Espacios colaborativos

**Solicitud de reserva**

* **Texto

  Descripción generada automáticamente**Además de todos los datos que brinda el enunciado acerca de qué debe contener una solicitud de reserva, se añadió un código de la solicitud para luego en la reserva propiamente dicha, se pueda hacer referencia a la solicitud de su reserva, para que, físicamente, cuando un solicitante esté en la recepción de la sede el día de la reserva, el recepcionista (operador) pueda, a través de la interfaz del sistema, saber los datos de la reserva que requiere el enunciado y otros (asistentes, accesorios, espacio, etc.).
* Otra alternativa para resolver este requerimiento (no funcional) es volver a poner los datos del solicitante, algunos de la solicitud de la reserva, en la reserva concreta o poner la solitud (para esta opción se da por hecho que el estado debe ser CONFIRMADA) en la reserva, pero no lo hice así para no poner la misma información dos veces.
* No se especifica si la solitud puede tener más de un medio de notificación, por cuestiones de simplicidad (tiempo en escenario de rendir un parcial) se planteó que solo tiene uno, se creó la entidad MedioNotificación para hacer la propuesta extensible para que un futuro una solitud sí pueda tener más de un medio de notificación y que ese atributo pueda ser una lista. Se implementó el patrón STRATEGY y ADAPTER para modelarlas.
* La cantidad de asistentes es un entero y no una lista de asistentes con sus atributos porque a la hora de reservar solo importa la cantidad y no la información de cada uno, esto se pide en la lista que luego en el mensaje recordatorio solicitará.
* El dato del solicitante es importante para saber a quién corresponde esa solicitud y luego y también para luego notificarle el estado de su solicitud.

**Estado de las solicitudes**

Un requerimiento es asegurar la trazabilidad en todo momento del estado de la solicitud, para lograr esto se creó en la solicitud de la reserva una lista de estados, donde cada instancia tendrá el estado concreto, la fecha y hora en la que pasó a ese estado y el usuario para concretar la auditoría (se asume que en cuanto a “usuario” se refiere al operador que realizó el cambio de estado y es el ÚNICO que va a usar el sistema, entonces se decidió no crear la entidad “USUARIO” con nombre y contraseña, además **el enunciado dice que el sistema de login y autenticación es externo**).

Diagrama

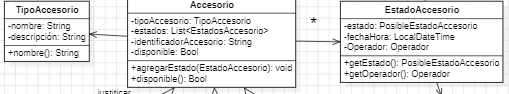
Descripción generada automáticamente

**Reserva**

Al tener que poder gestionar las reservas, se crea la clase RESERVA que contendrá su código (con el que el solicitante, presencialmente, podrá verificar que efectivamente él hizo esa reserva), el código de la solicitud de la reserva, el espacio de la sede exacto y los asistentes de dicha reserva.

**Accesorios (elementos)**

CONSIDERACIÓN: Se dar por hecho que solo se puede reservar un accesorio por cada reserva (una notebook, un proyector, etc.), a excepción de las sillas (que la cantidad se puede deducir por la cantidad de asistentes).

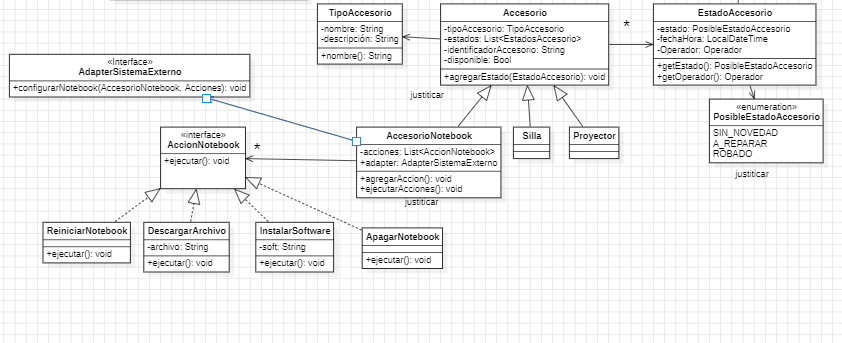


Se tiene el tipo modelado como una clase ya que pueden aparecer otros (extensibilidad), los estados están modelados con la misma analogía que los estados de la solicitud de la reserva, un identificador y un booleano “disponible” que representa si el accesorio está listo para su uso. Se creó como una clase y no como una interface porque hay atributos que comparten todos los accesorios y algunos (solo la notebook en este caso pero pueden aparecer otros como equipo de audio, una pc de escritorio) requieren ser configurados con comportamiento propio.

**Accesorio “Notebook”**

“*A los elementos de tipo “notebooks” se los deberá poder customizar con la intención de tenerlos configurados según se requiera. Para ello, el Sistema deberá permitir que el interesado defina una serie de acciones (previamente implementadas) a realizar en cada una de las notebooks: instalar un software, reiniciar el equipo, apagar el equipo,* descargar un archivo, etc. Estas secuencias de acciones deberán poder ser configuradas completamente por el usuario que realiza la reserva, o también pueden existir pre-configuraciones realizadas por un administrador, que deberán poder ser seleccionables por el interesado a la hora de realizar la reserva.”

Los posibles estados de un accesorio se abstrajeron como un enumerado porque se considera que no tiene mucha (o la necesaria) frecuencia al cambio.



Se implementó el patrón COMMAND para poder reflejar las acciones con las que se pueden configurar las notebooks, primero porque todas las customizaciones tienen su propia lógica para ejecutarse. El accesorio “notebook” tiene una lista de comandos a realizar y que se efectuará en el método “ejecutarAcciones”. Es donde se hará llamado al sistema externo y “pasarle” las acciones a ejectuar junto con la notebook. La integración con el sistema externo es mediante un adapter para poder seguir con en desarrollo de la solución, ya que es el sistema externo quién realiza esta tarea y no sabemos cómo.

**Recordatorio**

*Un día antes de la reserva el Sistema deberá enviarle una notificación al solicitante recordándole del evento y solicitando el listado de los asistentes. Este listado deberá ser cargado a través de una interfaz gráfica indicando apellido, nombre y DNI de cada asistente.*

La clase “TransmisorDeRecordatorios” es la que se encarga de mandar los mensajes con los recordatorios a cada persona que realizó una reserva. Va a ser una **tarea programada** que se ejecutará un día antes de la fecha estipulada en la reserva.  
El método principal del componente necesita un CONTACTABLE, se decidió crear una interface porque en un futuro también se podría querer notificar a todos los asistentes de la reserva, además como también se solicita los datos de los mismos en el mensaje, no está mal haber modelado a los asistentes con una clase, con nombre, apllido, dni (para identificarse) y la reserva a la que pertence. Entonces en un futuro pueden implementar dicha interface para poder ser contactados. Y también se requiere el MENSAJE, con el remitente y el contenido del mensaje.

**Finalización de reservas**

*El bloqueo del usuario además deberá poder ser realizado por un Sistema contable externo, en caso de que el mismo esté adeudando algún pago a la organización.*

Para permitir que el sistema contable pueda definir si un reservador está habilitado o no, en la clase Solicitante se creó el atributo “tieneDeuda: Bool” que permitirá saber el usuario está bloqueado o no. El sistema contable puede simplemente tener permisos de modificación en la tabla solicitantes de nuestra base de datos y nosotros vemos el estado reflejado en nuestro repositorio y acceder al dato.

**Consideración**



Cuando se habla de gestionar las sedes, se deduce que el sistema cuenta con más de una, entonces en la clase, además de los espacios, también se tiene la dirección, que no está normalizada por simpleza de la solución.



Cada reserva va a poder exportarse con la utilización de algún método de exportación, que son las que se mencionan. Para cumplir este requerimiento se implementó el patrón STRATEGY junto con el patrón ADAPTER