. El usuario confirma la eliminación de la notificación de la lista. 4. El sistema actualiza la lista de notificaciones después de realizar cualquier acción. |
| **Pos condición** | Se han creado, editado o eliminado notificaciones según las acciones realizadas por el usuario. |
| **Permisos** | Colaboradores |
| **Tareas Necesarias** | 1 - Generar una clase capaz de almacenar los siguientes atributos:   |  |  |  |  | | --- | --- | --- | --- | | **Notificaciones** | | | | |  | Tipo | Obli | Observaciones | | IdNotificacion | Int32 | SI | Identificar único de la clase | | IdAreaTrabajo | Int32 | SI | Clave foránea referenciando al área de trabajo. | | IdColaborador | Int32 | SI | Clave foránea referenciando al colaborador. | | Mensaje | String | SI | Mensaje de la notificación. | | FechaCreacion | Date | SI | Por defecto getDate(). | | FechaVencimiento | Date | NO | Fecha de cuando ya no se mostrará la notificación. | | Vigente | Bool | SI | Por defecto verdadero. | | AceptaNotificacion | Bool | SI | Por defecto false. | | FechaAceptacion | Date | NO | Fecha de cuando el usuario acepto la notificación. | | IdEmpresa | Int32 | SI | Empresa a la que pertenece la notificación. | | IdTennant | Int32 | NO | Indica el tennant al que pertenece. |   2 – Generar una migración para la clase anterior  3 – Generar el CRUD para la clase definida en la tarea 1 |
| **Excepciones** |  |
| **Observaciones** |  |

# Estados Novedades

|  |  |
| --- | --- |
| **Gestión de Estados de las Novedades** | |
| **Nombre** | CRUD Estados de las novedades |
| **Descripción** | Este caso de uso describe el proceso de gestión de estados de las novedades del sistema. |
| **Actores** | Administrador del sistema |
| **Precondición** | El usuario debe tener permisos necesarios para gestionar los Estados de novedades. |
| **Flujo Normal** | 1. El usuario accede a la sección de gestión de estado de novedades del sistema. 2. El sistema muestra la lista de estados de novedades del colaborador. 3. El usuario puede realizar las siguientes acciones:    1. Agregar un nuevo estado de novedad:       1. El usuario selecciona la opción de crear un nuevo estado de novedad.       2. El sistema muestra un formulario para ingresar los detalles del nuevo estado de novedad.       3. El usuario completa los campos requeridos, como la descripción del estado de novedad.       4. El usuario guarda el nuevo estado de novedad.    2. Editar estado de novedad existente:       1. El usuario selecciona la opción de editar uno de los estados de novedad.       2. El usuario modifica los campos como la descripción.       3. El usuario guarda los cambios realizados.    3. Eliminar un estado de novedad de la lista existente:       1. El usuario selecciona la opción de eliminar un estado de novedad de la lista existente.       2. El sistema muestra una confirmación para asegurarse de la eliminación.       3. El usuario confirma la eliminación del estado de novedad de la lista. 4. El sistema actualiza la lista de estado de novedades después de realizar cualquier acción. |
| **Pos condición** | Se han creado, editado o eliminado estados de novedades según las acciones realizadas por el usuario. |
| **Permisos** | Parámetros: Estados Novedades |
| **Tareas Necesarias** | 1 - Generar una clase capaz de almacenar los siguientes atributos:   |  |  |  |  | | --- | --- | --- | --- | | **Estados Novedades** | | | | |  | Tipo | Obli | Observaciones | | IdEstadoNovedad | Int32 | SI | Identificar único de la clase | | Descripcion | String | SI | Nombre del estado. | | Baja | Bool | SI | Por defecto false. | | Observaciones | String | NO | Detalles del estado. | | IdEmpresa | Int32 | SI | Clave foránea referenciando a la empresa. |   2 – Generar una migración para la clase anterior  3 – Generar el CRUD para la clase definida en la tarea 1 |
| **Excepciones** |  |
| **Observaciones** |  |

# Tipos de Novedades

|  |  |
| --- | --- |
| **Gestión de Tipos de Novedades** | |
| **Nombre** | CRUD TiposNovedades |
| **Descripción** | Este caso de uso describe el proceso de gestión de los tipos de novedades. |
| **Actores** | Administrador del sistema |
| **Precondición** | El usuario debe tener permisos necesarios para gestionar los Tipos de novedades. |
| **Flujo Normal** | 1. El usuario accede a la sección de gestión de tipos de novedades. 2. El sistema muestra la lista de tipos de novedades. 3. El usuario puede realizar las siguientes acciones:    1. Agregar un nuevo tipo de novedad:       1. El usuario selecciona la opción de añadir un nuevo tipo de novedad.       2. El usuario carga los datos requeridos como el nombre.       3. El usuario guarda el nuevo tipo de novedad.    2. Editar un tipo de novedad existente:       1. El usuario selecciona la opción de editar uno de los tipos de novedad.       2. El usuario modifica los campos como el nombre.       3. El usuario guarda los cambios realizados.    3. Eliminar un tipo de novedad existente:       1. El usuario selecciona la opción de eliminar un tipo de novedad de la lista existente.       2. El sistema muestra una confirmación para asegurarse de la eliminación.       3. El usuario confirma la eliminación del tipo de novedad de la lista. 4. El sistema actualiza la lista de tipos de novedades después de realizar cualquier acción. |
| **Pos condición** | Se han creado, editado o eliminado tipos de novedades según las acciones realizadas por el usuario. |
| **Permisos** | Novedades |
| **Tareas Necesarias** | 1 - Generar una clase capaz de almacenar los siguientes atributos:   |  |  |  |  | | --- | --- | --- | --- | | **TiposNovedades** | | | | |  | Tipo | Obli | Observaciones | | IdTipoNovedad | Int32 | SI | Identificar único de la clase | | NombreTipoNovedad | String | SI | Nombre de la novedad. | | Baja | Bool | SI | Por defecto false. | | IdEmpresa | Int32 | SI | Clave foránea referenciado a la empresa. |   2 – Generar una migración para la clase anterior  3 – Generar el CRUD para la clase definida en la tarea 1 |
| **Excepciones** |  |

## TipoNovedadEstados

|  |  |
| --- | --- |
| **Gestión de Tipo de novedades estados** | |
| **Nombre** | CRUD tiposNovedadesEstados |
| **Descripción** | Este caso de uso describe el proceso de gestión de los diferentes estados en los que puede estar un tipo de novedad. |
| **Actores** | Administrador del sistema |
| **Precondición** | El usuario debe tener permisos necesarios para gestionar los tipos de novedades. |
| **Flujo Normal** | 1. El usuario accede a la sección de gestión tipo estados novedad del sistema. 2. El sistema muestra la lista de estados posibles por cada tipo de novedad. 3. El usuario puede realizar las siguientes acciones:    1. Agregar un nuevo estado a un tipo de novedad.       1. El usuario selecciona la opción de añadir un nuevo estado.       2. El sistema muestra una lista desplegable con los estados de novedades cargados en el sistema.       3. El usuario selecciona una de las opciones de la lista.       4. El usuario guarda el nuevo estado para el tipo de novedad.    2. Editar la los estados de un tipo de novedad:       1. El usuario selecciona la opción de editar un tipo de estado de una novedad.       2. El usuario selecciona de la lista deplegable el nuevo estado.       3. El usuario guarda los cambios realizados.    3. Eliminar un tipo de estado de la lista existente:       1. El usuario selecciona la opción de eliminación de un tipo de estado de novedad de la lista existente.       2. El sistema muestra una confirmación para asegurarse de la eliminación.       3. El usuario confirma la eliminación del tipo de estado de novedad de la lista. 4. El sistema actualiza la lista de tipo de estados de novedad después de realizar cualquier acción. |
| **Pos condición** | Se han creado, editado o eliminado tipos de estados de novedades según las acciones realizadas por el usuario. |
| **Permisos** | Colaboradores |
| **Tareas Necesarias** | 1 - Generar una clase capaz de almacenar los siguientes atributos:   |  |  |  |  | | --- | --- | --- | --- | | **TipoNovedadEstados** | | | | |  | Tipo | Obli | Observaciones | | IdTipoNovedadEstados | Int32 | SI | Identificar único de la clase | | IdTipoNovedad | Int32 | SI | Clave foránea referenciando al tipo de novedad. | | IdEstadoNovedad | Int32 | SI | Clave foránea referenciando al estado de la novedad. |   2 – Generar una migración para la clase anterior  3 – Generar el CRUD para la clase definida en la tarea 1 |
| **Excepciones** |  |
| **Observaciones** |  |

# Novedades

|  |  |
| --- | --- |
| **Gestión de Novedades** | |
| **Nombre** | CRUD Novedades |
| **Descripción** | Este caso de uso describe el proceso de gestión de las novedades del sistema. |
| **Actores** | Administrador del sistema |
| **Precondición** |  |
| **Flujo Normal** | 1. El usuario accede a la sección de gestión novedades del sistema. 2. El sistema muestra la lista de novedades. 3. El usuario puede realizar las siguientes acciones:    1. Agregar una nueva novedad:       1. El usuario selecciona la opción de añadir una novedad al colaborador       2. El sistema muestra una lista desplegable tipos de novedades cargadas en el sistema.       3. El usuario selecciona una de las opciones de la lista.       4. El usuario guarda la nueva novedad.    2. Editar novedad de la lista existente:       1. El usuario selecciona la opción de editar una de las novedades.       2. El usuario modifica los campos.       3. El usuario guarda los cambios realizados.    3. Eliminar una novedad de la lista existente:       1. El usuario selecciona la opción de eliminar una novedad de la lista existente.       2. El sistema muestra una confirmación para asegurarse de la eliminación.       3. El usuario confirma la eliminación de la novedad de la lista. 4. El sistema actualiza la lista de novedades después de realizar cualquier acción. |
| **Pos condición** | Se han creado, editado o eliminado novedades según las acciones realizadas por el usuario. |
| **Permisos** | Novedades |
| **Tareas Necesarias** | 1 - Generar una clase capaz de almacenar los siguientes atributos:   |  |  |  |  | | --- | --- | --- | --- | | **Novedades** | | | | |  | Tipo | Obli | Observaciones | | IdNovedad | Int32 | SI | Identificar único de la clase | | IdColaborador | Int32 | SI | Clave foránea referenciando al colaborador. | | FechaSolicitud | Date | SI | Por defecto getDate(). | | FechaDesde | Date | SI | Fecha desde cuando tendría efecto la novedad. | | FechaHasta | Date | SI | Fecha hasta cuando tendría efecto la novedad | | IdTipoNovedad | Int32 | SI | Clave foránea referenciando al tipo de novedad. | | IdEstadoNovedad | Int32 | SI | Clave foránea referenciado al estado de la novedad. | | DocumentosRelacionados | String | NO | Documentos guardados. | | ImagenesRelacionadas | String | NO | Imágenes guardadas. | | Observaciones | String | NO | Detalles relevantes. | | IdColaborador | Int32 | SI | Clave foránea referenciando al colaborador de la novedad. | | Cacelada | Bool | SI | Por defecto false. | | IdEmpresa | Int32 | SI | Clave foránea referenciando a la empresa. |   2 – Generar una migración para la clase anterior  3 – Generar el CRUD para la clase definida en la tarea 1 |
| **Excepciones** |  |
| **Observaciones** |  |

## Novedad Estados

|  |  |
| --- | --- |
| **Gestión de Novedades** | |
| **Nombre** | CRUD Novedad Estados |
| **Descripción** | Este caso de uso describe el proceso de gestión de los diferentes estados que puede estar una novedad. |
| **Actores** | Administrador del sistema |
| **Precondición** |  |
| **Flujo Normal** | 1. El usuario accede a la sección de gestión de estados de novedades del sistema. 2. El sistema muestra la lista de estados de novedades. 3. El usuario puede realizar las siguientes acciones:    1. Agregar un nuevo estado:       1. El usuario selecciona la opción de añadir un nuevo estado.       2. El usuario carga los datos requeridos como el nombre.       3. El usuario guarda el nuevo estado.    2. Editar la lista de estados existente:       1. El usuario selecciona la opción de editar un estado de la lista.       2. El usuario modifica los campos como el nombre.       3. El usuario guarda los cambios realizados.    3. Eliminar un estado de la lista existente:       1. El usuario selecciona la opción de eliminar un estado de la lista existente.       2. El sistema muestra una confirmación para asegurarse de la eliminación.       3. El usuario confirma la eliminación del estado de la lista. 4. El sistema actualiza la lista de astados después de realizar cualquier acción. |
| **Pos condición** | Se han creado, editado o eliminado estados según las acciones realizadas por el usuario. |
| **Permisos** | Estados novedades |
| **Tareas Necesarias** | 1 - Generar una clase capaz de almacenar los siguientes atributos:   |  |  |  |  | | --- | --- | --- | --- | | **NovedadesEstados** | | | | |  | Tipo | Obli | Observaciones | | IdNovedadEstados | Int32 | SI | Identificar único de la clase | | IdNovedad | Int32 | SI | Clave foránea referenciando a la novedad. | | IdEstadoNovedad | Int32 | SI | Clave foránea referenciando al estado de la novedad. | | FechaCambio | Date | NO | Fecha del último cambio de estado. | | IdColaborador | Int32 | SI | Clave foránea referenciando al colaborador. | | Justificacion | String | SI | Detalles de porque se modificó el estado de la novedad. | | FechaModificacion | Date | NO | Fecha de la última actualización. |   2 – Generar una migración para la clase anterior  3 – Generar el CRUD para la clase definida en la tarea 1 |
| **Excepciones** |  |
| **Observaciones** |  |