**Organismo de Investigación Judicial**

***Gestor de Reuniones***

**Manual Técnico**

**Versión final**

**11, 2020**

# Tabla de contenido

[**Tabla de contenido**](#_heading=h.z86h4w2zb864) **2**

[**Introducción**](#_heading=h.rku1hprj07in) **3**

[**Arquitectura del Software**](#_heading=h.gp9diwz6s26t) **4**

[**Requerimientos técnicos del sistema**](#_heading=h.3whwml4) **5**

[Requerimientos de Software](#_heading=h.bxqi54qt1ssp) 5

[Requerimientos de Hardware](#_heading=h.7j8lzjpf05kb) 6

[**Librerías o componentes de Terceros**](#_heading=h.u7zylw76v357) **6**

[**Manejo de Controles**](#_heading=h.3whwml4) **8**

[**Estándar de Programación**](#_heading=h.3whwml4) **10**

[**Errores y mensajes Comunes**](#_heading=h.3whwml4) **11**

[**Parámetros Generales**](#_heading=h.3whwml4) **12**

[**Descripción técnica de la aplicación**](#_heading=h.3whwml4) **14**

[Componentes de Interfaz](#_heading=h.qrox6qycsown) 14

[Componentes de la Lógica de Negocios](#_heading=h.ira279xxsjrt) 17

[Componentes de Acceso a Datos](#_heading=h.rn3lycjzo224) 27

[**Base de Datos**](#_heading=h.3whwml4) **40**

[**Aspectos de Seguridad**](#_heading=h.3whwml4) **53**

[**Instalación en los diferentes ambientes**](#_heading=h.3whwml4) **54**

[12.1 Requisitos de hardware y software](#_heading=h.9ofiqjqbl0jb) 54

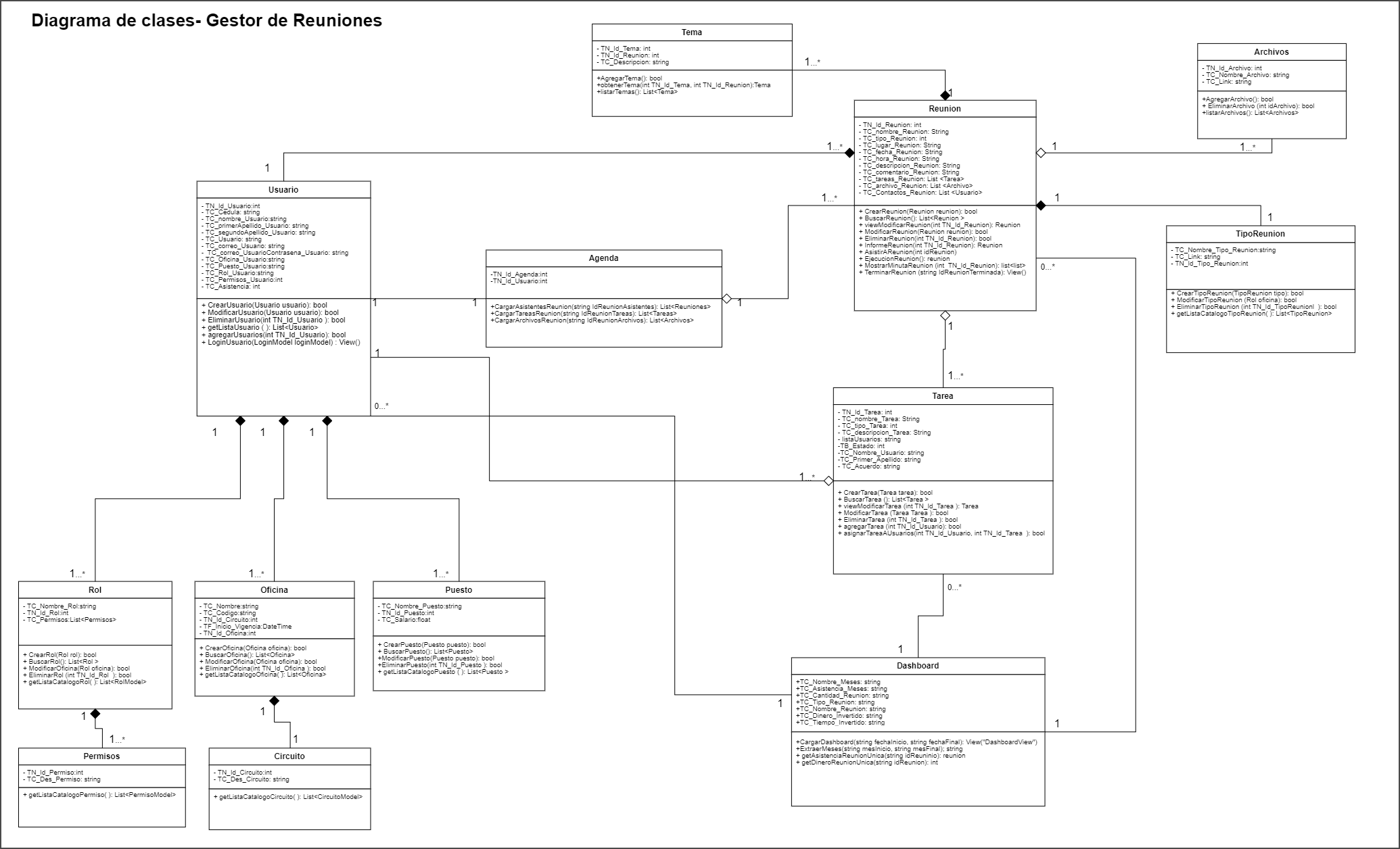
[12.2 Compilación](#_heading=h.20e4hudsdeq1) 55

[12.3 Instalación](#_heading=h.1zqj6zrn3x9p) 55

# Introducción

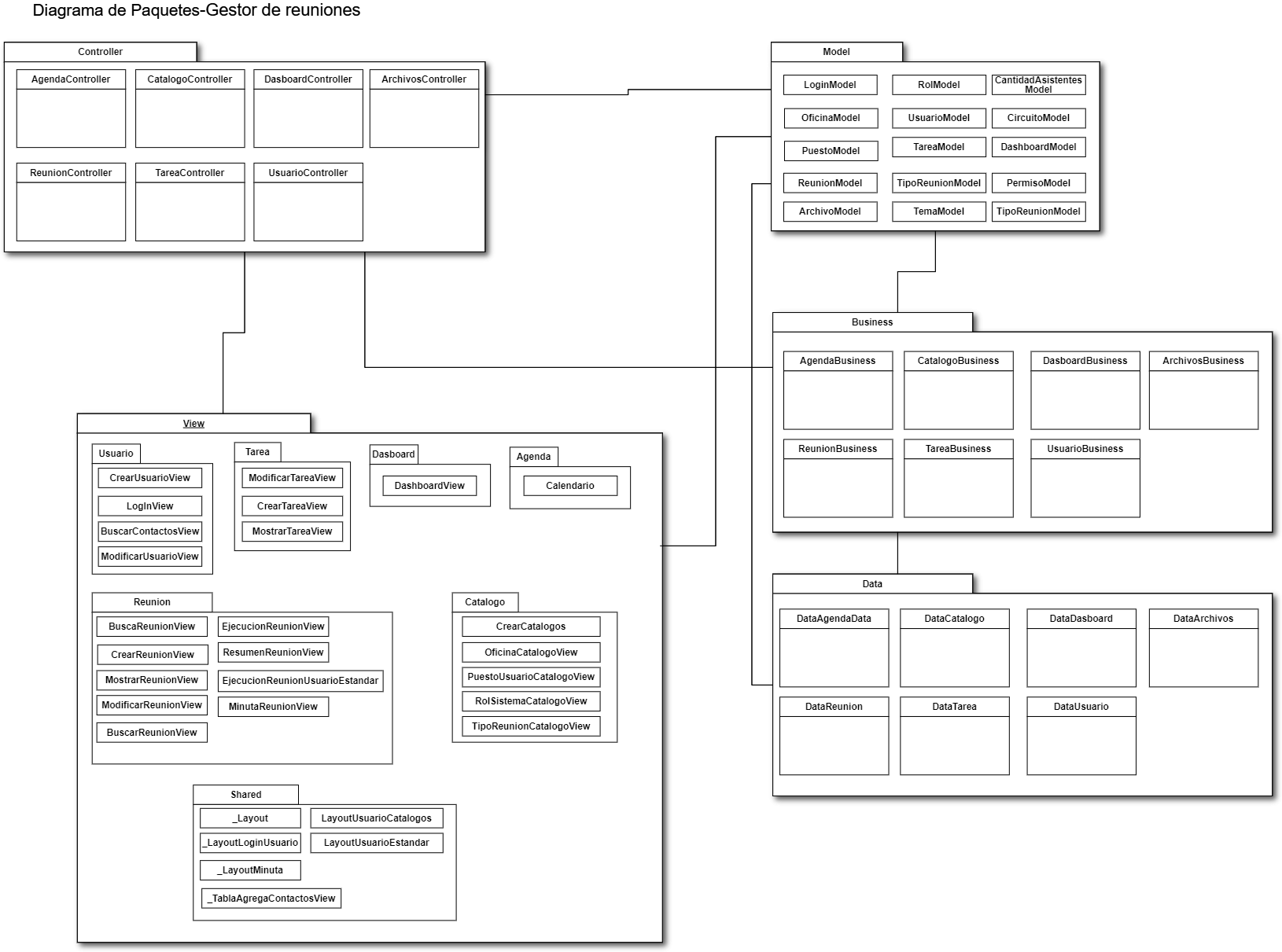
El sistema que se le entregará al OIJ fue realizado en Visual Studio 2019, como el motor de Bases de Datos de SQL, el presente documento se realiza con el fin de poder gestionar la forma en la que el OIJ organiza sus reuniones actualmente, para lograr una mejora en la organización de las mismas, Los principales usuarios que realizaran uso de este sistema serán el administrador del sistema, los administradores de catálogos y los usuarios estándar que van a tener el uso más básico del sistema. Se realizó el uso de bootstrap, además de algunas librerías como lo son la rotativa y el uso de Chart para las realización de los gráficos, que brindan la información de la reunión, y uso de diferentes funciones de JavaScript para el control de la interfaz como lo son los datables, a su vez el uso de Modals. Una de las áreas funcionales del Poder Judicial son los departamentos de TI.

# Arquitectura del Software



Para el desarrollo de la aplicación se utilizó la arquitectura “N Capas” la cual permite que el sistema posea la capacidad de recibir mantenimiento y soporte de manera más sencilla, además de que esta arquitectura nos permite una mayor flexibilidad para poder agregar nuevas funcionalidades al sistema o quitar algunas sin afectar a las demás. Otro punto importante que nos otorga la utilización de esta arquitectura es el aumento en la robustez de la aplicación lo cual brinda mayor seguridad al sistema debido a la segmentación del código.

Con respecto a la arquitectura de N-Capas, a continuación se muestra el modelo que describe los módulos que contiene el sistema.



# Requerimientos técnicos del sistema

## Requerimientos de Software

En caso de mantenimiento o escalabilidad del sistema se deben tomar en cuenta las herramienta que se utilizaron para el desarrollo de este proyecto. En este caso se trabajó con la arquitectura MVC en .net core 3.1 , para lo cual se requiere Microsoft Visual Studio, específicamente Microsoft Visual Studio Community 2019 en su versión 16.6.2.Además se requiere Asynchronous JavaScript (Ajax),

Por otra parte, el gestor de bases de datos implementado es SQL Server V18.5, por lo tanto se requiere SQL Server Management Studio Versión 15.0.18330.0.

## Requerimientos de Hardware

Para el desarrollo, mantenimiento de este sistema solo se requiere una computadora que cumpla los siguientes requisitos:

* 8GB de memoria.
* 1,0 GHz de procesamiento (mínimo).
* 500 MB de almacenamiento.
* Sistema Operativo Windows 10.
* Conexión a red. (Intranet)

Para la implementación del sistema aproximadamente se requiere un servidor con las siguientes características:

* RAM 24gb
* SO: windows server r2
* CPU 3.47 ghz

# Librerías o componentes de Terceros

A continuación se describen las librerías que se implementaron en el proyecto :

* **Rotativa.io**: es una aplicación en la nube que permite crear PDF Se basa en tecnologías de nube potentes y fiables. La solicitud de servidor para crear un PDF será procesada por una infraestructura distribuida y el archivo resultante se entregará directamente a su usuario de una manera segura y privada. Está librería requiere conexión a internet. La forma de la implementación hace una una vista en html + css + bootstrap pueda ser visualizada en un pdf el cual se puede guardar. Se incorporó en este proyecto por medio del administrador de paquetes NuGet. Se utiliza la versión Rotativa.AspNetCore 1.2.0-beta. Su implementación también requiere dos archivos .exe que se encuentran en la carpeta “Rotativa” en la raíz del proyecto.
* **System.Data.SqlClient.**  Crea una instancia de SqlConnection la cual crea un SqlConnection que abre y se establece como la conexión con el gestor de bases de datos, en este caso SQL server. Está conexión es el canal donde viajan los datos y transacciones directamente en la base de datos. Su implementación se realiza por medio del administrador de paquetes NuGet. Se debe agregar la siguiente instrucción en la clase appsettings.json, en la cual van las credenciales de la BD:

"ConnectionStrings": { "DB\_Connection": "Server=163.178.107.10; Database=Gestor\_Reuniones2; User Id=laboratorios; Password=KmZpo.2796; Trusted\_Connection=False; MultipleActiveResultSets=true;" }

* **Bootstrap**: es una librería con un conjunto de herramientas de [código abierto](https://es.wikipedia.org/wiki/C%C3%B3digo_abierto) para diseño de sitios y aplicaciones web. De acá se utilizaron plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en [HTML](https://es.wikipedia.org/wiki/HTML) y [CSS](https://es.wikipedia.org/wiki/Hojas_de_estilo_en_cascada), así como extensiones de [JavaScript](https://es.wikipedia.org/wiki/JavaScript) adicionales. Se puede incluir por medio de su cdn, o como en este caso descargando los archivos directamente desde la página oficial. https://getbootstrap.com/
* **Chart**: es una biblioteca para la visualización de datos por medio de gráficos JS. En este proyecto se trabajó en el dashboard que muestra información importante sobre las reuniones, tiempos invertidos, asistentes. Se utilizaron 4 tipos diferentes de gráficos, pero la librería cuenta con 8 diferentes tipos, y sus respectivas variaciones. Se puede incluir por medio de su cdn, o como en este caso descargando los archivos directamente desde la página oficial. https://www.chartjs.org/ .
* **DataTables**: es una librería que permite mejorar la experiencia UX/UI de la tablas básica html, para esto recurre al CSS, bootstrap y claramente HTML. Está herramienta permite buscar por columnas, indexar y organizar de mayor a menor y viceversa al gusto del usuario.Su ducumentación se encuentra en <https://datatables.net/examples/styling/bootstrap4> En este caso se implementa por su CDN Versión DataTables 1.10

href="<https://cdn.datatables.net/1.10.22/css/jquery.dataTables.css>"

src="<https://cdn.datatables.net/1.10.22/js/jquery.dataTables.js>"

* **Fullcalendar**: es una librería que permite gestionar eventos, en este caso reuniones agendadas. De una manera agradable, mejora lo que sería una tabla html, y la transforma en un calendario similar al de Google. Su documentación y archivos se encuentran en: https://fullcalendar.io/

# Manejo de Controles

Para distintos módulos de la aplicación se hace uso de DataTables versión 1.10 de Bootstrap 4, el cual lo obtuvimos de la siguiente página web [https://datatables.net/examples/styling/bootstrap](https://datatables.net/examples/styling/bootstrap4). Por medio de esa documentación se obtuvo el javascript, HTML y las bibliotecas de CSS que nos permitieron darle formato a los DataTables. Estos controles permiten crear tablas que a la hora de que el usuario interactúen con ellas, se haga de una manera dinámica, donde tiene un campo de búsqueda, contiene una etiqueta <select> donde el usuario puede seleccionar la cantidad de tuplas que se le carguen en una vista, además según la cantidad de filas que contenga la tabla se crea una paginación en la misma. Se usó este tipo de tabla para listar reuniones, catálogos y tareas.

**Javascript**

$(document).ready(function() {

$('#example').DataTable();

} );

**CSS**

* <https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/4.5.2/css/bootstrap.css>
* <https://cdn.datatables.net/1.10.22/css/dataTables.bootstrap4.min.css>

Por otra parte, se implementó Fullcalendar, la cual es una librería que permite a los usuarios tener un control de las reuniones. El siguiente método permite añadir evento al calendario.

calendar.addEvent({

id: reunion.TN\_Id\_Reunion,

title: reunion.TC\_Nombre\_Reunion,

start: salida,

allDay: true,

eventResourceEditable: false,

droppable: false,

editable: false,

resourceEditable: false

});

# Estándar de Programación

En el desarrollo de este proyecto se utilizaron estándares de programación. El primero de ellos es el Estándar de la Contraloría general de la república y Normas técnicas para la gestión y el control de las Tecnologías de Información. Además del estándar interno de desarrollo de software del Poder Judicial. Como punto principal nuestro equipo se basó en el capítulo III Normas técnicas para la gestión y el control de las Tecnologías de Información que trata sobre la Implementación de tecnologías de información. En el cual podemos nombrar algunos de las pautas que se siguieron:

* Desarrollar y aplicar un marco metodológico que guíe los procesos de implementación y considere la definición de requerimientos, los estudios de factibilidad, la elaboración de diseños, la programación y pruebas, el desarrollo de la documentación, la conversión de datos y la puesta en producción, así como también la evaluación post implantación de la satisfacción de los requerimientos.
* Establecer los controles y asignar las funciones, responsabilidades y permisos de acceso al personal a cargo de las labores de implementación y mantenimiento de software.
* Controlar la implementación del software en el ambiente de producción y garantizar la integridad de datos y programas en los procesos de conversión y migración.
* Definir los criterios para determinar la procedencia de cambios y accesos de emergencia al software y datos, y los procedimientos de autorización, registro, supervisión y evaluación técnica, operativa y administrativa de los resultados de esos cambios y accesos.
* Controlar las distintas versiones de los programas que se generen como parte de su mantenimiento

# Errores y mensajes Comunes

El sistema está validado de forma que los usuario no puedan generar un colapso y que la página se caiga. Lo cual no quiere decir que no existan errores. Dentro de los mensaje más comunes que se pueden encontrar están:

* Error en las credenciales. Se muestra en la pantalla de login y se genera cuando el nickname del usuario y la contraseña no coinciden con credenciales guardados en la base de datos.
* La fecha es menor a la fecha actual. Cuando se crea una reunión pero la fecha es menos a la actual.
* La lista de Usuarios, Tareas, Temas o Archivos está vacía: cuando se crea una reunión, sin usuarios, ni temas, ni tareas.
* Error, los datos ya existen: Cuando el usuario crea una reunión, tarea, usuario, o un nuevo catálogo, y ya existe esos datos de entrada en la base de datos.
* Reunión no disponible: Cuando un usuario quiere ingresar a una reunión, pero ya está terminó, o es antes de la fecha de inicio.

Si bien estos errores no hacen que el sistema colapse, los siguientes si pueden botar la página.

* Fallo con el internet. Algunas librerías necesitan de internet, como es el caso de Rotativa, la cual por su sistema de procesamiento en la nube, si existiera algún fallo con el internet la página colapsaría.
* Fallos con la base de datos. En caso que la base de datos dure una cierta cantidad en responder, o esté desconectada la página va a mostrar un error con el siguiente: SqlException: No se puede abrir la base de datos "nombreBD" solicitada por el inicio de sesión. Error de inicio de sesión. Error de inicio de sesión del usuario ‘user’.

# Parámetros Generales

Los parámetros generales mostrarán los datos iniciales que debe tener para lograr el funcionamiento inicial del sistema. Dicha configuración se hará por parte del administrador y en este caso desde el motor de base de datos, debido a que se necesita un usuario inicial el cual agregara nuevos usuarios y demás datos a la aplicación para sacar todo el provecho de la misma .

Lo cual se debe ejecutar el “Scrip\_Datos\_Iniciales\_del\_Sistema” el cual será proveido por los creadores listo para ejecutarse en la base de datos la cual será consumida por el sistema. A continuación se muestra detalladamente cada apartado dentro de este script para tener claro lo que se hará.

**Parámetros obligatorios para el funcionamiento inicial de la aplicación**

* Se debe crear el primer circuito al que pertenece la primera oficina del sistema a la cual va a pertenecer el primer empleado que se encargará de crear los catálogos.

insert into T\_Circuito(TC\_Des\_Circuito) values('01');

* Se debe crear una oficina, los datos que se ingresan son el nombre, código, id del circuito, fecha de inicio de vigencia.

exec Sp\_Insert\_Oficina 'Planes y Operaciones','PO',1,'2020-11-13'

* Estos son los permisos que se asignan a los diferentes roles en el sistema, estos permisos están quemados en el sistema por lo cual si se cambian se debe de cambiar en el sistema.

insert into T\_Permiso(TC\_Nombre\_Permiso) values ('Acceso total');

insert into T\_Permiso(TC\_Nombre\_Permiso) values ('Acceso estandar');

insert into T\_Permiso(TC\_Nombre\_Permiso) values ('Administardor catalogos');

* Se debe crear un rol, en este caso se recomienda la creación de un usuario 'Administardor catálogos para que este pueda ingresar todos los datos que ocupa el sistema para funcionar.

exec Sp\_Insert\_Rol 'Jefe',1

exec Sp\_Insert\_Rol 'Usuarios estandar',2

select\*from T\_Rol

* Se debe crear un puesto al cual va a pertenecer el primer usuario del sistema, para el cual se especifica el nombre y el precio del salario por hora.

exec Sp\_Insert\_Puesto 'Ing. Sistemas',3500

exec Sp\_Insert\_Puesto 'Administrados DB',3500

select\*from T\_Puesto

* Se debe crear un Usuario, para esto se especifica los datos en el siguiente orden: usuario, contraseña, identificación, nombre, primer apellido, segundo apellido, correo, id del puesto, id de la oficina, id del rol.

exec Sp\_Insert\_Usuario 'Admin','admin123','301200164','Juan','Perez','Abarca','juan.perez@gmail.com',1,1,1

exec Sp\_Insert\_Usuario 'Usuario','usuario123','302450210','Maria','Romero','Sanchez','maria.romero@gmail.com',2,1,2

select\*from T\_Usuario

# Descripción técnica de la aplicación

El presente sistema gestor de reuniones está elaborado en un único proyecto, siguiendo la arquitectura MVC. Si bien el sistema no tiene módulos por aparte (otros proyectos), sí cuenta con 4 capas la cuales vamos a describir a continuación:

## Componentes de Interfaz

* Login
  + Nombre físico: LoginView.cshtml
  + Descripción de la funcionalidad: Da acceso a que los usuarios puedan ingresar al sistema. Muestras dos campos de texto y un botón.
  + Relación con la Lógica de Negocio: UsuarioController
* Nueva reunión
  + Nombre físico: CrearReunionView.cshtml
  + Ruta en el menú de la pantalla: Reunión/Nueva
  + Descripción de la funcionalidad: Permite crear nuevas reuniones, para esto se tienen 6 campos de texto, tres tablas presentadas por medio de modals. El usuario ingresa los datos de la reunión, y selecciona los usuarios, las tareas, y los temas.
  + Relación con la Lógica de Negocio: ReunionController
* Modificar reunión
  + Nombre físico: ModificarReunionView.cshtml
  + Ruta en el menú de la pantalla: Reunion/Modificar
  + Descripción de la funcionalidad: La interfaz muestra cargados los campos de texto con los datos de la reunión, el usuario cambia los datos que desea.
  + Relación con la Lógica de Negocio: ReunionController
* Historial reunión
  + Nombre físico: ResumenReunionView.cshtml
  + Ruta en el menú de la pantalla: Reunion/Historial
  + Descripción de la funcionalidad: Muestra una tabla con las reuniones finalizadas. Cada tupla de la tabla tiene un boton la cual dirige a la pantalla de resumen de reunión. En esta última se muestra los detalles de la reunión seleccionada. Nombre, tipo, comentario, descripción, contactos, tareas,temas. Además de un par de gráficos con la participación y las cifras de tiempo, y dinero gastado.
  + Relación con la Lógica de Negocio:ReunionController
* Tarea nueva
  + Nombre físico: CrearTareaView.cshtml
  + Ruta en el menú de la pantalla: Tareas/Nueva
  + Descripción de la funcionalidad: Permite crear nuevas tareas, para esto se tienen nombre de la tarea, descripción y usuarios encargados.
  + Relación con la Lógica de Negocio: TareaController
* Modificar tarea
  + Nombre físico: ModificarTareaView.cshtml
  + Ruta en el menú de la pantalla: Tarea/Modificar
  + Descripción de la funcionalidad: La interfaz muestra cargados los campos de texto con los datos de la tarea, el usuario cambia los datos que desea.
  + Relación con la Lógica de Negocio: TareaController
* Nuevo Usuario
  + Nombre físico: CrearUsuario.cshtml
  + Ruta en el menú de la pantalla: Usuario/Nuevo
  + Descripción de la funcionalidad: Permite crear nuevos usuarios, para esto se tienen que ingresar nombre, primer y segundo apellido, cédula, usuario, correo, contraseña, oficina, puesto, rol.
  + Relación con la Lógica de Negocio: UsuarioController
* Modificar Usuario
  + Nombre físico: ModificarUsuario.cshtml
  + Ruta en el menú de la pantalla: Usuario/Modificar
  + Descripción de la funcionalidad: La interfaz muestra cargados los campos de texto con los datos del usuario, el cliente cambia los datos que desea.
  + Relación con la Lógica de Negocio: UsuarioController
* Tipo Reunión
  + Nombre físico:TipoReunionCatalogoView.cshtml
  + Ruta en el menú de la pantalla: Catálogo/Tipo reunión
  + Descripción de la funcionalidad: Permite crear nuevos tipos de reunión, cuenta con un único campo de texto, destinado para el nombre.Además se puede hacer el CRUD completo, se presenta por pestañas desplazables
  + Relación con la Lógica de Negocio: CatalogoController
* Oficina
  + Nombre físico:OficinaCatalogoView.cshtml
  + Ruta en el menú de la pantalla: Catálogo/Oficina
  + Descripción de la funcionalidad: Permite nuevas oficinas, cuenta con campos de texto, destinado para el nombre, código, circuito, y fecha .Además se puede hacer el CRUD completo, se presenta por pestañas desplazables
  + Relación con la Lógica de Negocio: CatalogoController
* Rol sistema
  + Nombre físico: RolSistemaCatalogoView.cshtml
  + Ruta en el menú de la pantalla: Catálogo/Rol sistema
  + Descripción de la funcionalidad: Permite nuevos roles del sistema, cuenta con campos de texto, destinado para el nombre, acceso al sistema. Además se puede hacer el CRUD completo, se presenta por pestañas desplazables
  + Relación con la Lógica de Negocio: CatalogoController
* Puesto usuario
  + Nombre físico: PuestoUsuarioCatalogoView.cshtml
  + Ruta en el menú de la pantalla: Catalogo/Puesto del usuario
  + Descripción de la funcionalidad: Permite nuevos puestos de usuario, cuenta con campos de texto, destinado para el nombre y pago por horas. Además se puede hacer el CRUD completo, se presenta por pestañas desplazables
  + Relación con la Lógica de Negocio: CatalogoController
* Agenda Personal
  + Nombre físico: Calendario.cshtml
  + Ruta en el menú de la pantalla: Agenda Personal
  + Descripción de la funcionalidad: Muestra un calendario con las reuniones agendadas para dicha persona. Cada reunión se puede clickear y ver información completa de dicha reunión.
  + Relación con la Lógica de Negocio: AgendaController
* Dashboard
  + Nombre físico: DashboardView.cshtml
  + Ruta en el menú de la pantalla: Dashboard
  + Descripción de la funcionalidad: Presenta los datos de las reuniones finalizadas por medios gráficos. Se requiere que el usuario ingrese dos rangos de fechas.
  + Relación con la Lógica de Negocio: DashboardController

## Componentes de la Lógica de Negocios

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre físico | ReunionController.cs | Nombre | ReunionController |
| Descripción | En este controlador se gestiona toda la lógica que está relacionada con las reuniones. Solicita los datos a la capa de reglas de negocio, y estas a su vez a la capa de datos. En resumen este controlador se encarga de empaquetar los datos en listas (ViewBag) y enviarlos a la vista. | | |
| Nombre Servicios (Métodos ) | | viewCrearReunion | |
| Descripción | Datos entrada | Datos de salida | Data |
| Redirecciona a la vista CrearReunionView. Es un método de salida de datos. |  | List<TipoReunionModel>  List<UsuarioModel> listaUsuarios  List<TareaModel> listaTareas | DataReunión |
| Nombre Servicios (Métodos ) | | CrearReunion | |
| Descripción | Datos entrada | Datos de salida | Data |
| Redirecciona a la vista CrearReunionView. Es un método de salida de datos. | ReunionModel |  | DataReunión |
| Nombre Servicios (Métodos ) | | EliminarReunion | |
| Descripción | Datos entrada | Datos de salida | Data |
| Salida de datos.Redirecciona a BuscaReunionView | string idReunion | List<ReunionModel> | DataReunión |
| Nombre Servicios (Métodos ) | | ModificarReunion | |
| Descripción | Datos entrada | Datos de salida | Data |
| Salida de datos.Devuelve una respuesta de validación | ReunionModel | List<ReunionModel> | DataReunión |
| Nombre Servicios (Métodos ) | | viewBuscaReunion | |
| Descripción | Datos entrada | Datos de salida | Data |
| Redirección de vista |  | List<ReunionModel> | DataReunión |
| Nombre Servicios (Métodos ) | | viewMostrarReunion | |
| Descripción | Datos entrada | Datos de salida | Data |
| Redirección de vista |  | List<ReunionModel> | DataReunión |
| Nombre Servicios (Métodos ) | | viewResumenReunion | |
| Descripción | Datos entrada | Datos de salida | Data |
| Redirección de vista | string idReunion | List<UsuarioModel>  List<TareaModel> ;  List<ArchivoModel> | DataReunión |
| Nombre Servicios (Métodos ) | | ViewModificarReunion | |
| Descripción | Datos entrada | Datos de salida | Data |
| Redirección de vista | string idReunion | List<TareaModel> List<UsuarioModel>  List<ArchivoModel>  List<TipoReunionModel> | DataReunión |
| Nombre Servicios (Métodos ) | | viewEjecucionReunion | |
| Descripción | Datos entrada | Datos de salida | Data |
| Redirección de vista | string IdReunionAsistir |  | DataReunión |
| Nombre Servicios (Métodos ) | | ViewReunionEnEjecucion | |
| Descripción | Datos entrada | Datos de salida | Data |
| Redirección de vista |  | List<TareaModel> List<UsuarioModel>  List<ArchivoModel>  ReunionModel  List<TemasModel> | DataReunión |
| Nombre Servicios (Métodos ) | | mostrarMinutaReunion | |
| Descripción | Datos entrada | Datos de salida | Data |
| Redirección de vista | string idReunion | List<TareaModel> List<UsuarioModel>  List<ReunionModel> | DataReunión |
| Nombre Servicios (Métodos ) | | AgregarAcuerdosTemas | |
| Descripción | Datos entrada | Datos de salida | Data |
| Salida de datos.Devuelve una respuesta de validación | string idTemas, string acuerdoTema |  | DataReunión |
| Nombre Servicios (Métodos ) | | AgregarAcuerdosTareas | |
| Descripción | Datos entrada | Datos de salida | Data |
| Salida de datos.Devuelve una respuesta de validación | string idTemas, string acuerdoTarea |  | DataReunión |
| Nombre Servicios (Métodos ) | | TerminarReunion | |
| Descripción | Datos entrada | Datos de salida | Data |
| Salida de datos.Devuelve una respuesta de validación | string IdReunionTerminada | List<TareaModel> List<UsuarioModel>  List<ArchivoModel>  ReunionModel  List<TemasModel> | DataReunión |

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre físico | TareaController.cs | Nombre | TareaController |
| Descripción | En este controlador se gestiona toda la lógica que está relacionada con las tareas. Solicita los datos a la capa de reglas de negocio, y estas a su vez a la capa de datos. En resumen este controlador se encarga de empaquetar los datos en listas (ViewBag) y enviarlos a la vista. | | |
| Nombre Servicios (Métodos ) | | viewCrearTarea | |
| Descripción | Datos entrada | Datos de salida | Data |
| Redirección de vista |  | List<UsuarioModel> | DataTarea |
| Nombre Servicios (Métodos ) | | viewBuscarTarea | |
| Descripción | Datos entrada | Datos de salida | Data |
| Redirección de vista |  | List<TareaModel> | DataTarea |
| Nombre Servicios (Métodos ) | | viewModificarTarea | |
| Descripción | Datos entrada | Datos de salida | Data |
| Redirección de vista | string idTarea | List<UsuarioModel> TareaModel | DataTarea |
| Nombre Servicios (Métodos ) | | EliminarTarea | |
| Descripción | Datos entrada | Datos de salida | Data |
| Método transaccional. Redirección de vista | string idTarea | List<TareaModel> listaTareas | DataTarea |
| Nombre Servicios (Métodos ) | | CrearTarea | |
| Descripción | Datos entrada | Datos de salida | Data |
| Método transaccional. Devuelve una respuesta de validación | TareaModel |  | DataTarea |
| Nombre Servicios (Métodos ) | | ModificarTarea | |
| Descripción | Datos entrada | Datos de salida | Data |
| Método transaccional. Devuelve una respuesta de validación | TareaModel |  | DataTarea |

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre físico | UsuarioController.cs | Nombre | UsuarioController |
| Descripción | En este controlador se gestiona toda la lógica que está relacionada con los usuarios. Solicita los datos a la capa de reglas de negocio, y estas a su vez a la capa de datos. En resumen este controlador se encarga de empaquetar los datos en listas (ViewBag) y enviarlos a la vista. Tambien realiza operaciones transaccionales, como inserciones, modificaciones, consultas y eliminaciones, para los cuales solo da un mensaje de resultado. | | |
| Nombre Servicios (Métodos ) | | MuestraCrearUsuario | |
| Descripción | Datos entrada | Datos de salida | Data |
| Método transaccional. Devuelve una respuesta de validación |  | List<PuestoModel>  List<OficinaModel>  List<RolModel> | UsuarioData |
| Nombre Servicios (Métodos ) | | MuestraLoginUsuario | |
| Descripción | Datos entrada | Datos de salida | Data |
| Método transaccional. Redirección de vista |  | List<ReunionModel> | UsuarioData |
| Nombre Servicios (Métodos ) | | CrearUsuario | |
| Descripción | Datos entrada | Datos de salida | Data |
| Método transaccional. Redirección de vista | UsuarioModel | List<PuestoModel>  List<OficinaModel>  List<RolModel> | UsuarioData |
| Nombre Servicios (Métodos ) | | EliminarUsuario | |
| Descripción | Datos entrada | Datos de salida | Data |
| Método transaccional. Redirección de vista | UsuarioModel | List<UsuarioModel> | UsuarioData |
| Nombre Servicios (Métodos ) | | viewListarUsuario | |
| Descripción | Datos entrada | Datos de salida | Data |
| Redirección de vista |  | List<UsuarioModel> | UsuarioData |
| Nombre Servicios (Métodos ) | | VistaModificarUsuario | |
| Descripción | Datos entrada | Datos de salida | Data |
| Redirección de vista | UsuarioModel | List<PuestoModel>  List<OficinaModel>  List<RolModel> | UsuarioData |
| Nombre Servicios (Métodos ) | | ModificarUsuario | |
| Descripción | Datos entrada | Datos de salida | Data |
| Método transaccional. Redirección de vista | UsuarioModel | List<PuestoModel>  List<OficinaModel>  List<RolModel> | UsuarioData |
| Nombre Servicios (Métodos ) | | IniciarLogin | |
| Descripción | Datos entrada | Datos de salida | Data |
| Redirección de vista |  |  | UsuarioData |

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre físico | CatalogoController.cs | Nombre | CatalogoController |
| Descripción | En este controlador se gestiona toda la lógica que está relacionada con los Catálogos, los cuales son cuatro: tipo reunión, rol, puesto, oficina. Solicita los datos a la capa de reglas de negocio, y estas a su vez a la capa de datos. En resumen este controlador se encarga de empaquetar los datos en listas (ViewBag) y enviarlos a la vista. También realiza operaciones transaccionales, como inserciones, modificaciones, consultas y eliminaciones, para los cuales da un mensaje de resultado. | | |
| Nombre Servicios (Métodos ) | | VistaCrearCatalogos | |
| Descripción | Datos entrada | Datos de salida | Data |
| Redirección de vista |  |  |  |
| Nombre Servicios (Métodos ) | | viewTipoReunionCatalogo | |
| Descripción | Datos entrada | Datos de salida | Data |
| Redirección de vista |  | List<TipoReunionModel> | DataCatalogo |
| Nombre Servicios (Métodos ) | | CrearTipoReunion | |
| Descripción | Datos entrada | Datos de salida | Data |
| Método transaccional. Devuelve una respuesta de validación | TipoReunionModel |  | DataCatalogo |
| Nombre Servicios (Métodos ) | | ModificarTipoReunion | |
| Descripción | Datos entrada | Datos de salida | Data |
| Método transaccional. Devuelve una respuesta de validación | TipoReunionModel |  | DataCatalogo |
| Nombre Servicios (Métodos ) | | EliminarTipoReunion | |
| Descripción | Datos entrada | Datos de salida | Data |
| Método transaccional. Devuelve una respuesta de validación | TipoReunionModel |  | DataCatalogo |
| Nombre Servicios (Métodos ) | | viewOfinaCatalogo | |
| Descripción | Datos entrada | Datos de salida | Data |
| Redirección de vista |  | List<OficinaModel>  List<CircuitoModel> | DataCatalogo |
| Nombre Servicios (Métodos ) | | CrearOficina | |
| Descripción | Datos entrada | Datos de salida | Data |
| Método transaccional. Devuelve una respuesta de validación | OficinaModel |  | DataCatalogo |
| Nombre Servicios (Métodos ) | | ModificarOficina | |
| Descripción | Datos entrada | Datos de salida | Data |
| Método transaccional. Devuelve una respuesta de validación | OficinaModel |  | DataCatalogo |
| Nombre Servicios (Métodos ) | | EliminarOficina | |
| Descripción | Datos entrada | Datos de salida | Data |
| Método transaccional. Devuelve una respuesta de validación | OficinaModel |  | DataCatalogo |
| Nombre Servicios (Métodos ) | | viewRolSistemaCatalogo | |
| Descripción | Datos entrada | Datos de salida | Data |
| Redirección de vista |  | List<RolModel>  List<PermisosModel> | DataCatalogo |
| Nombre Servicios (Métodos ) | | CrearRol | |
| Descripción | Datos entrada | Datos de salida | Data |
| Método transaccional. Devuelve una respuesta de validación | RolModel |  | DataCatalogo |
| Nombre Servicios (Métodos ) | | ModificarRol | |
| Descripción | Datos entrada | Datos de salida | Data |
| Método transaccional. Devuelve una respuesta de validación | RolModel |  | DataCatalogo |
| Nombre Servicios (Métodos ) | | EliminarRol | |
| Descripción | Datos entrada | Datos de salida | Data |
| Método transaccional. Devuelve una respuesta de validación | RolModel |  | DataCatalogo |
| Nombre Servicios (Métodos ) | | viewPuestoUsuarioCatalogo | |
| Descripción | Datos entrada | Datos de salida | Data |
| Redirección de vista |  | List<PuestoModel> | DataCatalogo |
| Nombre Servicios (Métodos ) | | CrearPuesto | |
| Descripción | Datos entrada | Datos de salida | Data |
| Método transaccional. Devuelve una respuesta de validación | PuestoModel |  | DataCatalogo |
| Nombre Servicios (Métodos ) | | ModificarPuesto | |
| Descripción | Datos entrada | Datos de salida | Data |
| Método transaccional. Devuelve una respuesta de validación | PuestoModel |  | DataCatalogo |
| Nombre Servicios (Métodos ) | | EliminarPuesto | |
| Descripción | Datos entrada | Datos de salida | Data |
| Método transaccional. Devuelve una respuesta de validación | PuestoModel |  | DataCatalogo |

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre físico | AgendaController.cs | Nombre | AgendaController |
| Descripción | En este controlador se gestiona toda la lógica que está relacionada con la agenda. Solicita los datos a la capa de reglas de negocio, y estas a su vez a la capa de datos. En resumen este controlador se encarga de empaquetar los datos en listas (ViewBag) y enviarlos a la vista. También realiza operaciones transaccionales, como inserciones, modificaciones, consultas y eliminaciones, para los cuales da un mensaje de resultado | | |
| Nombre Servicios (Métodos ) | | ViewAgendaPersonal | |
| Descripción | Datos entrada | Datos de salida | Data |
| Redirección de vista |  |  |  |
| Nombre Servicios (Métodos ) | | CargarAsistentesReunion | |
| Descripción | Datos entrada | Datos de salida | Data |
| Cargar datos en la vista | string IdReunionAsistentes | List<UsuarioModel>  List<ReunionModel> | DataAgenda |
| Nombre Servicios (Métodos ) | | CargarTareasReunion | |
| Descripción | Datos entrada | Datos de salida | Data |
| Cargar datos en la vista | string IdReunionTareas | List<TareaModel>  List<ReunionModel> | DataAgenda |
| Nombre Servicios (Métodos ) | | CargarArchivosReunion | |
| Descripción | Datos entrada | Datos de salida | Data |
| Cargar datos en la vista | string IdReunionArchivos | List<ArchivosModel>  List<ReunionModel> | DataAgenda |

## Componentes de Acceso a Datos

* Data Reunión
  + Descripción: Esta clase extrae datos directamente desde la base de datos. Hace transacciones relacionadas con las reuniones.
  + Métodos
    - Crear Reunion
      * Descripción. Inserta una reunión.
      * Datos entrada: reunionModel, idUsuario
      * Datos Salida: int[ ] valido
      * Procedimiento almacenado: Sp\_Insert\_Reunion
    - ModificarReunion
      * Descripción. Modifica una reunión.
      * Datos entrada: reunionModel
      * Datos Salida: bool
      * Procedimiento almacenado: Sp\_Update\_Reunion
    - ListarTemasReunion
      * Descripción. Lista los temas de una reunión
      * Datos entrada: string IdReunionTerminada
      * Datos Salida: bool
      * Procedimiento almacenado: Sp\_Finalizar\_Reunion
    - validarFechaReunion
      * Descripción. Valida que una fecha de una reunión este en el rango de fecha validas
      * Datos entrada: string IdReunion
      * Datos Salida: List<TemasModel>
      * Procedimiento almacenado: Sp\_Listar\_Reunion\_Temas
    - EliminarReunion
      * Descripción. Elimina una reunión
      * Datos entrada: string idReunion
      * Datos Salida: List<ReunionModel>
      * Procedimiento almacenado: Sp\_Delete\_Reunion
    - getListarReunion
      * Descripción. Extrae una lista de reuniones para presentarla en interfaz
      * Datos entrada:
      * Datos Salida: List<ReunionModel>
      * Procedimiento almacenado: Sp\_Listar\_Reunion
    - getListarReunionFinalizadas
      * Descripción. Lista las reuniones finalizadas
      * Datos entrada: string idReunion
      * Datos Salida: List<ReunionModel>
      * Procedimiento almacenado: Sp\_Delete\_Reunion
    - getReunionModificar
      * Descripción. Lista reuniones aptas para modificar
      * Datos entrada:
      * Datos Salida: List<ReunionModel>
      * Procedimiento almacenado: Sp\_Listar\_Reunion\_Finalizada
    - AgregarAcuerdosTareas
      * Descripción. agregar acuerdos a una reunión
      * Datos entrada: string idTarea, string acuerdoTarea
      * Datos Salida: bool
      * Procedimiento almacenado: Sp\_Insert\_Tarea\_Acuerdo
    - AgregarAcuerdosTemas
      * Descripción. agregar temas a una reunión
      * Datos entrada: string idTemas, string acuerdoTema
      * Datos Salida: bool
      * Procedimiento almacenado: Sp\_Insert\_Temas\_Acuerdo
    - UsuarioCreadorReunion
      * Descripción. Asigna los usuarios de una reunión
      * Datos entrada: string IdReunionAsistir
      * Datos Salida: string
      * Procedimiento almacenado: Sp\_Validar\_Host
    - getReunionMinuta
      * Descripción. Extrae las reuniones
      * Datos entrada: string idReunion
      * Datos Salida: ReunionModel
      * Procedimiento almacenado: Sp\_Validar\_Host
    - getTemasMinuta
      * Descripción. Extrae los temas de la minuta
      * Datos entrada: string idReunion
      * Datos Salida: List<ReunionModel>
      * Procedimiento almacenado: Sp\_Listar\_Temas\_Minuta
* Data Usuario
  + Descripción: Esta clase extrae datos directamente desde la base de datos. Hace transacciones relacionadas con los usuarios.
  + Métodos
    - Login Usuario
      * Descripción. Usuario inicia sesión en el sistema
      * Datos entrada: loginModel
      * Datos Salida: usuarioModel
      * Procedimiento almacenado: Sp\_Login\_Usuario
    - Crear Usuario
      * Descripción. Inserta un usuario.
      * Datos entrada: usuarioModel
      * Datos Salida: bool
      * Procedimiento almacenado: Sp\_Insert\_Usuario
    - Eliminar Usuario
      * Descripción. Elimina de manera lógica a un usuario.
      * Datos entrada: usuarioModel
      * Datos Salida: getListarUsuario()
      * Procedimiento almacenado: Sp\_Delete\_Usuario
    - get Listar Usuario
      * Descripción: Lista a todos los usuarios.
      * Datos entrada:
      * Datos Salida: List<UsuarioModel>
      * Procedimiento almacenado: Sp\_Listar\_Usuario
    - getUsuarioModificar
      * Descripción: Obtiene la información del usuario que se va a modificar.
      * Datos entrada: usuarioModel
      * Datos Salida: UsuarioModel
      * Procedimiento almacenado: Sp\_Listar\_Usuario\_Unico
    - Modificar Usuario
      * Descripción: Se modifica al usuario.
      * Datos entrada: usuarioModel
      * Datos Salida: bool
      * Procedimiento almacenado: Sp\_Update\_Usuario
    - get Usuarios Minuta
      * Descripción: Se obtienen los usuarios para la minuta de una reunión.
      * Datos entrada: idReunion
      * Datos Salida: List<UsuarioModel>
      * Procedimiento almacenado: Sp\_Listar\_Usuarios\_Minuta
* Data Tarea
  + Descripción: Esta clase extrae datos directamente desde la base de datos. Hace transacciones relacionadas con las tareas.
  + Métodos
    - Crear Tarea
      * Descripción. Insertar Tareas
      * Datos entrada: tareaModel
      * Datos Salida: bool
      * Procedimiento almacenado: Sp\_Insert\_Tarea
    - Modificar Tarea
      * Descripción. Modifica una Tarea
      * Datos entrada: tareaModel
      * Datos Salida: bool
      * Procedimiento almacenado: Sp\_Update\_Tarea
    - Eliminar Tarea
      * Descripción. Elimina una Tarea
      * Datos entrada: idTarea
      * Datos Salida: List<TareaModel>
      * Procedimiento almacenado: Sp\_Delete\_Tarea
    - getListarTarea
      * Descripción. Lista todas las Tareas
      * Datos entrada:
      * Datos Salida: List<TareaModel>
      * Procedimiento almacenado: Sp\_Listar\_Tarea
    - getListarTareaReunion
      * Descripción. Lista todas las Tareas de una reunión
      * Datos entrada: idReunion
      * Datos Salida: List<TareaModel>
      * Procedimiento almacenado: Sp\_Listar\_Reunion\_Tareas
    - getTareaModificar
      * Descripción. Obtiene la información de la tarea a modificar
      * Datos entrada: idTarea
      * Datos Salida: TareaModel
      * Procedimiento almacenado: Sp\_Listar\_Tarea\_Unico
    - getListarTareaMinuta
      * Descripción. Se obtienen los usuarios para la minuta de una reunión.
      * Datos entrada: idReunion
      * Datos Salida: List<TareaModel>
      * Procedimiento almacenado: Sp\_Listar\_Tareas\_Minuta
* Data Catalogo
  + Descripción: Esta clase extrae datos directamente desde la base de datos. Hace transacciones relacionadas con los catálogos.
  + Métodos
    - CrearRol
      * Descripción. Crear un nuevo rol en la base
      * Datos entrada: RolModel rolModel
      * Datos Salida:bool
      * Procedimiento almacenado:
    - CrearPuesto
      * Descripción.. Crea un nuevo puesto
      * Datos entrada: PuestoModel
      * Datos Salida:bool
      * Procedimiento almacenado:
    - CrearOficina
      * Descripción. crear una nueva oficina
      * Datos entrada: TipoReunionModel reunionModel
      * Datos Salida: bool
      * Procedimiento almacenado:
    - ModificarTipoReunion
      * Descripción..modifica una reunión
      * Datos entrada: TipoReunionModel
      * Datos Salida:bool
      * Procedimiento almacenado:
    - EliminarTipoReunion
      * Descripción.. Eliminar un tipo de reunión
      * Datos entrada: TipoReunionModel
      * Datos Salida:bool
      * Procedimiento almacenado:
    - ModificarOficina
      * Descripción. modifica una oficina
      * Datos entrada: OficinaModel
      * Datos Salida:bool
      * Procedimiento almacenado:
    - EliminarOficina
      * Descripción. Elimina oficinas
      * Datos entrada: OficinaModel
      * Datos Salida:bool
      * Procedimiento almacenado:
    - ModificarPuesto
      * Descripción.. Modifica puestos de los usuarios
      * Datos entrada: PuestoModel
      * Datos Salida:bool
      * Procedimiento almacenado:
    - EliminarPuesto
      * Descripción..Elimina puestos de usuario
      * Datos entrada: PuestoModel
      * Datos Salida:bool
      * Procedimiento almacenado:
    - ModificarRol
      * Descripción..Modifica roles de usuario
      * Datos entrada: RolModel
      * Datos Salida:bool
      * Procedimiento almacenado:
    - EliminarRol
      * Descripción..Elimina roles de usuario
      * Datos entrada: RolModel
      * Datos Salida:bool
      * Procedimiento almacenado:
    - getListarCatalogoCircuito
      * Descripción..Extrae los circuitos de la oficinas
      * Datos entrada:
      * Datos Salida:List<CircuitoModel>
      * Procedimiento almacenado:
    - getListarCatalogoPermisos
      * Descripción.. Extrae la lista de permisos para lo catalogos
      * Datos entrada:
      * Datos Salida: List<PermisosModel>
      * Procedimiento almacenado:
    - getListaCatalogoPuesto
      * Descripción.. Extrae una lista de puestos
      * Datos entrada:
      * Datos Salida:List<PuestoModel>
      * Procedimiento almacenado:
    - getListaCatalogoOficina
      * Descripción.. Extraer una lista de oficinas
      * Datos entrada:
      * Datos Salida:List<OficinaModel>
      * Procedimiento almacenado:
    - getListaCatalogoRol
      * Descripción..Extrae una lista de roles
      * Datos entrada:
      * Datos Salida:List<RolModel>
      * Procedimiento almacenado:
    - getListarTipoReunion
      * Descripción. Extrae la lista de tipos de reuniones
      * Datos entrada:
      * Datos Salida: List<TipoReunionModel>
      * Procedimiento almacenado:
* Data Dashboard
  + Descripción: Esta clase extrae datos directamente desde la base de datos. Hace transacciones relacionadas con los dashboard.
  + Métodos
    - getCantidadReuniones
      * Descripción. Obtiene la cantidad de reuniones realizadas en un rango de fechas
      * Datos entrada: fechaInicio, fechaFinal
      * Datos Salida: string
      * Procedimiento almacenado: Sp\_Cantidad\_Reuniones
    - getCantidadTiempoReuniones
      * Descripción. Obtiene el tiempo invertido en reuniones realizadas en un rango de fechas
      * Datos entrada: fechaInicio, fechaFinal
      * Datos Salida: string
      * Procedimiento almacenado: Sp\_Cantidad\_Tiempo\_Reuniones
    - getCantidadDineroReuniones
      * Descripción. Obtiene el dinero invertido en reuniones realizadas en un rango de fechas
      * Datos entrada: fechaInicio, fechaFinal
      * Datos Salida: string
      * Procedimiento almacenado: Sp\_Cantidad\_Dinero\_Reuniones
    - getCantidadAsistentesReuniones
      * Descripción. Obtiene la cantidad de usuarios que asistieron a las reuniones realizadas en un rango de fechas
      * Datos entrada: fechaInicio, fechaFinal
      * Datos Salida: List<CantidadAsistentesModel>
      * Procedimiento almacenado: Sp\_Asistencia\_Reuniones
    - getListaTiposReuniones
      * Descripción. Obtiene los tipos de reuniones realizadas en un rango de fechas
      * Datos entrada: fechaInicio, fechaFinal
      * Datos Salida: List<TipoReunionModel>
      * Procedimiento almacenado: Sp\_Nombre\_Tipo\_Reunion\_Mes
    - getCantidadAsistentesReuniones
      * Descripción. Obtiene la cantidad de usuarios que asistieron a una reunión
      * Datos entrada: idReunion
      * Datos Salida: CantidadAsistentesModel
      * Procedimiento almacenado: Sp\_Asistencia\_Reunion\_Unica
    - getDineroReunionUnica
      * Descripción. Obtiene el dinero invertido en una reunión
      * Datos entrada: idReunion
      * Datos Salida: int
      * Procedimiento almacenado: Sp\_Cantidad\_Dinero\_Por\_Reunion
    - getDuracionReunionUnica
      * Descripción. Obtiene la duración de una reunión
      * Datos entrada: idReuninio
      * Datos Salida: string
      * Procedimiento almacenado: Sp\_Tiempo\_Por\_Reunion
* Data Archivo
  + Descripción: Esta clase extrae datos directamente desde la base de datos. Hace transacciones relacionadas con los archivos.
  + Métodos
    - listarArchivos
      * Descripción. Lista los archivos de una reunión
      * Datos entrada:idElemento
      * Datos Salida: List<ArchivoModel>
      * Procedimiento almacenado: Sp\_Listar\_Archivo\_Reunion
    - EliminarArchivo
      * Descripción. Elimina un archivo
      * Datos entrada: TN\_Id\_Archivo
      * Datos Salida: bool
      * Procedimiento almacenado: Sp\_Delete\_Archivo\_Reunion\_Unico
* Data Agenda
  + Descripción: Esta clase extrae datos directamente desde la base de datos. Hace transacciones relacionadas con los dashboard.
  + Métodos
    - getReunionesData
      * Descripción. Extrae las reuniones para ser mostradas en la calendario
      * Datos entrada: string usuario
      * Datos Salida: List<ReunionModel>
      * Procedimiento almacenado: Sp\_Cantidad\_Reuniones
    - getAsistentesReunion
      * Descripción. Extrae los asistentes de una reunión
      * Datos entrada: string IdReunionAsistentes
      * Datos Salida: List<UsuarioModel>
      * Procedimiento almacenado: Sp\_Cantidad\_Reuniones
    - getTareasReunion
      * Descripción. Extrae las tareas de una reunión
      * Datos entrada: string IdReunionTareas
      * Datos Salida: List<TareaModel>
      * Procedimiento almacenado: Sp\_Cantidad\_Reuniones

# Base de Datos

La base de datos del sistema está realizada en SQL Server en la cual existen una serie de tablas, procedimientos y funciones que permiten llevar a cabo las funciones necesarias para el correcto funcionamiento del sistema.

USUARIO

**Nombre:** Sp\_Insert\_Usuario

**Descripción:** Este procedure se encarga de realizar la inserción de datos de los usuarios, y afecta las tablas T\_Usuario y T\_Usuario\_Rol.

**Datos de entrada:** Como datos de entrada recibe el usuario, contraseña, identificación, nombre, primer apellido, segundo apellido, correo electrónico, id del puesto asignado, id de la oficina para la cual trabaja y id del rol dentro del sistema.

**Datos de salida:** Retorna valido=1 si logró completar el procedimiento con éxito o valido=0 si ocurrió algún error o no pasó alguna restricción.

**Nombre:** Sp\_Delete\_Usuario

**Descripción:** Este procedure se encarga de hacer un delete lógico sobre los usuarios y afecta únicamente la tabla de T\_Usuario.

**Datos de entrada:** Recibe como único dato de entrada el id del usuario que se desea eliminar.

**Datos de salida:Datos de salida:** Retorna valido=1 si logró completar el procedimiento con éxito o valido=0 si ocurrió algún error o no pasó alguna restricción.

**Nombre:** Sp\_Update\_Usuario

**Descripción:** Este procedure se encarga de actualizar la información de los usuarios del sistema y afecta las tablas T\_Usuario y T\_Usuario\_Rol.

**Datos de entrada:** Recibe el id del usuario, usuario, contraseña, identificación, nombre, primer apellido, segundo apellido, correo electrónico, id del puesto laboral, id de la oficina a la cual pertenece y el id del rol dentro del sistema.

**Datos de salida:**Retorna valido=1 si logró completar el procedimiento con éxito o valido=0 si ocurrió algún error o no pasó alguna restricción.

TIPO REUNIÓN

**Nombre:** Sp\_Insert\_Tipo\_Reunion

**Descripción:** Este procedure se encarga de hacer el insert de los tipos de reunión y afecta la tabla T\_Tipo\_Reunion.

**Datos de entrada:** Recibe como único dato el nombre del tipo de reunión.

**Datos de salida:** Retorna valido=1 si logró completar el procedimiento con éxito o valido=0 si ocurrió algún error o no pasó alguna restricción.

**Nombre:** Sp\_Delete\_Tipo\_Reunion

**Descripción:** Este procedure se encarga de hacer el delete de los tipos de reunión y afecta la tabla T\_Tipo\_Reunion.

**Datos de entrada:** Recibe como único dato el id del tipo de reunión.

**Datos de salida:** Retorna valido=1 si logró completar el procedimiento con éxito o valido=0 si ocurrió algún error o no pasó alguna restricción.

**Nombre:** Sp\_Update\_Tipo\_Reunion

**Descripción:** Este procedure se encarga de hacer el update de los tipos de reunión y afecta la tabla T\_Tipo\_Reunion.

**Datos de entrada:** Recibe el id del tipo de reunión y nombre del tipo.

**Datos de salida:** Retorna valido=1 si logró completar el procedimiento con éxito o valido=0 si ocurrió algún error o no pasó alguna restricción.

TAREA

**Nombre:** Sp\_Insert\_Tarea

**Descripción:** Este procedure se encarga de realizar los insert de la tareas, afecta la tablas T\_Tarea y T\_Tarea\_Usuario.

**Datos de entrada:** Recibe el nombre de la tarea, descripción y una lista de id de usuarios los cuales serán los asignados de la tarea.

**Datos de salida:** Retorna valido=1 si logró completar el procedimiento con éxito o valido=0 si ocurrió algún error o no pasó alguna restricción.

**Nombre:** Sp\_Delete\_Tarea

**Descripción:** Este procedure se encarga de realizar los delete de la tareas, afecta la tablas T\_Tarea y T\_Tarea\_Usuario.

**Datos de entrada:** Recibe el id de la tarea a eliminar.

**Datos de salida:** Retorna valido=1 si logró completar el procedimiento con éxito o valido=0 si ocurrió algún error o no pasó alguna restricción.

**Nombre:** Sp\_Update\_Tarea

**Descripción:** Este procedure se encarga de realizar los update de la tareas, afecta la tablas T\_Tarea y T\_Tarea\_Usuario.

**Datos de entrada:** Recibe el id de la tarea, nombre de tarea, descripción y una lista de id de usuarios asignados a esta tarea.

**Datos de salida:** Retorna valido=1 si logró completar el procedimiento con éxito o valido=0 si ocurrió algún error o no pasó alguna restricción.

**Nombre:** Sp\_Insert\_Tarea\_Acuerdo

**Descripción:** Este procedure se encarga de realizar los update a tareas para poder asignar los acuerdos de cada una, afecta únicamente la tabla T\_Tarea.

**Datos de entrada:** Recibe el id de la tarea y el acuerdo de la misma.

**Datos de salida:** Retorna valido=1 si logró completar el procedimiento con éxito o valido=0 si ocurrió algún error o no pasó alguna restricción.

REUNIÓN

**Nombre:** Sp\_Insert\_Reunion

**Descripción:** Este procedure se encarga de realizar los insert de las reuniones y afecta las tablas T\_Reunion, T\_Temas, T\_Lista\_Tareas, T\_Agenda y T\_Archivo\_Reunion.

**Datos de entrada:** Se recibe el nombre de la reunión, id del tipo de la reunión, descripción, comentario, lugar de la reunión, fecha inicio y hora de inicio, lista de id de usuarios que serán invitados a la reunión, lista de temas de la reunión, lista de id de las tareas a llevar a cabo, lista de nombre de los archivos ligados a la misma y el nombre de usuario de la persona que está creado la reunión.

**Datos de salida:** Retorna valido=1 si logró completar el procedimiento con éxito o valido=0 si ocurrió algún error o no pasó alguna restricción, además se retorna el id de la reunión que se acaba de crear.

**Nombre:** Sp\_Delete\_Reunion

**Descripción:** Este procedure se encarga de realizar los delete de las reuniones y afecta las tablas T\_Reunion, T\_Temas, T\_Lista\_Tareas, T\_Agenda y T\_Archivo\_Reunion.

**Datos de entrada:** Recibe el id de la reunión a eliminar.

**Datos de salida:** Retorna valido=1 si logró completar el procedimiento con éxito o valido=0 si ocurrió algún error o no pasó alguna restricción.

**Nombre:** Sp\_Update\_Reunion

**Descripción:** Este procedure se encarga de realizar los update de las reuniones y afecta las tablas T\_Reunion, T\_Temas, T\_Lista\_Tareas, T\_Agenda y T\_Archivo\_Reunion.

**Datos de entrada:** Se recibe el id de la reunión, nombre de la reunión, id del tipo de la reunión, descripción, comentario, lugar de la reunión, fecha inicio y hora de inicio, lista de id de usuarios que serán invitados a la reunión, lista de temas de la reunión, lista de id de las tareas a llevar a cabo, lista de nombre de los archivos ligados a la misma.

**Datos de salida:** Retorna valido=1 si logró completar el procedimiento con éxito o valido=0 si ocurrió algún error o no pasó alguna restricción.

**Nombre:** Sp\_Valida\_Fecha\_Reunion

**Descripción:** Este procedure se encarga de validar a la hora de presionar el botón de asistir a la reunión, valida que la fecha y hora coincidan con la hora de inicio de la misma, además si se cumple la hora marca al usuario como que asistió a dicha reunión, y afecta la tabla T\_Agenda.

**Datos de entrada:** Se recibe el id de la reunión y el usuario.

**Datos de salida:** Retorna valido=1 si logró completar el procedimiento con éxito o valido=0 si ocurrió algún error o no pasó alguna restricción.

**Nombre:** Sp\_Finalizar\_Reunion

**Descripción:** Este procedure se encarga de hacer un update para finalizar las reuniones, y afecta la tabla T\_Reunion

**Datos de entrada:** Se recibe el id de la reunión.

**Datos de salida:** Retorna valido=1 si logró completar el procedimiento con éxito o valido=0 si ocurrió algún error o no pasó alguna restricción.

T\_LISTA\_TAREAS

**Nombre:** Sp\_Insert\_Lista\_Tareas

**Descripción:** Este procedure se encarga de hacer los insert de las listas de tareas de las reuniones, y afecta la tabla T\_Lista\_Tareas.

**Datos de entrada:** Se recibe el id de la reunión y la lista de tareas.

**Datos de salida:** No posee datos de salida.

**Nombre:** Sp\_Delete\_Lista\_Tareas

**Descripción:** Este procedure se encarga de hacer los delete de las listas de tareas de las reuniones, y afecta la tabla T\_Lista\_Tareas.

**Datos de entrada:** Se recibe el id de la reunión.

**Datos de salida:** No posee datos de salida.

ROL

**Nombre:** Sp\_Insert\_Rol

**Descripción:** Este procedure se encarga de insertar el Rol que luego se le asigna al usuario, afecta la tabla T\_Rol

**Datos de entrada:** Recibe como datos el nombre del rol, y el id del permiso que se le asigna a ese rol

**Datos de salida:** Retorna valido=1 si logró completar el procedimiento con éxito o valido=0 si ocurrió algún error o no pasó alguna restricción.

**Nombre:** Sp\_Delete\_Rol

**Descripción:** Este procedure se encarga de eliminar un Rol, afecta la tabla T\_Rol

**Datos de entrada:** Recibe como único dato el id del Rol

**Datos de salida:** Retorna valido=1 si logró completar el procedimiento con éxito o valido=0 si ocurrió algún error o no pasó alguna restricción.

**Nombre:** Sp\_Update\_Rol

**Descripción:** Este procedure se encarga de actualizar un Rol, afecta la tabla T\_Rol

**Datos de entrada:** Recibe como datos de entrada el id del Rol, el nuevo nombre y el permiso que se le asigna a ese rol

**Datos de salida:** Retorna valido=1 si logró completar el procedimiento con éxito o valido=0 si ocurrió algún error o no pasó alguna restricción.

OFICINA

**Nombre:** Sp\_Insert\_Oficina

**Descripción:** Este procedure se encarga de insertar una oficina que luego se le asigna al usuario, afecta la tabla T\_Oficina

**Datos de entrada:** Recibe como datos el nombre, código, id del circuito y la fecha de inicio de vigencia de la oficina

**Datos de salida:** Retorna valido=1 si logró completar el procedimiento con éxito o valido=0 si ocurrió algún error o no pasó alguna restricción.

**Nombre:** Sp\_Delete\_Oficina

**Descripción:** Este procedure se encarga de eliminar una oficina, afecta la tabla T\_Oficina

**Datos de entrada:** Recibe como único dato el id de la oficina

**Datos de salida:** Retorna valido=1 si logró completar el procedimiento con éxito o valido=0 si ocurrió algún error o no pasó alguna restricción.

**Nombre:** Sp\_Update\_Oficina

**Descripción:** Este procedure se encarga de actualizar la oficina, afecta la tabla T\_Oficina

**Datos de entrada:** Recibe como datos el id de la oficina el nombre y el código

**Datos de salida:** Retorna valido=1 si logró completar el procedimiento con éxito o valido=0 si ocurrió algún error o no pasó alguna restricción.

PUESTO

**Nombre:** Sp\_Insert\_Puesto

**Descripción:** Este procedure se encarga de insertar un puesto que luego se le asigna al usuario, afecta la tabla T\_Puesto

**Datos de entrada:** Recibe como datos el nombre del puesto y el salario que tiene ese puesto

**Datos de salida:** Retorna valido=1 si logró completar el procedimiento con éxito o valido=0 si ocurrió algún error o no pasó alguna restricción.

**Nombre:** Sp\_Delete\_Puesto

**Descripción:** Este procedure se encarga de eliminar un puesto , afecta la tabla T\_Puesto

**Datos de entrada:** Recibe como único dato el id del puesto

**Datos de salida:** Retorna valido=1 si logró completar el procedimiento con éxito o valido=0 si ocurrió algún error o no pasó alguna restricción.

**Nombre:** Sp\_Update\_Puesto

**Descripción:** Este procedure se encarga de actualizar un puesto , afecta la tabla T\_Puesto

**Datos de entrada:** Recibe como datos el id del puesto, el nombre y el salario.

**Datos de salida:** Retorna valido=1 si logró completar el procedimiento con éxito o valido=0 si ocurrió algún error o no pasó alguna restricción.

TEMAS

**Nombre:** Sp\_Insert\_Temas

**Descripción:**Este procedure se encarga de insertar los temas de una reunión, afecta la tabla T\_Temas

**Datos de entrada:** Recibe como datos el id de la reunión a la que está ligado y una lista de los temas que se van a asignar a esa reunión

**Datos de salida:** No posee datos de salida.

**Nombre:** Sp\_Insert\_Temas\_Acuerdo

**Descripción:** Este procedure se encarga de añadirle acuerdos a los temas, afecta la tabla T\_Temas

**Datos de entrada:** Recibe como datos de entrada el id del tema al cual se le va a asignar el acuerdo, y el nombre del acuerdo que se ingreso

**Datos de salida:** Retorna valido=1 si logró completar el procedimiento con éxito o valido=0 si ocurrió algún error o no pasó alguna restricción.

AGENDA

**Nombre:** Sp\_Insert\_Agenda

**Descripción:** Este procedure se encarga de ligar la reunión que se creo con todos los usuarios para luego colocar todo en la agenda, afecta la tabla T\_Agenda

**Datos de entrada:** Recibe como datos el id de la reunión a la que está ligado y una lista de los usuarios que se van a asignar a esa reunión

**Datos de salida:** No posee datos de salida.

**Nombre:** Sp\_Delete\_Agenda

**Descripción:** Este procedure se encarga borrar la tabla T\_Agenda de las reuniones, afecta la tabla T\_Agenda

**Datos de entrada:** Recibe como datos de entrada el id de la reunión

**Datos de salida:** No posee datos de salida

**Nombre:** Sp\_Listar\_Agenda

**Descripción:**Este procedure se encarga de listar todas las reuniones que posee un usuario, no afecta ninguna tabla

**Datos de entrada:** Recibe como datos de entrada el nombre de usuario

**Datos de salida:** Retorna válido =1 si ese usuario si posee reuniones, además del id de reunión, nombre, descripción, comentario, lugar, y la fecha de la reunión.

ARCHIVOS REUNIÓN

**Nombre:** Sp\_Insert\_Archivo\_Reunion

**Descripción:** Este procedure se encarga de insertar la lista de los archivos que tiene una reunión en específico, afecta la tabla T\_Archivo\_Reunion.

**Datos de entrada:** Recibe como datos de entrada el id de la reunión y una lista con el nombre de todos los archivos que se le asignaron a esa reunión.

**Datos de salida:** No posee datos de salida.

**Nombre:** Sp\_Listar\_Archivo\_Reunion

**Descripción:** Este procedure se encarga de listar todos los archivos que tenga una reunión en específico no afecta ninguna tabla

**Datos de entrada:** Recibe como datos de entrada el id de la reunión

**Datos de salida:** Retorna válido = 1 , el id del archivo, y la ruta del mismo

DASHBOARD

**Nombre:** Sp\_Cantidad\_Reuniones

**Descripción:** Este procedure se encarga de listar la cantidad total de las reuniones realizadas en rango de fechas.

**Datos de entrada:** Recibe dos fechas una inicial y otra final

**Datos de salida:** Retorna valido=1 si logró completar el procedimiento con éxito o valido=0 si ocurrió algún error o no pasó alguna restricción, además de la cantidad total de reuniones realizadas.

**Nombre:** Sp\_Cantidad\_Tiempo\_Reuniones

**Descripción:** Este procedure se encarga de retornar una lista de las cantidades de tiempo que se invirtieron en un rango de fechas no afecta ninguna tabla.

**Datos de entrada:** Recibe como datos de entrada dos fechas, una de inicio y una final

**Datos de salida:** Retorna valido=1 si logró completar el procedimiento con éxito o valido=0 si ocurrió algún error o no pasó alguna restricción, además un string que lleva todas las cantidades de tiempo que fueron invertidas.

**Nombre:** Sp\_Cantidad\_Dinero\_Reuniones

**Descripción:** Este procedure se encarga de retornar una lista de la cantidad de dinero que se invirtieron en un rango de fechas, no afecta ninguna tabla.

**Datos de entrada:** Recibe como datos de entrada dos fechas, una de inicio y una final

**Datos de salida:** Retorna valido=1 si logró completar el procedimiento con éxito o valido=0 si ocurrió algún error o no pasó alguna restricción, además un string que lleva todas las cantidades de dinero que fue invertido en ese rango de fechas.

**Nombre:** Sp\_Nombre\_Tipo\_Reunion\_Mes

**Descripción:** Este procedure se encarga de listar la cantidad de reuniones realizadas separadas por los tipos.

**Datos de entrada:** Recibe dos fechas una inicial y otra final

**Datos de salida:** Retorna valido=1 si logró completar el procedimiento con éxito o valido=0 si ocurrió algún error o no pasó alguna restricción, además retorna cada tipo de reunión con la cantidad de reuniones que se realizaron

**Nombre:** Sp\_Cantidad\_Dinero\_Por\_Reunion

**Descripción:** Este procedure se encarga de retornar la cantidad de dinero que se invirtió en una reunión en específico, no afecta ninguna tabla.

**Datos de entrada:** Recibe como datos de entrada el id de la reunión.

**Datos de salida:** Retorna valido=1 si logró completar el procedimiento con éxito o valido=0 si ocurrió algún error o no pasó alguna restricción, además de la cantidad de dinero invertida

**Nombre:** Sp\_Cantidad\_Tareas\_Finalizadas

**Descripción:** Este procedure se encarga de listar la cantidad de tareas finalizadas de una reunión.

**Datos de entrada:** Recibe id de la reunión.

**Datos de salida:** Retorna valido=1 si logró completar el procedimiento con éxito o valido=0 si ocurrió algún error o no pasó alguna restricción, además retorna la cantidad de tareas finalizadas y las que siguen activas.

**Nombre:** Sp\_Asistencia\_Reunion\_Unica

**Descripción:** Este procedure se encarga de retornar la cantidad de personas que fueron y las que no fueron a una reunión en específico.

**Datos de entrada:** Recibe como datos de entrada el id de la reunión.

**Datos de salida:** Retorna valido=1 si logró completar el procedimiento con éxito o valido=0 si ocurrió algún error o no pasó alguna restricción, además de la cantidad de personas que fueron y la cantidad de personas que no fueron.

**Nombre:** Sp\_Listar\_Unico\_Reunion\_Minuta

**Descripción:** Este procedure se encarga de listar todos los datos de la reunión necesarios para cargar la minuta.

**Datos de entrada:** Recibe id de la reunión.

**Datos de salida:** Retorna valido=1 si logró completar el procedimiento con éxito o valido=0 si ocurrió algún error o no pasó alguna restricción, además retorna el id de la reunión, nombre de la reunión, id del tipo de reunion, descripción, comentario, lugar de la reunión, fecha de inicio y fecha final de la misma, nombre del usuario que la creó y el nombre del tipo de reunión.

**Nombre:** Sp\_Listar\_Usuarios\_Minuta

**Descripción:** Este procedure se encarga de retornar la lista de usuarios de una reunión para colocarlos en la minuta.

**Datos de entrada:** Recibe como datos de entrada el id de la reunión.

**Datos de salida:** Retorna valido=1 si logró completar el procedimiento con éxito o valido=0 si ocurrió algún error o no pasó alguna restricción, además el nombre del usuario, primer apellido, segundo apellido, nombre puesto, y si asistió o no.

**Nombre:** Sp\_Listar\_Tareas\_Minuta

**Descripción:** Este procedure se encarga de listar las tareas de la reunión para cargarlas sobre la minuta

**Datos de entrada:** Recibe id de la reunión.

**Datos de salida:** Retorna valido=1 si logró completar el procedimiento con éxito o valido=0 si ocurrió algún error o no pasó alguna restricción, además retorna el nombre de la tarea, descripción, estado, los acuerdos,nombre de las personas ligadas a la tarea.

**Nombre:** Sp\_Listar\_Temas\_Minuta

**Descripción:** Este procedure se encarga de retornar la lista de temas de una reunión para colocarlos en la minuta.

**Datos de entrada:** Recibe como datos de entrada el id de la reunión.

**Datos de salida:** Retorna valido=1 si logró completar el procedimiento con éxito o valido=0 si ocurrió algún error o no pasó alguna restricción, además el nombre de la tarea y el acuerdo de la misma.

**Nombre:** Sp\_Asistencia\_Reuniones

**Descripción:** Este procedure se encarga de retornar la asistencia que se tuvo en un rango de fechas

**Datos de entrada:** Recibe como datos de entrada una fecha de inicio y una fecha final

**Datos de salida:** Retorna valido=1 si logró completar el procedimiento con éxito o valido=0 si ocurrió algún error o no pasó alguna restricción, además el mes, año, cantidad de asistencia, y cantidad de invitados en la reunión.

**Nombre:** Sp\_Tiempo\_Por\_Reunion

**Descripción:** Este procedure se encarga de devolver el tiempo total invertido en la reunión que se indica.

**Datos de entrada:** Recibe id de la reunión.

**Datos de salida:** Retorna valido=1 si logró completar el procedimiento con éxito o valido=0 si ocurrió algún error o no pasó alguna restricción, además retorna la duración total de la reunión.

# Aspectos de Seguridad

Para la seguridad del sistema se realizó un módulo Login para que los usuario se autentican en el sistema y que se pudiera acceder a este de acuerdo a los permisos que tenga el usuario.

Estos permisos son los siguiente:

**Acceso total:** Los usuarios con este permiso puede visualizar todas las funcionalidades del sistema. CRUD de reuniones, CRUD de Tareas, CRUD de catálogos, la agenda personal, los dashboard y asistir a reuniones.

**Acceso estándar:** Los usuarios de acceso estándar al ingresar al sistema solo podrán visualizar la agenda personal correspondiente al mismo y asistir a reuniones.

**Administrador catálogos:** Este usuario solo va a tener acceso a los CRUD de los catálogos, a la agenda personal y asistir a reuniones.

El sistema también tiene la opción de cerrar la sesión del usuario para mayor seguridad. Y en caso de que se ingrese un url de alguna ventana del sistema, si el usuario no ha iniciado sesión este lo devolverá al login. Para este aspecto de seguridad lo que se maneja son las variables session.

# Instalación en los diferentes ambientes

Se debe tener en cuenta los requisitos previos a la instalación lo cual se debe tener en cuenta la información de todo el requerimiento previo al inicio de la instalación. Los requisitos precios que cabe mencionar son:

* Tener la ubicación del sistema definida y lista para utilizarse
* Obtener el código del sistema.
* Tener el ambiente de instalación listo para utilizarse.
* Acceso directo al motor de base de datos con los script de configuración de la misma listos para ser consumidos por el sistema.
* Interfaces a las cuales se debe tener acceso Google Chrome y Mozilla Firefox.
* Se debe contar con un motor de base de datos Microsoft SQL Server 2017

Una vez abarcado cada requerimiento anteriormente mencionado se debe proceder a verificar los recursos de hardware y software necesarios, compilar y por último proceder con la instalación del sistema gestor de reuniones.

## 12.1 Requisitos de hardware y software

**Servidor**

Para el servidor necesario para implementación del sistema se detalla a continuación sus requisitos necesarios para la instalación del mismo ya sean servidores virtuales o físicos.

|  |  |  |
| --- | --- | --- |
| **Servidor 1** | | |
| **Dato** | **Valor mínimo** | **Valor recomendado** |
| Procesador | 1.8GHz | 2.5GHz |
| Memoria RAM | 8 GB hz | 12 GB hz |
| Tamaño Almacenamiento | 600 GB | 1 TB |

**Estaciones cliente**

Detalla los requisitos necesarios para la ejecución en las máquinas de los clientes o bien de usuarios de este sistema.

|  |  |  |
| --- | --- | --- |
| **Dato** | **Valor mínimo** | **Valor recomendado** |
| Procesador | 1.5GHz | 1.80GHz |
| Memoria RAM | 2 GB hz | 4 GB hz |
| Tamaño Almacenamiento | 256 GB | 500GB |

## 12.2 Compilación

Se mostrará un cuadro con las tareas que se deben realizar y en orden específico para lograr configurar el ambiente de compilación, lo cual permite corroborar que no se generan errores en el sistema.

|  |  |
| --- | --- |
| **Procedimiento de compilación** | |
| **Paso** | **Descripción** |
| **1** | Corroborar que se tenga configurado la herramienta de compilación de visual studio |
| **2** | Presionar el botón de compilar en la interfaz de visual studio. |
| **3** | Verificar que no se haya disparado ningún tipo de error. |

## 12.3 Instalación

En este apartado se mostrará la información necesaria para obtener una instalación efectiva del sistema, por lo cual se mostrarán detalladamente las acciones secuenciales necesarias para llevar a cabo la instalación. Para cada sección se deberá indicar claramente todos los pasos a seguir para completar la instalación.

Para dar una explicación más detallada se mostrará un cuadro a continuación con los pasos a seguir para la instalación del sistema.

|  |  |
| --- | --- |
| **Procedimiento de instalación** | |
| **Paso 1** | |
| **Tipo** | Base de datos |
| **Componente** | Microsoft SQL Server 2017 |
| **Permisos** | Permisos para ejecutar script. |
| **Descripción** | Se configura el servidor de base de datos, con los script y tablas necesarias para que el sistema logre consumir los datos. El orden en el cual se deben ejecutar los scripts “CrearTablas”, luego se ejecutan todos los que contienen los procedures de las diferentes tablas y por ultimo se debe verificar el documento “Script\_Datos\_Iniciales\_del\_Sistema” ya que en este encuentran algunos datos que son requeridos para poder inicializar la aplicación por primera vez. |
| **Paso 2** | |
| **Tipo** | Sistema de gestor de reuniones |
| **Componente** | Visual studio |
| **Permisos** | Permisos para poder compilar y ejecutar el código. |
| **Descripción** | Se debe de modificar la configuración donde se conecta a la base de datos, para esto se debe entrar en appsetting./json y colocar las credenciales que tenga el OIJ en su base de datos de SQL.  Al tener configurado todo lo anteriormente mencionado, se ejecuta el programa desde la aplicación de visual studio lo cual permite la instalación del sistema y empezará a correr sobre el navegador establecido. |

# 