Sistema de Trámite Documentario

Documento de Microdiseño

Índice

[1. Descripción General de funcionalidades 3](#_Toc370457159)

[1.1. Gestionar tramite 3](#_Toc370457160)

[1.2. Registrar requisitos 3](#_Toc370457161)

[1.3. Gestionar requisitos 4](#_Toc370457162)

[1.4. Registrar Informe 5](#_Toc370457163)

[1.5. Gestionar informes 5](#_Toc370457164)

[1.6. Validar usuario 6](#_Toc370457165)

[1.7. Registrar usuarios 7](#_Toc370457166)

[1.8. Gestionar usuarios 7](#_Toc370457167)

[1.9. Consultar reportes 8](#_Toc370457168)

[2. Detalle de Implementación a nivel de Modelos 9](#_Toc370457169)

[3. Detalle de Implementación a nivel de Controladores 14](#_Toc370457170)

[4. Detalle de Implementación a nivel de Vistas 15](#_Toc370457171)

[5. Detalle de Implementación a nivel de Tablas 16](#_Toc370457172)

## Descripción General de funcionalidades

El presente documento esta designado a la elaboración de las funcionalidades del sistema de trámite documentario.

## Gestionar tramite

|  |  |  |  |
| --- | --- | --- | --- |
| **Elaborado Por:** |  | | |
| **Descripción General:** | Gestionar tramite | | |
| Esta funcionalidad tiene como finalidad realizar consultas, actualizaciones y eliminaciones de trámites a la base de datos. | | |
| **Referencias** | Requerimientos: | Especificación del caso de Uso Gestionar Tramite | |
| STD\_GES\_TRA.docx | |
| Diseño de Interfaz Usuario: | Especificación del caso Gestionar Tramite. | |
| STD\_GES\_TRA.docx | |
| Diseño de Arquitectura: | Diagrama de Arquitectura del software | |
| STD\_DAS.docx | |
| Otros: | Matriz de trazabilidad de casos de uso vs. Requerimientos funcionales STD\_MATR\_TRAZ.xlsx | |
| Documento de Diseño | |
| STD\_ANA\_DOC.docx | |
| **Clases Requeridas:** | **Nombre de la Clase** | **Tipo – Stereotype** | **Acción** |
| tramites.java | Model | Lectura, Escritura |
| ctramites.java | Controller | Lectura, Escritura |
| registratramites.jsp | View | Lectura, Escritura |
| **Tablas:** | **Nombre de la tabla** | | **Tipo de Objeto** |
| tb\_std\_tramites | | Tabla |

## Registrar requisitos

|  |  |  |  |
| --- | --- | --- | --- |
| **Elaborado Por:** |  | | |
| **Descripción General:** | Registrar requisitos | | |
| Esta funcionalidad tiene como finalidad realizar el registro de datos de los requisitos de un tramite. | | |
| **Referencias** | Requerimientos: | Especificación del caso de Uso Registrar requisitos. | |
| STD\_REG\_REQ.docx | |
| Diseño de Interfaz Usuario: | Especificación del caso Gestionar Tramite. | |
| STD\_REG\_REQ.docx | |
| Diseño de Arquitectura: | Diagrama de Arquitectura del software | |
| STD\_DAS.docx | |
| Otros: | Matriz de trazabilidad de casos de uso vs. Requerimientos funcionales STD\_MATR\_TRAZ.xlsx | |
| Documento de Diseño | |
| STD\_ANA\_DOC.docx | |
| **Clases Requeridas:** | **Nombre de la Clase** | **Tipo – Stereotype** | **Acción** |
| requisitos.java | Model | Escritura |
| crequisitos.java | Controller | Escritura |
| registrarequisitos.jsp | View | Escritura |
| **Tablas:** | **Nombre de la tabla** | | **Tipo de Objeto** |
| tb\_std\_requisitos | | Tabla |

## Gestionar requisitos

|  |  |  |  |
| --- | --- | --- | --- |
| **Elaborado Por:** |  | | |
| **Descripción General:** | Gestionar requisitos | | |
| Esta funcionalidad tiene como finalidad realizar consultas, actualizaciones y eliminaciones de requisitos a la base de datos. | | |
| **Referencias** | Requerimientos: | Especificación del caso de Uso Gestionar Requisitos | |
| STD\_GES\_REQ.docx | |
| Diseño de Interfaz Usuario: | Especificación del caso Gestionar Requisitos. | |
| STD\_GES\_REQ.docx | |
| Diseño de Arquitectura: | Diagrama de Arquitectura del software | |
| STD\_DAS.docx | |
| Otros: | Matriz de trazabilidad de casos de uso vs. Requerimientos funcionales STD\_MATR\_TRAZ.xlsx | |
| Documento de Diseño | |
| STD\_ANA\_DOC.docx | |
| **Clases Requeridas:** | **Nombre de la Clase** | **Tipo – Stereotype** | **Acción** |
| requisitos.java | Model | Lectura, Escritura |
| crequisitos.java | Controller | Lectura, Escritura |
| gestionarequisitos.jsp | View | Lectura, Escritura |
| **Tablas:** | **Nombre de la tabla** | | **Tipo de Objeto** |
| tb\_std\_requisitos | | Tabla |

## Registrar Informe

|  |  |  |  |
| --- | --- | --- | --- |
| **Elaborado Por:** |  | | |
| **Descripción General:** | Registrar Informe | | |
| Esta funcionalidad tiene como finalidad realizar registros de los expedientes a la base de datos. | | |
| **Referencias** | Requerimientos: | Especificación del caso de Uso Registra informe | |
| STD\_REG\_INF.docx | |
| Diseño de Interfaz Usuario: | Especificación del caso Registrar Informe. | |
| STD\_REG\_INF.docx | |
| Diseño de Arquitectura: | Diagrama de Arquitectura del software | |
| STD\_DAS.docx | |
| Otros: | Matriz de trazabilidad de casos de uso vs. Requerimientos funcionales STD\_MATR\_TRAZ.xlsx | |
| Documento de Diseño | |
| STD\_ANA\_DOC.docx | |
| **Clases Requeridas:** | **Nombre de la Clase** | **Tipo – Stereotype** | **Acción** |
| informe.java | Model | Escritura |
| cinforme.java | Controller | Escritura |
| registrainforme.java | View | Escritura |
| **Tablas:** | **Nombre de la tabla** | | **Tipo de Objeto** |
| tb\_std\_expediente | | Tabla |

## Gestionar informes

|  |  |  |  |
| --- | --- | --- | --- |
| **Elaborado Por:** |  | | |
| **Descripción General:** | Gestionar informes | | |
| Esta funcionalidad tiene como finalidad realizar consultas, actualizaciones y eliminaciones de informe a la base de datos. | | |
| **Referencias** | Requerimientos: | Especificación del caso de Uso Gestionar Informes | |
| STD\_GES\_INF.docx | |
| Diseño de Interfaz Usuario: | Especificación del caso Gestionar Tramite. | |
| STD\_GES\_INF.docx | |
| Diseño de Arquitectura: | Diagrama de Arquitectura del software | |
| STD\_DAS.docx | |
| Otros: | Matriz de trazabilidad de casos de uso vs. Requerimientos funcionales STD\_MATR\_TRAZ.xlsx | |
| Documento de Diseño | |
| STD\_ANA\_DOC.docx | |
| **Clases Requeridas:** | **Nombre de la Clase** | **Tipo – Stereotype** | **Acción** |
| informes.java | Model | Lectura, Escritura |
| cinformes.java | Controller | Lectura, Escritura |
| gestioninformes.java | View | Lectura, Escritura |
| **Tablas:** | **Nombre de la tabla** | | **Tipo de Objeto** |
| tb\_std\_expediente | | Tabla |

## Validar usuario

|  |  |  |  |
| --- | --- | --- | --- |
| **Elaborado Por:** |  | | |
| **Descripción General:** | Validar Usuario | | |
| Esta funcionalidad tiene como finalidad realizar la validación de usuarios. | | |
| **Referencias** | Requerimientos: | Especificación del caso de Uso Validar usuario | |
| STD\_VAL\_USU.docx | |
| Diseño de Interfaz Usuario: | Especificación del caso Gestionar Tramite. | |
| STD\_VAL\_USU.docx | |
| Diseño de Arquitectura: | Diagrama de Arquitectura del software | |
| STD\_DAS.docx | |
| Otros: | Matriz de trazabilidad de casos de uso vs. Requerimientos funcionales STD\_MATR\_TRAZ.xlsx | |
| Documento de Diseño | |
| STD\_ANA\_DOC.docx | |
| **Clases Requeridas:** | **Nombre de la Clase** | **Tipo – Stereotype** | **Acción** |
| usuarios.java | Model | Lectura |
| cusuarioss.java | Controller | Lectura |
| usuarios.jsp | View | Lectura |
| **Tablas:** | **Nombre de la tabla** | | **Tipo de Objeto** |
| tb\_std\_usuario | | Tabla |

## Registrar usuarios

|  |  |  |  |
| --- | --- | --- | --- |
| **Elaborado Por:** |  | | |
| **Descripción General:** | Registrar usuarios | | |
| Esta funcionalidad tiene como finalidad realizar el registro de usuarios a la base de datos. | | |
| **Referencias** | Requerimientos: | Especificación del caso de Uso Registrar Usuarios | |
| STD\_REG\_USU.docx | |
| Diseño de Interfaz Usuario: | Especificación del caso Gestionar Tramite. | |
| STD\_REG\_USU.docx | |
| Diseño de Arquitectura: | Diagrama de Arquitectura del software | |
| STD\_DAS.docx | |
| Otros: | Matriz de trazabilidad de casos de uso vs. Requerimientos funcionales STD\_MATR\_TRAZ.xlsx | |
| Documento de Diseño | |
| STD\_ANA\_DOC.docx | |
| **Clases Requeridas:** | **Nombre de la Clase** | **Tipo – Stereotype** | **Acción** |
| usuarios.java | Model | Escritura |
| cusuarios.java | Controller | Escritura |
| registrausuarios.java | View | Escritura |
| **Tablas:** | **Nombre de la tabla** | | **Tipo de Objeto** |
| tb\_std\_usuario | | Tabla |

## Gestionar usuarios

|  |  |  |  |
| --- | --- | --- | --- |
| **Elaborado Por:** |  | | |
| **Descripción General:** | Gestionar usuarios | | |
| Esta funcionalidad tiene como finalidad realizar consultas, actualizaciones y eliminaciones de usuarios a la base de datos. | | |
| **Referencias** | Requerimientos: | Especificación del caso de Uso Gestionar Usuarios | |
| STD\_GES\_USU.docx | |
| Diseño de Interfaz Usuario: | Especificación del caso Gestionar Usuarios. | |
| STD\_GES\_USU.docx | |
| Diseño de Arquitectura: | Diagrama de Arquitectura del software | |
| STD\_DAS.docx | |
| Otros: | Matriz de trazabilidad de casos de uso vs. Requerimientos funcionales STD\_MATR\_TRAZ.xlsx | |
| Documento de Diseño | |
| STD\_ANA\_DOC.docx | |
| **Clases Requeridas:** | **Nombre de la Clase** | **Tipo – Stereotype** | **Acción** |
| usuarios.java | Model | Escritura |
| cusuarios.java | Controller | Escritura |
| gestionausuarios.java | View | Escritura |
| **Tablas:** | **Nombre de la tabla** | | **Tipo de Objeto** |
| tb\_std\_usuario | | Tabla |

## Consultar reportes

|  |  |  |  |
| --- | --- | --- | --- |
| **Elaborado Por:** |  | | |
| **Descripción General:** | Consulta reporte | | |
| Esta funcionalidad tiene como finalidad realizar consultas estadísticas de los trámites. | | |
| **Referencias** | Requerimientos: | Especificación del caso de Uso Consultar reportes | |
| STD\_CON\_REP.docx | |
| Diseño de Interfaz Usuario: | Especificación del caso Gestionar Tramite. | |
| STD\_CON\_REP.docx | |
| Diseño de Arquitectura: | Diagrama de Arquitectura del software | |
| STD\_DAS.docx | |
| Otros: | Matriz de trazabilidad de casos de uso vs. Requerimientos funcionales STD\_MATR\_TRAZ.xlsx | |
| Documento de Diseño | |
| STD\_ANA\_DOC.docx | |
| **Clases Requeridas:** | **Nombre de la Clase** | **Tipo – Stereotype** | **Acción** |
| tramites.java | Model | Escritura |
| ctramites.java | Controller | Escritura |
| registratramites.java | View | Escritura |
| **Tablas:** | **Nombre de la tabla** | | **Tipo de Objeto** |
| Tramite | | Tabla |

## Detalle de Implementación a nivel de Modelos

|  |  |
| --- | --- |
| **Modelo** | **Descripción** |
| tramites.java | Clase persistente utiliza para hacer persistente a la entidad trámites. |
| requisitos.java | Clase persistente utiliza para hacer persistente a la entidad requisitos. |
| areas.java | Clase persistente utiliza para hacer persistente a la entidad áreas. |
| expedientes.java | Clase persistente utiliza para hacer persistente a la entidad expedientes. |
| alumnos.java | Clase persistente utiliza para hacer persistente a la entidad alumnos. |
| usuarios.java | Clase persistente utiliza para hacer persistente a la entidad usuarios. |
| perfil.java | Clase persistente utiliza para hacer persistente a la entidad perfil. |

**tramites.java**

* **Ruta:**

app/models/.java

* **Descripción:**

Clase de modelo utilizada para hacer persistencia a la entidad **trámites,** extendiendo de la claseModel,debe ser declarada como una entidad, y poseer relación con la tabla *“tb\_std\_tramites”*:

1. @Entity
2. @Table(name = "tramites")
3. public class Tramites extends Model

* **Atributos:**

Dado que se trata de una clase entidad de playFramework, los atributos deben poseer una referencia a una columna de la tabla “tb\_std\_tramites”:

Para los atributos que tienen primary\_key en la base de datos:

1. @Requiered
2. @Colum(name = "nombre\_atributo", nullable = false)
3. public tipo atributo;

Para los atributos que no posean primary\_key en la base de datos:

1. @Colum(name = "nombre\_atributo", nullable = true)
2. public tipo atributo;

* **Métodos:**

Debe poseer el método que sobrescriba el método toString() para mostrar el código del almacén y su nombre referente al mismo:

1. @Override
2. public String toString()

**requisitos.java**

* **Ruta:**

app/models/.java

* **Descripción:**

Clase de modelo utilizada para hacer persistencia a la entidad **requisitos,** extendiendo de la claseModel,debe ser declarada como una entidad, y poseer relación con la tabla *“tb\_std\_requisitos”*:

1. @Entity
2. @Table(name = "requisitos")
3. public class Requisitos extends Model

* **Atributos:**

Dado que se trata de una clase entidad de playFramework, los atributos deben poseer una referencia a una columna de la tabla “tb\_std\_requisitos”:

Para los atributos que tienen primary\_key en la base de datos:

1. @Requiered
2. @Colum(name = "nombre\_atributo", nullable = false)
3. public tipo atributo;

Para los atributos que no posean primary\_key en la base de datos:

1. @Colum(name = "nombre\_atributo", nullable = true)
2. public tipo atributo;

* **Métodos:**

Debe poseer el método que sobrescriba el método toString() para mostrar el código del almacén y su nombre referente al mismo:

1. @Override
2. public String toString()

**areas.java**

* **Ruta:**

app/models/.java

* **Descripción:**

Clase de modelo utilizada para hacer persistencia a la entidad **areas,** extendiendo de la claseModel,debe ser declarada como una entidad, y poseer relación con la tabla *“tb\_std\_areas”*:

1. @Entity
2. @Table(name = "areas")
3. public class Areas extends Model

* **Atributos:**

Dado que se trata de una clase entidad de playFramework, los atributos deben poseer una referencia a una columna de la tabla “tb\_std\_areas”:

Para los atributos que tienen primary\_key en la base de datos:

1. @Requiered
2. @Colum(name = "nombre\_atributo", nullable = false)
3. public tipo atributo;

Para los atributos que no posean primary\_key en la base de datos:

1. @Colum(name = "nombre\_atributo", nullable = true)
2. public tipo atributo;

* **Métodos:**

Debe poseer el método que sobrescriba el método toString() para mostrar el código del almacén y su nombre referente al mismo:

1. @Override
2. public String toString()

**expedientes.java**

* **Ruta:**

app/models/.java

* **Descripción:**

Clase de modelo utilizada para hacer persistencia a la entidad **expedientes,** extendiendo de la claseModel,debe ser declarada como una entidad, y poseer relación con la tabla *“tb\_std\_expedientes”*:

1. @Entity
2. @Table(name = "expedientes")
3. public class Expedientes extends Model

* **Atributos:**

Dado que se trata de una clase entidad de playFramework, los atributos deben poseer una referencia a una columna de la tabla “tb\_std\_expedientes”:

Para los atributos que tienen primary\_key en la base de datos:

1. @Requiered
2. @Colum(name = "nombre\_atributo", nullable = false)
3. public tipo atributo;

Para los atributos que no posean primary\_key en la base de datos:

1. @Colum(name = "nombre\_atributo", nullable = true)
2. public tipo atributo;

* **Métodos:**

Debe poseer el método que sobrescriba el método toString() para mostrar el código del almacén y su nombre referente al mismo:

1. @Override
2. public String toString()

**usuarios.java**

* **Ruta:**

app/models/.java

* **Descripción:**

Clase de modelo utilizada para hacer persistencia a la entidad **usuarios,** extendiendo de la claseModel,debe ser declarada como una entidad, y poseer relación con la tabla *“tb\_std\_usuarios”*:

1. @Entity
2. @Table(name = "usuarios")
3. public class Usuarios extends Model

* **Atributos:**

Dado que se trata de una clase entidad de playFramework, los atributos deben poseer una referencia a una columna de la tabla “tb\_std\_usuarios”:

Para los atributos que tienen primary\_key en la base de datos:

1. @Requiered
2. @Colum(name = "nombre\_atributo", nullable = false)
3. public tipo atributo;

Para los atributos que no posean primary\_key en la base de datos:

1. @Colum(name = "nombre\_atributo", nullable = true)
2. public tipo atributo;

* **Métodos:**

Debe poseer el método que sobrescriba el método toString() para mostrar el código del almacén y su nombre referente al mismo:

1. @Override
2. public String toString()

**perfil.java**

* **Ruta:**

app/models/.java

* **Descripción:**

Clase de modelo utilizada para hacer persistencia a la entidad **perfil,** extendiendo de la claseModel,debe ser declarada como una entidad, y poseer relación con la tabla *“tb\_std\_perfil”*:

1. @Entity
2. @Table(name = "perfil")
3. public class Perfil extends Model

* **Atributos:**

Dado que se trata de una clase entidad de playFramework, los atributos deben poseer una referencia a una columna de la tabla “tb\_std\_perfil”:

Para los atributos que tienen primary\_key en la base de datos:

1. @Requiered
2. @Colum(name = "nombre\_atributo", nullable = false)
3. public tipo atributo;

Para los atributos que no posean primary\_key en la base de datos:

1. @Colum(name = "nombre\_atributo", nullable = true)
2. public tipo atributo;

* **Métodos:**

Debe poseer el método que sobrescriba el método toString() para mostrar el código del almacén y su nombre referente al mismo:

1. @Override
2. public String toString()

**alumnos.java**

* **Ruta:**

app/models/.java

* **Descripción:**

Clase de modelo utilizada para hacer persistencia a la entidad **alumnos,** extendiendo de la claseModel,debe ser declarada como una entidad, y poseer relación con la tabla *“tb\_std\_alumnos”*:

1. @Entity
2. @Table(name = "alumnos")
3. public class Alumnos extends Model

* **Atributos:**

Dado que se trata de una clase entidad de playFramework, los atributos deben poseer una referencia a una columna de la tabla “tb\_std\_alumnos”:

Para los atributos que tienen primary\_key en la base de datos:

1. @Requiered
2. @Colum(name = "nombre\_atributo", nullable = false)
3. public tipo atributo;

Para los atributos que no posean primary\_key en la base de datos:

1. @Colum(name = "nombre\_atributo", nullable = true)
2. public tipo atributo;

* **Métodos:**

Debe poseer el método que sobrescriba el método toString() para mostrar el código del almacén y su nombre referente al mismo:

1. @Override
2. public String toString()

## Detalle de Implementación a nivel de Controladores

|  |  |
| --- | --- |
| **Controlador** | **Descripción** |
| ctramites.java | Clase controlador la cual permite hacer el proceso gestionar los datos de trámites. |
| crequisitos.java | Clase controlador la cual permite hacer el proceso gestionar los datos de requisitos. |
| careas.java | Clase controlador la cual permite hacer el proceso gestionar los datos de áreas. |
| cexpedientes.java | Clase controlador la cual permite hacer el proceso gestionar los datos de expedientes. |
| calumnos.java | Clase controlador la cual permite hacer el proceso gestionar los datos de alumnos. |
| cusuarios.java | Clase controlador la cual permite hacer el proceso gestionar los datos de trámites. |
| cperfil.java | Clase controlador la cual permite hacer el proceso gestionar los datos de los perfiles. |

**ctramites.java**

* **Ruta:**

app/controllers/ctramites.java

* **Descripción:**

Clase controlador la cual permite hacer el proceso del negocio, recuperar la información de un tramite.

* **Métodos:**

Debe de habilitar la vista del formulario de gestión y registro de trámites.

public static void tramites()

**crequisitos.java**

* **Ruta:**

app/controllers/crequisitos.java

* **Descripción:**

Clase controlador la cual permite hacer el proceso del negocio, recuperar la información de un requisitos.

* **Métodos:**

Debe de habilitar la vista del formulario de gestión y registro de requisitos.

public static void tramites()

**cexpedientes.java**

* **Ruta:**

app/controllers/cexpedientes.java

* **Descripción:**

Clase controlador la cual permite hacer el proceso del negocio, recuperar la información de un expediente.

* **Métodos:**

Debe de habilitar la vista del formulario de gestión y registro de expedientes

public static void tramites()

## Detalle de Implementación a nivel de Vistas

|  |  |
| --- | --- |
| **Vista** | **Descripción** |
| registrotramites.html | Objeto de vista que registra datos en la entidad tramite. |
| gestiontramites.html | Objeto de vista que gestiona, muestra, actualiza, elimina, datos en la entidad tramite. |
| registrorequisitos.html | Objeto de vista que registra datos en la entidad tramite. |
| gestionrequisitos.html | Objeto de vista que gestiona, muestra, actualiza, elimina, datos en la entidad tramite. |
| registroareas.html | Objeto de vista que registra datos en la entidad tramite. |
| gestionareas.html | Objeto de vista que gestiona, muestra, actualiza, elimina, datos en la entidad tramite. |
| registroexpedientes.html | Objeto de vista que registra datos en la entidad tramite. |
| gestionexpedientes.html | Objeto de vista que gestiona, muestra, actualiza, elimina, datos en la entidad tramite. |
| registroalumnos.html | Objeto de vista que registra datos en la entidad tramite. |
| gestionalumnos.html | Objeto de vista que gestiona, muestra, actualiza, elimina, datos en la entidad tramite. |
| usuarios.html | Objeto de vista que registra datos en la entidad tramite. |
| registrousuarios.html | Objeto de vista que gestiona, muestra, actualiza, elimina, datos en la entidad tramite. |

**Registrotramites.list.html**

* **Ruta:**

app/views/productos/list.html

* **Descripción:**

Objeto de vista, extiende del layout ‘CRUD/layout.html’, muestra la búsqueda de una entidad *Producto*, debe de contar de un campo de entrada numérica y un botón envié el valor recibido como parámetro y haga referencia al método del controlador *CProducto*:

1. href = "@{CProducto.producto()}"

**mostrar.html**

* **Ruta**:

app/views/cproducto/mostrar.html

* **Descripción:**

Objeto de vista, extiende del layout ‘main.html’, donde se muestra la información encontrada por el controlador CProducto, la cual deberá tener la siguiente especificación:

1. Código
2. Tipo
3. Marca
4. Modelo
5. Precio (S/.)
6. Punto de reposición
7. Código del almacén donde se encuentra

Para cada atributo del producto listado con anterioridad se declarará:

1. <input type =”text” name=“producto.atributo” id =”atributo” value=”${product.atributo}”>

Para el atributo, ‘código del almacén’ del almacén de la lista de almacenes que concuerde con uno el del almacén del producto se especificará:

<option value="${almacen.codAlmacen}" > ${almacen.toString()} </option>

## Detalle de Implementación a nivel de Tablas

-- ------------------------------------------------------------

-- Tabla que almacena los datos los privilegios de acceso en el sistema.

-- ------------------------------------------------------------

CREATE TABLE tb\_std\_perfil (

idtb\_std\_perfil INTEGER NOT NULL ,

nombre\_perfil VARCHAR ,

estado\_perfil INTEGER ,

PRIMARY KEY(idtb\_std\_perfil));

COMMENT ON TABLE tb\_std\_perfil IS 'Tabla que almacena los datos los priviliegios de acceso en el sistema.';

-- ------------------------------------------------------------

-- Tabla que almacena los datos de los requisitos para realizar un tramite.

-- ------------------------------------------------------------

CREATE TABLE tb\_std\_requisitos (

idtb\_std\_requisito INTEGER NOT NULL ,

nombre\_requisito VARCHAR ,

costo\_requisito DECIMAL ,

fecha\_registro DATETIME ,

observacion\_requisito VARCHAR ,

PRIMARY KEY(idtb\_std\_requisito));

COMMENT ON TABLE tb\_std\_requisitos IS 'Tabla que almacena los datos de los requisitos para realizar un tramite.';

-- ------------------------------------------------------------

-- Tabla que almacena los datos de los estados de los tramites.

-- ------------------------------------------------------------

CREATE TABLE tb\_std\_estado\_tramites (

idtb\_std\_estado\_tramite INTEGER NOT NULL ,

nombre\_estadotramite VARCHAR ,

PRIMARY KEY(idtb\_std\_estado\_tramite));

COMMENT ON TABLE tb\_std\_estado\_tramites IS 'Tabla que almacena los datos de los estados de los tramites.';

-- ------------------------------------------------------------

-- Tabla que almacena los datos de los alumnos que inician un trámite en la unidad de Postgrado.

-- ------------------------------------------------------------

CREATE TABLE tb\_std\_alumnos (

idtb\_std\_alumno INTEGER NOT NULL ,

codigo\_alumno INTEGER ,

nombre\_alumno VARCHAR ,

apellidopaterno\_alumno VARCHAR ,

apellidomaterno\_alumno VARCHAR ,

correo\_electronico VARCHAR ,

PRIMARY KEY(idtb\_std\_alumno));

COMMENT ON TABLE tb\_std\_alumnos IS 'Tabla que almacena los datos de los alumnos que inician un tramite en la unidad de Postgrado.';

-- ------------------------------------------------------------

-- Tabla que almacena los datos de las areas de la Unidad de Postgrado.

-- ------------------------------------------------------------

CREATE TABLE tb\_std\_areas (

idtb\_std\_area INTEGER NOT NULL ,

nombre\_area VARCHAR ,

observacion\_area INTEGER ,

PRIMARY KEY(idtb\_std\_area));

COMMENT ON TABLE tb\_std\_areas IS 'Tabla que almacena los datos de las areas de la Unidad de Postgrado.';

-- ------------------------------------------------------------

-- Tabla que almacena los datos de los usuarios, personal administrativo, de las areas de la unidad de Postgrado

-- ------------------------------------------------------------

CREATE TABLE tb\_std\_usuarios (

idtb\_std\_usuario INTEGER NOT NULL ,

tb\_std\_perfil\_idtb\_std\_perfil INTEGER NOT NULL ,

nombre\_usuario VARCHAR ,

password\_usuario VARCHAR ,

correo\_electronico VARCHAR ,

fecha\_registro DATETIME ,

PRIMARY KEY(idtb\_std\_usuario) ,

FOREIGN KEY(tb\_std\_perfil\_idtb\_std\_perfil)

REFERENCES tb\_std\_perfil(idtb\_std\_perfil));

CREATE INDEX tb\_std\_usuarios\_FKIndex1 ON tb\_std\_usuarios (tb\_std\_perfil\_idtb\_std\_perfil);

COMMENT ON TABLE tb\_std\_usuarios IS 'Tabla que almacena los datos de los usuarios, personal administrativo, de las areas de la unidad de Postgrado';

CREATE INDEX IFK\_Rel\_02 ON tb\_std\_usuarios (tb\_std\_perfil\_idtb\_std\_perfil);

-- ------------------------------------------------------------

-- Tabla que almacena los datos de los tramites de la Unidad de Postgrado.

-- ------------------------------------------------------------

CREATE TABLE tb\_std\_tramites (

idtb\_std\_tramite INTEGER NOT NULL ,

tb\_std\_usuarios\_idtb\_std\_usuario INTEGER NOT NULL ,

tb\_std\_alumnos\_idtb\_std\_alumno INTEGER NOT NULL ,

tb\_std\_areas\_idtb\_std\_area INTEGER NOT NULL ,

tb\_std\_estado\_tramites\_idtb\_std\_estado\_tramite INTEGER NOT NULL ,

nombre\_tramite VARCHAR ,

fecha\_registro DATETIME ,

PRIMARY KEY(idtb\_std\_tramite) ,

FOREIGN KEY(tb\_std\_estado\_tramites\_idtb\_std\_estado\_tramite)

REFERENCES tb\_std\_estado\_tramites(idtb\_std\_estado\_tramite),

FOREIGN KEY(tb\_std\_areas\_idtb\_std\_area)

REFERENCES tb\_std\_areas(idtb\_std\_area),

FOREIGN KEY(tb\_std\_alumnos\_idtb\_std\_alumno)

REFERENCES tb\_std\_alumnos(idtb\_std\_alumno),

FOREIGN KEY(tb\_std\_usuarios\_idtb\_std\_usuario)

REFERENCES tb\_std\_usuarios(idtb\_std\_usuario));

CREATE INDEX tb\_std\_tramites\_FKIndex1 ON tb\_std\_tramites (tb\_std\_estado\_tramites\_idtb\_std\_estado\_tramite);

CREATE INDEX tb\_std\_tramites\_FKIndex2 ON tb\_std\_tramites (tb\_std\_areas\_idtb\_std\_area);

CREATE INDEX tb\_std\_tramites\_FKIndex3 ON tb\_std\_tramites (tb\_std\_alumnos\_idtb\_std\_alumno);

CREATE INDEX tb\_std\_tramites\_FKIndex4 ON tb\_std\_tramites (tb\_std\_usuarios\_idtb\_std\_usuario);

COMMENT ON TABLE tb\_std\_tramites IS 'Tabla que almacena los datos de los tramites de la Unidad de Postgrado.';

CREATE INDEX IFK\_Rel\_01 ON tb\_std\_tramites (tb\_std\_estado\_tramites\_idtb\_std\_estado\_tramite);

CREATE INDEX IFK\_Rel\_04 ON tb\_std\_tramites (tb\_std\_areas\_idtb\_std\_area);

CREATE INDEX IFK\_Rel\_05 ON tb\_std\_tramites (tb\_std\_alumnos\_idtb\_std\_alumno);

CREATE INDEX IFK\_Rel\_06 ON tb\_std\_tramites (tb\_std\_usuarios\_idtb\_std\_usuario);

-- ------------------------------------------------------------

-- Tabla que almacena los datos de los expedientes por los tramites realizados en la Unidad de Postgrado.

-- ------------------------------------------------------------

CREATE TABLE tb\_std\_expedientes (

idtb\_std\_expediente INTEGER NOT NULL ,

tb\_std\_tramites\_idtb\_std\_tramite INTEGER NOT NULL ,

PRIMARY KEY(idtb\_std\_expediente) ,

FOREIGN KEY(tb\_std\_tramites\_idtb\_std\_tramite)

REFERENCES tb\_std\_tramites(idtb\_std\_tramite));

CREATE INDEX tb\_std\_expedientes\_FKIndex1 ON tb\_std\_expedientes (tb\_std\_tramites\_idtb\_std\_tramite);

COMMENT ON TABLE tb\_std\_expedientes IS 'Tabla que almacena los datos de los expedientes por los tramites realizados en la Unidad de Postgrado.';

CREATE INDEX IFK\_Rel\_03 ON tb\_std\_expedientes (tb\_std\_tramites\_idtb\_std\_tramite);

CREATE TABLE tb\_std\_requisitos\_has\_tb\_std\_tramites (

tb\_std\_requisitos\_idtb\_std\_requisito INTEGER NOT NULL ,

tb\_std\_tramites\_idtb\_std\_tramite INTEGER NOT NULL ,

PRIMARY KEY(tb\_std\_requisitos\_idtb\_std\_requisito, tb\_std\_tramites\_idtb\_std\_tramite) ,

FOREIGN KEY(tb\_std\_requisitos\_idtb\_std\_requisito)

REFERENCES tb\_std\_requisitos(idtb\_std\_requisito),

FOREIGN KEY(tb\_std\_tramites\_idtb\_std\_tramite)

REFERENCES tb\_std\_tramites(idtb\_std\_tramite));

CREATE INDEX tb\_std\_requisitos\_has\_tb\_std\_tramites\_FKIndex1 ON tb\_std\_requisitos\_has\_tb\_std\_tramites (tb\_std\_requisitos\_idtb\_std\_requisito);

CREATE INDEX tb\_std\_requisitos\_has\_tb\_std\_tramites\_FKIndex2 ON tb\_std\_requisitos\_has\_tb\_std\_tramites (tb\_std\_tramites\_idtb\_std\_tramite);

CREATE INDEX IFK\_Rel\_07 ON tb\_std\_requisitos\_has\_tb\_std\_tramites (tb\_std\_requisitos\_idtb\_std\_requisito);

CREATE INDEX IFK\_Rel\_08 ON tb\_std\_requisitos\_has\_tb\_std\_tramites (tb\_std\_tramites\_idtb\_std\_tramite);