Documento de Arquitectura

Sistema de Reserva de aulas - FCEFYN.

26/06/2017

Grupo cmd.

Benítez Jeremías, Rao Maximiliano, Monsierra Lucas.

Índice.

Contenido

[Historial de revisiones. 2](#_Toc485941669)

[Introducción. 3](#_Toc485941670)

[Propósito. 3](#_Toc485941671)

[Alcance. 3](#_Toc485941672)

[Referencias. 3](#_Toc485941673)

[Arquitectura del sistema 4](#_Toc485941674)

[Representación arquitectónica. 4](#_Toc485941675)

[Objetivos y limitaciones. 5](#_Toc485941676)

[Vista Lógica. 6](#_Toc485941677)

[Vista Física. 6](#_Toc485941678)

[Vista de Desarrollo. 6](#_Toc485941679)

[Vista de Procesos. 7](#_Toc485941680)

[Vista de Escenarios. 8](#_Toc485941681)

[Patrón arquitectónico. 10](#_Toc485941682)

[Beneficios en relación a los Requerimientos No Funcionales. 10](#_Toc485941683)

[Vista. 11](#_Toc485941684)

[Controlador. 11](#_Toc485941685)

[Modelo. 11](#_Toc485941686)

[Pruebas de integración. 12](#_Toc485941687)

Historial de revisiones.

|  |  |  |  |
| --- | --- | --- | --- |
| Versión | Fecha | Observaciones | Autor |
| 1.0.0 | 26/05/17 | Creación del documento. | Monsierra, Lucas. |
| 1.0.1 | 27/05/17 | Primera versión del documento.  Índice.  Incisos 1 a 2.2. | Benítez, Jeremías. |
| 1.0.2 | 20/06/17 | Correcciones varias.  Agregadas Vistas “4+1” con diagramas. | Monsierra, Lucas. |
| 1.0.3 | 21/06/17 | Correcciones en diagramas.  Correcciones en sección Patrón de diseño.  Agregados beneficios del patrón MVC. | Monsierra, Lucas. |
| 1.0.4 | 22/06/17 | Agregadas descripciones MVC. | Monsierra, Lucas. |
| 1.0.5 | 22/06/17 | Correcciones en diagramas. | Monsierra, Lucas. |
| 1.0.6 | 22/06/17 | Finalizadas secciones Arquitectura del Sistema y Patrón Arquitectónico.  Agregado el Índice. | Monsierra, Lucas. |

Introducción.

Este documento contiene una visión general del Sistema de Reserva de aulas – FCEFYN (de ahora en más “Sistema”). El propósito principal de este Sistema es mejorar la convivencia en el ámbito de la institución, así como también simplificar procesos administrativos permitiendo que, tanto docentes como no docentes, desarrollen más eficientemente sus actividades dentro de la institución.

## Propósito.

El fin de este documento es expresar las decisiones arquitecturales importantes que se tomaron durante los procesos de diseño y planificación del Sistema. Sirve también para que las personas encargadas de diseñar, planear e implementarlo puedan entender mejor el enfoque que se espera que tomen a la hora de realizar sus tareas. Proporciona una descripción de alto nivel de los objetivos de la arquitectura, los distintos casos de uso del Sistema y los estilos arquitectónicos. Este marco permite entonces el desarrollo de los criterios de diseño y documentos que definen los estándares técnicos y de dominio en detalle.

Con el fin de representar el Sistema con la mayor precisión posible, la estructura de este documento se basa en el modelo de vista de la arquitectura “4+1”, el cual se describe en secciones posteriores.

## Alcance.

El alcance de este documento incluye al Sistema en su totalidad. Esto es, la descripción de sus interfaces gráficas, bases de datos, así también como de las interfaces que permitan la comunicación entre los distintos componentes que forman parte de este proyecto, es decir aspectos de diseño que se consideran arquitectónicamente significativos y fundamentales.

1.3 Definiciones y Abreviaturas

**// Editar después de terminar el documento.**

## Referencias.

Sommerville, I., (2011), Ingeniería de software, México, Pearson.

Freeman, E. et al, (2004) Head First Design Patterns, EEUU, O'Reilly.

Wikipedia, https://es.wikipedia.org/

Arquitectura del sistema

La arquitectura del Sistema fue diseña para cumplir con determinados objetivos, a saber:

* Cumplir con los requisitos presentados por la materia en la que la aplicación fue inicialmente pensada y diseñada.
* Facilitar la futura modificación. Para esto se buscó que el sistema fuese modular y escalable de forma que la inclusión de futuras mejoras no representara un problema para los equipos involucrados.
* Brindarle mayor flexibilidad al Sistema, permitiendo así que en un futuro el mismo pase a formar parte de otro sistema de mayor alcance.

## Representación arquitectónica.

Cualquier sistema que se desee implementar necesita ser representado desde distintos puntos de vista. Al igual que para construir una casa se necesita proveer a las partes que van a estar implicadas en su construcción diversos planos que representen la misma vivienda, vista desde distintos aspectos (planos del tendido eléctrico, planos de la estructura en sí, planos de las cañerías, etc.); un sistema de software debe representarse desde distintos puntos de vista.

Esto es así para ofrecer una representación más sólida de lo que se desea implementar, evitando que existan vaguedades o ambigüedades en la descripción del mismo. Uno de los beneficios de actuar de esta forma es que se disminuye la probabilidad de que un desarrollador deba tener que tomar por su cuenta decisiones de diseño, las cuales deberían ser tomadas por el equipo correspondiente. Esto lleva, en general, a sistemas más consistentes.

Teniendo en cuenta esto, se optó por detallar la arquitectura del Sistema utilizando las vistas definidas en el modelo “4+1”. La misma cuenta, como su nombre infiere, con cuatro vistas (lógica, física, de despliegue y de procesos) más una que las relaciona (de escenarios):

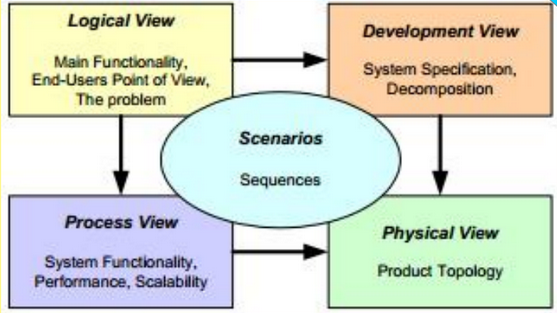


Diagrama básico de la vista 4+1

Para representar cada una de estas vistas se utilizarán diagramas UML. A continuación, se incluye una breve descripción de la finalidad y los diagramas usados en cada una:

* **Vista Lógica:** Esta se encarga de representar la funcionalidad principal de los usuarios finales. Se especifican las abstracciones clave en el sistema como objetos o clases de objeto. Se relacionan los requerimientos del sistema con entidades.

En este proyecto, la misma incluye diagramas de clase y arquitectura general.

* **Vista Física:** Es la que representa los componentes físicos del sistema, así como la conexión (física también) que hay entre ellos. Expone el hardware del sistema y cómo los componentes de software se distribuyen a través de los procesadores en el sistema.

Esta vista se representa mediante diagramas de despliegue, pero para nuestro sistema no son útiles, ya que no está distribuido, sino que todos sus componentes están contenidos en la aplicación.

* **Vista de Despliegue o Desarrollo:** Esta es a la que deberán recurrir los programadores y administradores. Muestra cómo el software está descompuesto para su desarrollo, es decir cada uno de los elementos que se implementen mediante un solo desarrollador o equipo de desarrollo.

Para este proyecto se incluirá un diagrama de paquetes.

* **Vista de Procesos:** Aquí se representan los procesos que hay en el sistema y la forma que tienen los mismos para comunicarse. Es útil para hacer juicios acerca de las características no funcionales del sistema, como el rendimiento y la disponibilidad.

Para describir esta vista se utilizarán diagramas de actividades.

* **Vista de Escenarios:** Es la que relaciona las vistas anteriores mediante casos de uso con lo que tendremos una trazabilidad de componentes, clases, equipos, paquetes, etc., para realizar cada caso de uso.

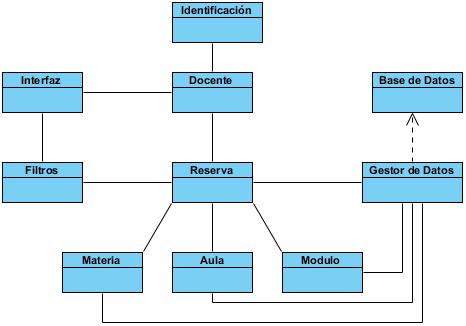
En este documento se puede ver como diagramas de casos de uso.

## Objetivos y limitaciones.

* **Servidor:** Todo nuestro Sistema estará alojado dentro de la aplicación, es decir, la base de datos y el “servidor” que con el que se comunica serán locales, de esta manera no se requerirán protocolos HTTPS o TCP/IP.
* **Cliente:** Debido a esta limitación, no es necesario que los usuarios dispongan de conexión a internet para acceder a los datos.
* **Seguridad:** Se provee un sistema de identificación para otorgar privilegios a ciertos usuarios.
* **Almacenamiento de datos:** Se utilizará una base de datos local.
* **Rendimiento:** Se prevé que el sistema responda a cualquier solicitud de base de datos con un tiempo de espera muy pequeño (menor a 10 segundos), ya que no depende de la conexión a internet. Puede depender del hardware donde se esté ejecutando.

## Vista Lógica.

Como se mencionó anteriormente, para describir esta vista se utilizará un diagrama de clases. El diagrama de clases es un tipo de diagrama de estructura estática que describe la estructura de un sistema mostrando las clases del sistema, sus atributos, operaciones (o métodos) y las relaciones entre objetos.



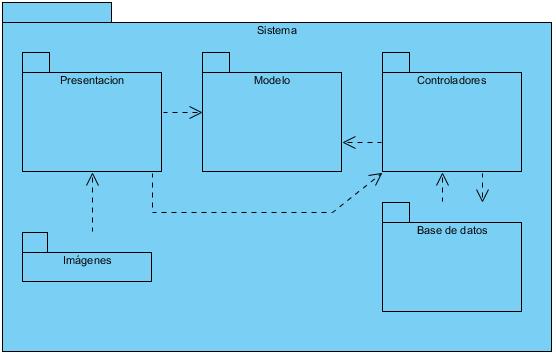
Nuestro sistema se basará en el patrón de diseño MVC (Model View Controller) que se desarrollará en una sección posterior, donde también se incluye un diagrama para describir la arquitectura general del Sistema.

## Vista Física.

Como se mencionó anteriormente, el sistema contará con una base de datos local, por lo tanto no necesitará de un servidor externo para obtener los datos. Todos los componentes del sistema se encuentran contenidos en el software que se instalará en el dispositivo, no se encuentra distribuido. Por lo tanto no se hará uso de diagramas de despliegue para mostrar esto.

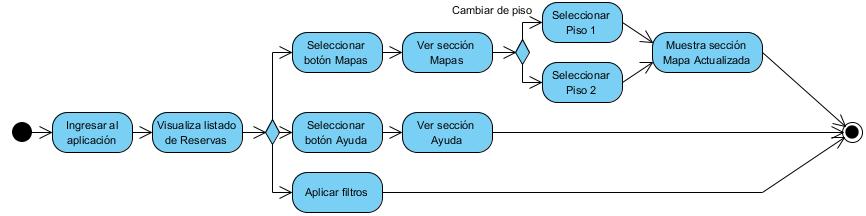
## Vista de Desarrollo.

Para describir esta vista se utilizará un diagrama de paquetes, el cual representa las dependencias entre los paquetes que componen un modelo. Es decir, muestra cómo un sistema está dividido en agrupaciones lógicas y las dependencias entre esas agrupaciones. Dado que normalmente un paquete está pensado como un directorio, los diagramas de paquetes suministran una descomposición de la jerarquía lógica de un sistema. Están normalmente organizados para maximizar la coherencia interna dentro de cada paquete y minimizar el acoplamiento externo entre los demás paquetes.

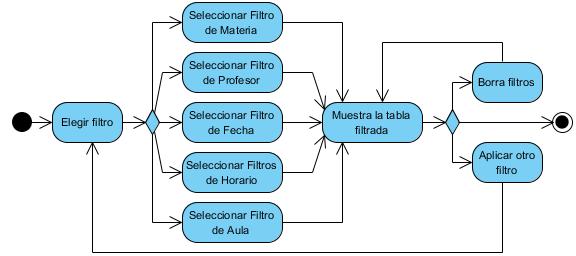


## Vista de Procesos.

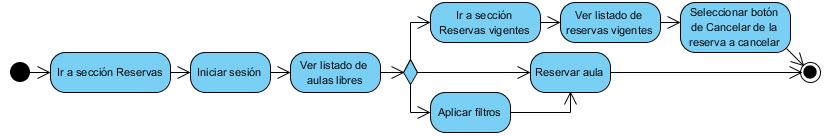
En esta vista se describirán los flujos de trabajo paso a paso del sistema con los siguientes diagramas de actividades. Vimos que en este sistema interactúan 2 tipos de actores, por lo tanto se comenzará desarrollando para un usuario común, tanto docente como no docente:



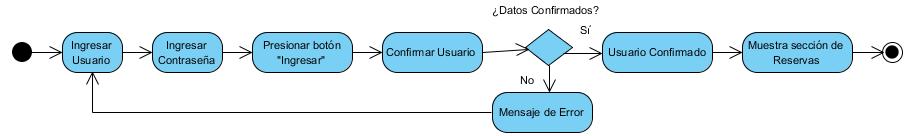
* Especificación del proceso “Aplicar filtros”:



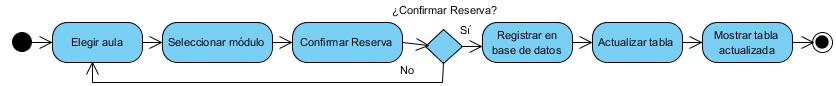
Los siguientes diagramas de actividades muestran los flujos de trabajo de las características propias que tiene un usuario docente, como es el hecho de reservar aulas.



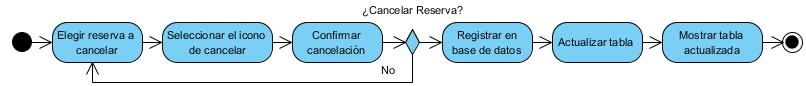
* Especificación del proceso “Iniciar sesión”:



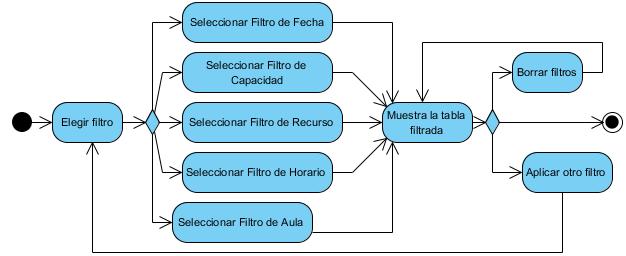
* Especificación del proceso “Reservar aula”:



* Especificación del proceso “Cancelar reserva”:

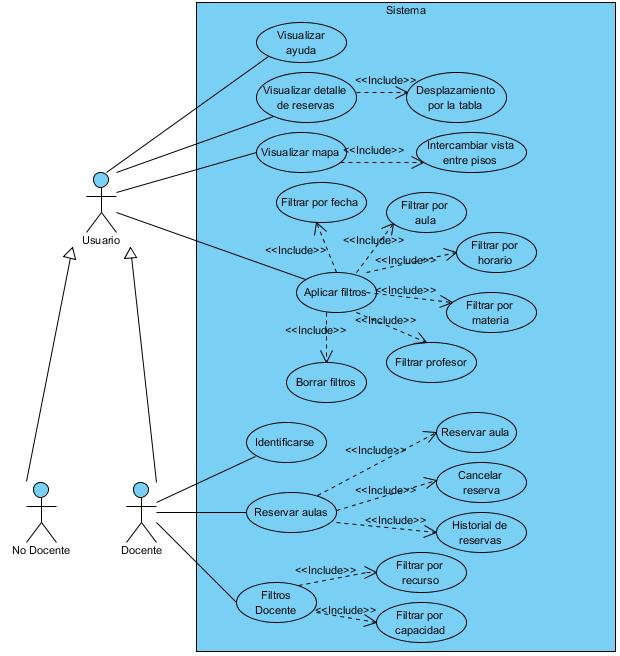


* Especificación del proceso “Aplicar filtros”:



## Vista de Escenarios.

La vista de escenarios consolida las vistas anteriores, donde los estos se convierten en una abstracción de los requerimientos más importantes. Se va a representar la a través de un diagrama de casos de uso:



**Actores:** Existen 2 tipos de usuarios los cuales tienen funciones en común, pero uno de los 2 tiene funcionalidades privilegiadas.

* Todos los usuarios: Se incluyen tanto docentes como no docentes, pueden realizar consultas de reservas, mapas y ayuda a través de la aplicación.
* Docentes: Aparte de las funciones anteriores, tienen otras privilegiadas, poder identificarse, realizar o cancelar reservas de aulas.

Patrón arquitectónico.

Debido a los requerimientos de los que se habló en la introducción a esta sección, fue que se optó por elegir un patrón de diseño conocido como MVC (Model View Controller).

Este patrón establece una separación entre la lógica de negocio (modelo) y la lógica de presentación (vista). El controlador es entonces responsable de reunir el modelo y la vista para proporcionar la aplicación.

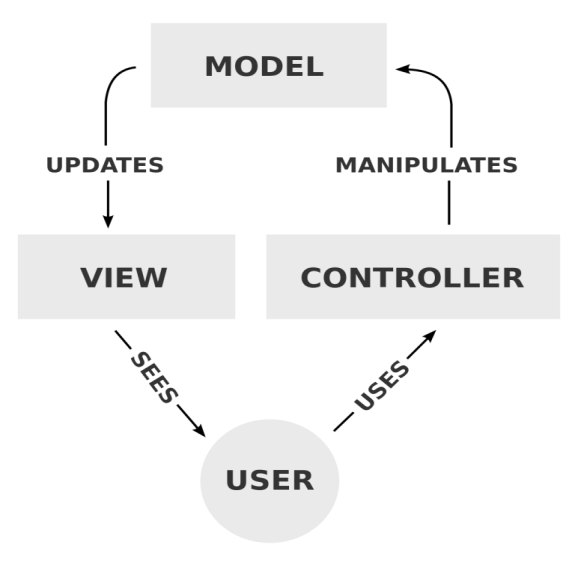
El patrón MVC asegura la separación entre las diferentes capas. Al separar el componente de vista, su desarrollo se puede asignar a un equipo de diseño gráfico, lo que requiere un conjunto de habilidades diferentes al desarrollo del modelo o controlador. El trabajo del equipo de diseño se puede integrar fácilmente de nuevo en la aplicación con el mínimo esfuerzo. El modelo también está totalmente separado, lo que le permite ser probado independientemente de la interfaz de usuario. Finalmente, la lógica del controlador se construye para reunir el modelo y la vista, ensamblando el producto completo en una aplicación Android utilizable.

Diagrama básico del patrón MVC

## Beneficios en relación a los Requerimientos No Funcionales.

* **Mantenimiento:** La división del sistema en 3 capas principales permite al desarrollador implementar cambios en cualquiera de estos tres componentes se ve drásticamente simplificado, siempre que se respeten las interfaces entre los tres. Los usuarios pueden preferir distintas opciones de representación, o requerir soporte para nuevos dispositivos como teléfonos celulares. Dado que el modelo no depende de las vistas, agregar nuevas opciones de presentación generalmente no afecta al modelo.
* **Flexibilidad:** Permite agregar nuevos componentes al sistema sin romper las características existentes. Es decir, dado que la vista se halla separada del modelo y no hay dependencia directa del modelo con respecto a la vista, la interfaz de usuario puede mostrar múltiples vistas de los mismos datos simultáneamente.
* **Facilidad de uso e Integración:** MVC ofrece una estructura de integración, que debido a la segmentación, permite que ciertas partes del sistema existan en áreas especializadas éste. Esto facilita la implementación, ya que las funciones principales se asignarán a una sección y permanecerán solamente en ésta.

## Vista.

La Vista es la información que se presenta al usuario. La vista gestiona la presentación de la información de la aplicación; todo lo relativo a la interfaz de usuario, los datos necesarios para que el usuario pueda seguir interactuando con la aplicación. La vista representa la lógica de presentación de la aplicación. Los componentes de la vista extraen el estado actual del sistema del modelo y proporcionan la interfaz de usuario para el protocolo que se está usando.

En nuestro sistema vamos a tener distintos tipos de vistas:

* Vista de Inicio.
* Vista Docente.
* Vista Mapa.
* Vista Ayuda.

## Controlador.

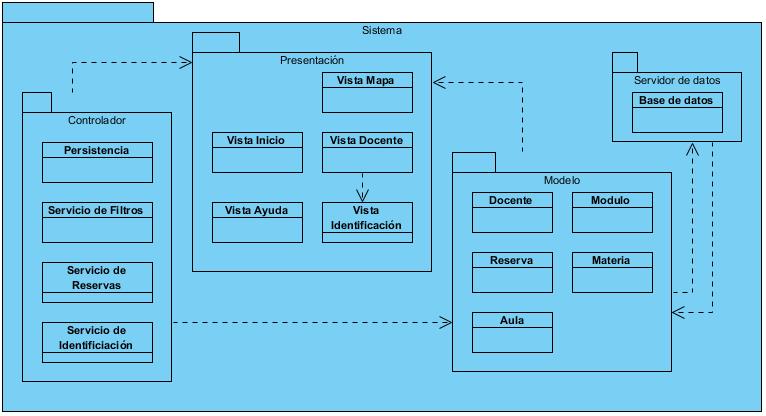
Es el indicado a responder a los eventos invocados desde la vista; y a su vez llamar a los métodos correspondientes el modelo para procesar y producir las respuestas necesarias con los datos adecuados para la interacción de la aplicación. Sirve como un intermediario entre el Modelo, la Vista y cualquier otro recurso necesario para procesar la solicitud. Es responsable de recibir los eventos, determinar el procesador del evento, invocar al procesador y finalmente provocar la generación de la vista apropiada.

Deben realizar las siguientes tareas:

* Control de la seguridad: Evitar que se editen datos incorrectos en la base de datos.
* Identificación de eventos: Notificar a las vistas de cambios en el modelo.
* Preparar el modelo: Realizar los cambios en el modelo.
* Procesar el evento: Responder ante la interacción de un usuario con la interfaz.
* Manejar los errores.
* Provocar la generación de la respuesta: Mostrar los cambios del modelo en la interfaz.

## Modelo.

El Modelo representa las estructuras de datos; es decir toda la información con la que opera la aplicación. Gestionan el comportamiento y los datos del dominio; contienen funciones que ayudan a devolver, insertar y actualizar información de su base de datos, independientemente de la Base de Datos a utilizar. Encapsular el modelo de una aplicación en componentes facilita la depuración, mejora la calidad y favorece la reutilización de código, y los cambios que el sistema pudiera sufrir a futuro.



Pruebas de integración.