Documento de Diseño e Implementación.

Sistema de Reserva de aulas - FCEFYN.

26/06/2017

Grupo cmd.

Benítez Jeremías, Rao Maximiliano, Monsierra Lucas.

Índice.

3. DATA DESIGN  3

3.1  Data Description  3

3.2  Data Dictionary  3

4. COMPONENT DESIGN  3

5. HUMAN INTERFACE DESIGN  4

5.1  Overview of User Interface  4

5.2  Screen Images 4

5.3  Screen Objects and Actions 4

6. UNIT TESTS

7. REQUIREMENTS MATRIX  4

Historial de revisiones.

|  |  |  |  |
| --- | --- | --- | --- |
| Versión | Fecha | Observaciones | Autor |
| 1.0.0 | 23/06/17 | Creación del documento.  Se agregó la Introducción. | Monsierra, Lucas. |
| 1.0.1 | 24/06/17 | Se agregó la sección Diseño de datos. | Monsierra, Lucas. |

Introducción.

El Documento de Diseño de Software (SDD), es un documento que proporciona la documentación que se utilizará para ayudar en el desarrollo del software, proporcionando los detalles de cómo se debe construir.

Dentro del Documento de Diseño de Software se encuentran la documentación narrativa y gráfica del diseño de software para el proyecto, incluyendo diagramas de clase, de secuencia, paquetes, etc. y otra información de requisitos de soporte. Además se incluirán en éste las pruebas unitarias y la descripción de los patrones de diseño Observer, Singleton y Strategy utilizados para el desarrollo del proyecto.

## Propósito.

El propósito del SDD es proporcionar una descripción del diseño de un sistema lo suficientemente completo como para permitir el desarrollo del software, así tener una comprensión de lo que se va a construir y cómo se espera que se construya.

## Alcance.

Este SDD se centra en descripción de bajo nivel del sistema y sus partes críticas, haciendo hincapié en la generación de los documentos, diagramas, pruebas unitarias y la codificación de los documentos.

## Definiciones y Abreviaturas.

* BDD: Base de datos.
* DER: Diagrama de Entidad-Relación.
* SDD: Documento de Diseño de Software (Document Desing Software).

## Referencias.

Template SDD: <https://sovannarith.files.wordpress.com/2012/07/sdd_template.pdf>

Diseño de datos.

## Estructura de datos interna del software.

Los datos residirán localmente en la memoria y se organizarán en función de las clases definidas más adelante en este documento. Dado que nuestro sistema puede considerarse centrado en los datos, las clases que manejan los datos se aislarán y se accederá a través de un sistema Model-ViewController, descripto en el Documento de Arquitectura. Todos datos de necesarios se solicitarán a la BDD en la inicialización de la aplicación y se actualizarán según sea necesario en función de las acciones del usuario.

El almacenamiento permanente de la información del usuario se realizará utilizando una base de datos local, primeramente diseñada en MySQL, y luego transformada a SQLite para facilitar la interacción de ésta con la aplicación. La Sección 3.8 (Componente del servidor) cubre este tema con mayor detalle. Las consultas de ésta se harán utilizando el mismo lenguaje SQLite.

## Estructura global de datos.

La estructura de datos global de esta aplicación se caracteriza mejor por la base de datos. La estructura de la base de datos muestra los datos implicados en la aplicación en su sentido más puro. Los usuarios del sistema nunca accederán a ésta directamente, sino que emitirán peticiones.

## Estructura de datos temporales.

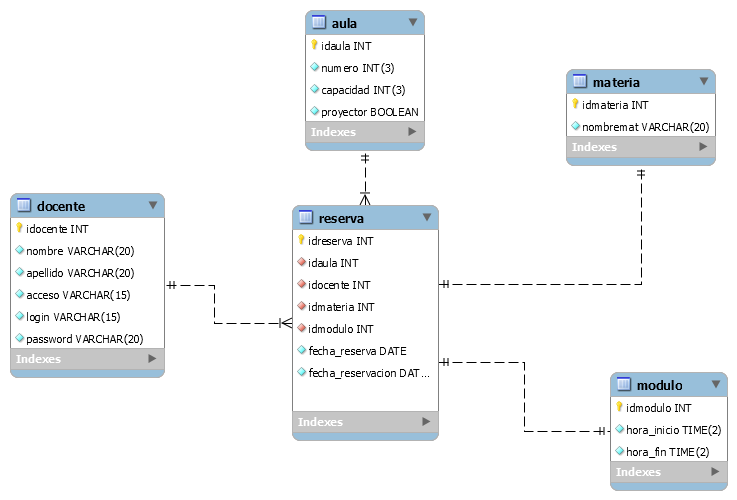
Las estructuras temporales de datos, se refieren a los objetos de datos que se crean en el sistema, los cuales sólo existirán durante el tiempo que la aplicación se ejecuta y posteriormente se destruirán.

## Diagrama de Entidad-Relación.

En este diagrama se trabaja con un nivel de abstracción más bajo, ya que se muestran las tablas en donde se guardará la información de la aplicación.

Para el caso de los profesores por ejemplo, se guardarán en tabla un ID, el nombre, apellido, acceso, login y contraseña. Para las reservas, se querrá guardar un ID, el aula reservada, el docente que la reservó, la materia, fecha y módulo en el que se reservó. El aula a su vez tendrá un ID, un número, información sobre su capacidad y si posee proyector o no. Para las materias se almacenará un ID y su nombre, y por último, para los módulos se almacenarán con un ID, una fecha de inicio y una de fin.

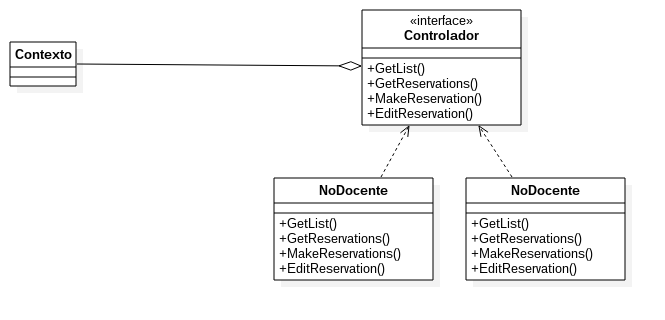
El sistema se debe diseñar para que solamente los usuarios Docentes puedan modificar el contenido de la tabla Reservas (de su propia reserva). Las otras tablas sólo podrán ser modificadas por los administradores de la aplicación. De esta manera se proporciona seguridad en los datos.



Patrones de diseño

La simplificación del desarrollo, y la posterior expansión, de este sistema de software, es el motivo que impulsó a adoptar tres patrones de diseño ampliamente utilizados en otros proyectos. A saber: Strategy, Singleton y Observer.

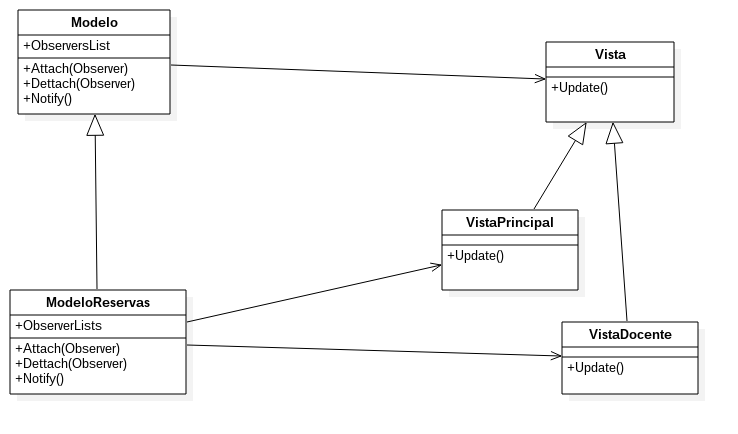
# Strategy

En este caso, la utilización de este patrón nos permite la implementación de los dos tipos de usuario posibles, Docente y NoDocente, y el intercambio de los mismos en tiempo de ejecución, lo cual permitirá que los usuarios se registren como docentes en caso de tener una cuenta en el sistema, y accedan a las funcionalidades pertinentes.

Cabe destacar que, si bien la clase NoDocente implementa todos los métodos de Controlador, no toda la funcionalidad será accesible desde la misma. Así, en caso de que un no docente intente acceder a estos métodos restringidos, un mensaje de error será mostrado.

# Observer

Este patrón define un tipo especial de relación uno-a-muchos entre clases. La principal función de dicha relación, es la capacidad de poder notificar a los objetos de una clase, cuando el estado de otra cambió.

En este proyecto, el patrón Observer se utiliza para mantener dos vistas actualizadas según los datos que ocurran en la base de datos. Si un Docente realiza una nueva reserva, tanto la vista principal (encargada de mostrar el listado de aulas con sus reservas si las hubiere), como la vista de docentes (en la cual, una vez registrados, los mismos pueden obtener detalles sobre sus reservas); son notificadas instantáneamente, y las mismas pueden modificar los valores mostrados. 

Esto nos permite ofrecer en todo momento la información más actual de la base de datos, evitando así conflictos que pudieran surgir por la visualización de datos que ya no sean válidos.

# Singleton

El patrón Singleton fue utilizado en la representación de la base de datos dentro de la aplicación. Esto da la posibilidad de acceder a la base de datos mediante una única instancia de la clase que la warpea, evitando así un uso ineficiente de la memoria el cual, al crecer la base de datos, podría afectar fuertemente el funcionamiento del sistema.