* + 1. **Nivel 2 Calendar Synchronizer**

El componente Calendar Synchronizer se divide en los módulos:

* **CalendarSync**
* **CalendarScheduler**
* **Persistence**



* + - 1. **Módulo CalendarSync**

Modulo encargado de iniciar la ejecución del componente Calendar Synchronizer, siguiente a esto obtiene las urls de todos los calendarios almacenados en base de datos para que puedan ser programados y sincronizados mediante la invocación del módulo de programación.

Adicionalmente, expone el método para la invocación remota cada vez que se registran nuevos calendarios en la aplicación, esto con el objetivo que se tengan en cuenta por el módulo de programación.



* + - * 1. **Responsabilidades funcionales**

Las responsabilidades funcionales de este módulo corresponden a las definidas en el nivel 1 para el componente Calendar Synchronizer.

* + - * 1. **Restricciones**

Las restricciones de este módulo corresponden a las definidas en el nivel 1 para el componente Calendar Synchronizer.

* + - * 1. **Responsabilidades no funcionales**

Las responsabilidades no funcionales de este módulo corresponden a las definidas en el nivel 1 para el componente Calendar Synchronizer.

* + - * 1. **Ciclo de vida**
* Este módulo inicia dos hilos de ejecución: el principal para agendar la sincronización de todos los calendarios almacenados en base de datos, el secundario actúa como listener en caso de la inscripción de nuevos calendarios. Esta la puerta de entrada desde el componente CalendarCore.
  + - 1. **Módulo CalendarScheduler**

Este módulo hace uso del API Quartz para soportar el agendamiento de tareas. Estas tareas corresponden a la sincronización de los calendarios externos registrados en la aplicación.



* + - * 1. **Responsabilidades funcionales**

Las responsabilidades funcionales de este módulo corresponden a las definidas en el nivel 1 para el componente Calendar Synchronizer.

* + - * 1. **Restricciones**

Las restricciones de este módulo corresponden a las definidas en el nivel 1 para el componente Calendar Synchronizer.

* + - * 1. **Responsabilidades no funcionales**

Las responsabilidades no funcionales de este módulo corresponden a las definidas en el nivel 1 para el componente Calendar Synchronizer.

* + - * 1. **Ciclo de vida**
* Se programa una nueva tarea (Job) por cada calendario enviado.
  + - 1. **Módulo Persistence**

Modulo que permite el acceso a la base de datos de la aplicación mediante JDBC.



* + - * 1. **Responsabilidades funcionales**

Las responsabilidades funcionales de este módulo corresponden a las definidas en el nivel 1 para el componente Calendar Synchronizer.

* + - * 1. **Restricciones**

Las restricciones de este módulo corresponden a las definidas en el nivel 1 para el componente Calendar Synchronizer.

* + - * 1. **Responsabilidades no funcionales**

Las responsabilidades no funcionales de este módulo corresponden a las definidas en el nivel 1 para el componente Calendar Synchronizer.

* + - * 1. **Ciclo de vida**
* Cada vez que se accede a las funcionalidades de este módulo se crea la conexión a base de datos, al terminar la ejecución de la funcionalidad utilizada la conexión es cerrada.
  + - 1. **Conexión CalendarSync a CalendarScheduler**

La conexión entre CalendarSync y CalendarScheduler se crea para permitir que por cada calendario obtenido en el módulo CalendarSync se programe una tarea (Job) repetitiva en el módulo CalendarScheduler que permita la sicronización de los horarios de un usuario de acuerdo al calendario externo.



* + - * 1. **Ciclo de vida**

CalendarSync solicita a CalendarScheduler la programación de tareas para la sincronización de uno o más calendarios, esto al inicio de la aplicación y cada vez que se registre un nuevo calendario cuando la aplicación este en ejecución.

* + - 1. **Conexión CalendarSync, CalendarScheduler a Persistence**

La conexión entre CalendarSync y Persistence se crea para permitir la consulta de todos los calendarios inscritos en base de datos.

La conexión entre CalendarScheduler y Persistence se crea para actualizar en base de datos los horarios de los usuarios para los cuales se identifiquen cambios en sus calendarios externos.



* + - * 1. **Ciclo de vida**

CalendarSync-Persistence: Cuando es iniciado el componente Canedar Synchronizer se invoca el método de persistencia para obtener todos los calendarios.

CalendarScheduler-Persistence: Cuando se identifica que hay cambios en los horarios de un usuario se invoca el método respectivo en persistencia para actualizar los horarios en base de datos.

* + 1. **Nivel 3 Calendar Synchronizer**

En este nivel se definen las clases de las cuales se componen cada uno de los módulos del componente Calendar Synchronizer.



* + - 1. **Clase Persistencia**
         1. **Descripción**

Contienen los métodos que permite ejecutar las consultas a base de datos de los calendarios inscritos por los usuarios y el registro de los nuevos horarios obtenidos de los calendarios externos.

****

* + - * 1. **Responsabilidades**

|  |  |
| --- | --- |
| **Método:** consultarCalendariosInscritos() | |
| **Entrada:** | N/A |
| **Descripción:** | - Obtiene la conexión a base de datos  - Consultan las urls de todos los calendarios inscritos, por cada uno también se consulta el login y password del usuario asociado  - Cierra la conexión a base de datos |
| **Salida:** | Collection que tiene todos los calendarios consultados  url, login y password |
| **Clases utilizadas** | Ninguna |

|  |  |
| --- | --- |
| **Método:** consultarHorariosUsuario(String) | |
| **Entrada:** | String **login** : Login del usuario |
| **Descripción:** | - Obtiene la conexión a base de datos  - Consulta los horarios de un usuario filtrando por el login recibido como entrada.  - Cierra la conexión a base de datos |
| **Salida:** | Lista de los horarios del usuario |
| **Clases utilizadas** | Ninguna |

|  |  |
| --- | --- |
| **Método:** almacenarHorariosUsuario(String, Collection) | |
| **Entrada:** | String **login** : Login del usuario  Collection **horarios** : Horarios que serán asociados |
| **Descripción:** | - Obtiene la conexión a base de datos  - Actualiza para el usuario sus horarios en base de datos  - Cierra la conexión a base de datos |
| **Salida:** | Registro de horarios exitoso |
| **Clases utilizadas** | Ninguna |

* + - 1. **Interfaz SincronizacionListenerRemote**
         1. **Descripción**

Interfaz con la firma del método que es invocado de forma remota por RMI para programar la sincronización de un calendario que acaba de ser registro en la aplicación.

Debe heredar de java.rmi.Remote y sus métodos deben propagar la excepción java.rmi.RemoteException



* + - * 1. **Responsabilidades**

|  |  |
| --- | --- |
| **Método:** programarSincronizacion(String, String, String, long) | |
| **Entrada:** | String **login**  String **password**  String **urlCalendario**  Long **tiempoSincronizacion** |
| **Descripción:** | Parte que puede ser accedida de forma remota |
| **Salida:** | N/A |
| **Clases utilizadas** | <<Extends>> Remote |

* + - 1. **Clase SincronizacionListener**
         1. **Descripción**

Clase que implementa los métodos definidos en la interfaz SincronizacionListenerRemote

****

* + - * 1. **Responsabilidades**

|  |  |
| --- | --- |
| **Método:** programarSincronizacion(String, String, String, long) | |
| **Entradas:** | String **login** : Login del usuario que registra el calendario  String **password** : Password de acceso al calendario  String **urlCalendario** : URL del calendario externo  Long **tiempoSincronizacion**: Tiempo en minutos que indica cada cuanto debe ser sincronizado el calendario. |
| **Descripción:** | - Invoca el método ***actualizarAgendamiento*** de la clase ***CalendarSynchronizer*** para que se programe un nuevo Job para sincronizar los horarios del calendario externo |
| **Salida:** | Nuevo calendario marcado para ser agendado |
| **Clases utilizadas** | *CalendarSynchronizer* |

* + - 1. **Clase CalendarSynchronizer**
         1. **Descripción**

Esta clase tiene una única instancia durante el ciclo de vida del CalendarSync. Contiene los métodos para administrar el agendamiento de Jobs por cada calendario según sea requerido.

****

* + - * 1. **Responsabilidades**

|  |  |
| --- | --- |
| **Método:** getInstance() | |
| **Entrada:** | N/A |
| **Descripción:** | - Crea la instancia de la clase si no existe una instancia actual y la retorna.  - Obtiene los parámetros para acceder a los calendarios registrados en base de datos. Para esto invoca el método ***consultarAccesoCalendariosUsuarios***de la clase ***Persistencia***  - Con la información de los calendarios consultados invoca el método *inicializarAgendarmiento()::void* para programar la sincronización de cada calendario. |
| **Salida:** | Instancia única de la clase |
| **Clases utilizadas** | ***Persistencia*** |

|  |  |
| --- | --- |
| **Método:** inicializarAgendamiento () | |
| **Entrada:** | N/A |
| **Descripción:** | Invoca el método ***agendarSincronizacion***de la clase ***SincronizacionCalendarioScheduler*** por cada uno de los calendarios consultados en el método *getInstance* |
| **Salida:** | Se agendan todos los calendarios en Quartz para ejecutar la sincronización. |
| **Clases utilizadas** | ***SincronizacionCalendarioScheduler*** |

|  |  |
| --- | --- |
| **Método:** actualizarAgendamiento (String, String, String, Long) | |
| **Entrada:** | String **login** : Login del usuario  String **password** : Clave de acceso al calendario externo  String **url** : URL del calendario externo  Long **tiempoSincronizacion**: Tiempo en minutos que indica cada cuanto debe ser sincronizado el calendario. |
| **Descripción:** | Invoca el método ***agendarSincronizacion*** de la clase ***SincronizacionCalendarioScheduler*** enviando los parámetros recibidos como entradas. |
| **Salida:** | Se agenda la sincronización de un nuevo calendario en Quartz |
| **Clases utilizadas** | ***SincronizacionCalendarioScheduler*** |

* + - 1. **Clase SincronizacionCalendarioScheduler**
         1. **Descripción**

Clase encarga del agendamiento de Jobs que permiten sincronizar los horarios de un calendario externo con los del asociado al usuario en base de datos.

****

* + - * 1. **Responsabilidades**

|  |  |
| --- | --- |
| **Método:** agendarSincronizacion(String, String, String, long) | |
| **Entrada:** | String **login** : Login del usuario  String **password** : Clave de acceso al calendario externo  String **url** : URL del calendario externo  Long **tiempoSincronizacion**: Tiempo en minutos que indica cada cuanto debe ser sincronizado el calendario. |
| **Descripción:** | - Utiliza las clases del API de Quartz para programar un Job del tipo ***SincronizacionCalendarioJob*.**  - Se deben configurar tres parámetros al Job: url, login y password los valores corresponden a los recibidos como entradas  - El Job es programado para que se ejecute durante todo el ciclo de vida del módulo, la frecuencia de repetición depende del campo de entrada **tiempoSincronizacion.** |
| **Salida:** | Se programa la ejecución de un Job de forma indefinida con repeticiones de acuerdo al tiempo recibido como entrada |
| **Clases utilizadas** | ***SincronizacionCalendarioJob*** |

* + - 1. **Clase SincronizacionCalendarioJob**
         1. **Descripción**

****

* + - * 1. **Responsabilidades**

|  |  |
| --- | --- |
| **Método:** execute(JobExectionContext) | |
| **Entrada:** | JobExectionContext **jec** : Contexto de ejecución del job |
| **Descripción:** | - Permite dio de objeto recibido como entrada obtiene los parámetros url, login y password  - Con los parámetros del paso anterior descarga el archivo que representa el calendario (.ics)  - Convierte el archivo previamente obtenido en una lista de horarios. Para esto invoca el método ***convertirCalendarioExterno*** de la clase ***ICSParser***  - Consulta los horarios actualmente registrados en base de datos para el usuario con el login recibido por parámetro. Para esto invoca el método ***consultarHorariosUsuario*** de la clase ***Persistencia***  - Compara y verifica diferencias entre los horarios obtenidos en los dos pasos anteriores.  - Si en el paso anterior se encuentran diferencias entre los horarios, se invoca el método ***almacenarHorariosUsuario*** de la clase ***Persistencia*** para ejecutar la actualización de los horarios en base de datos. |
| **Salida:** | Sincronización de un calendario realizada y horarios actualizados en base de datos en caso de ser necesario. |
| **Clases utilizadas** | ***ICSParser, Persistencia*** |

* + - 1. **Clase ICSParser**
         1. **Descripción**

Esta clase ofrece la lógica que permite descartar, leer e interpretar los archivos de extensión “.ics” de tal forma que la información contenida en ellos pueda ser entendida y utilizada por todo el módulo.



* + - * 1. **Responsabilidades**

|  |  |
| --- | --- |
| **Método:** convertirCalendarioExterno(File) | |
| **Entrada:** | File calendario : Archivo .ics descargado |
| **Descripción:** | Se encarga de leer el archivo recibido como entrada e interpretarlo para poderlo convertir a una lista con los horarios que este contiene. |
| **Salida:** | Lista con los horarios asociados al calendario |
| **Clases utilizadas** | N/A |