Documento de Arquitectura

Benítez Jeremías, Rao Maximiliano, Monsierra Lucas.

Índice.

1. Introducción.
   1. Propósito.
   2. Alcance.
   3. Definiciones y Abreviaturas.
   4. Referencias.
2. Arquitectura del Sistema.
   1. Vista.
   2. Patróndel sistema.
      1. Vista.
      2. Controlador.
      3. Modelo.
3. Vista Física.
4. Vista Lógica.
5. Vista de Despliegue.
6. Vista de Procesos.
7. Vista de Escenarios.

Historial de revisiones.

|  |  |  |  |
| --- | --- | --- | --- |
| Versión | Fecha | Observaciones | Autor |
| 1.0.0 | 26/05/17 | Creación del documento. | Monsierra, Lucas. |
| 1.0.1 | 27/05/17 | Primer 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 |

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

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):

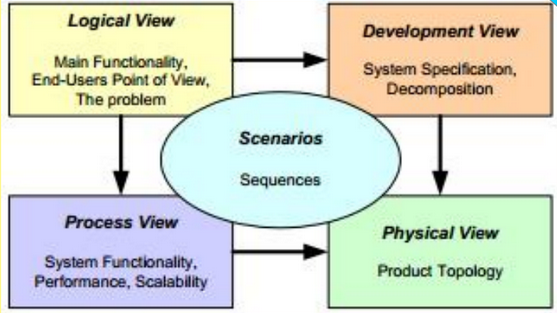


Diagrama básico de la vista 4+1

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, arquitectura general y comunicación.

* **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 incluirán diagramas de paquetes y diagramas de componentes.

* **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 y diagramas de secuencia.

* **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.

Nuestro sistema se basará en el patrón de diseño MVC (Model View Controller) que se desarrollará en una sección posterior.

## Vista Física.

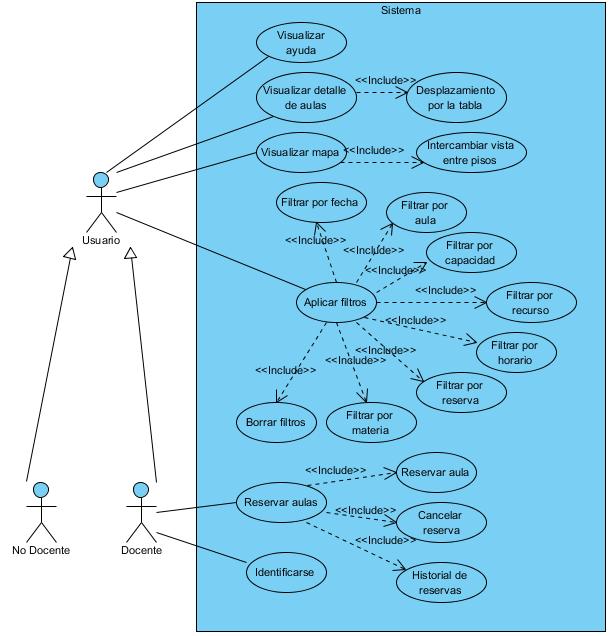
Como se mencionó anteriormente, el sistema contará con una base de datos local, por lo tanto no necesitará de un servidor externo para obtener los datos. Todos los componentes del sistema se encuentran contenidos en el software que se instalará en el dispositivo, no se encuentra distribuido. Por lo tanto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

## Vista de Desarrollo.

## Vista de Procesos.

## 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:



## 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).

La consecuencia de esta elección es que las vistas (en este caso las interfaces gráficas que serán brindadas al usuario), los controladores (que serán las porciones de código que se encarguen de manejar el comportamiento del sistema, sirviendo como intermediario si se quiere entre las vistas y el modelo) y el modelo (la abstracción del problema en sí, en este caso las bases de datos y las operaciones que se puedan realizar sobre ellas) estén lo más separadas posibles.

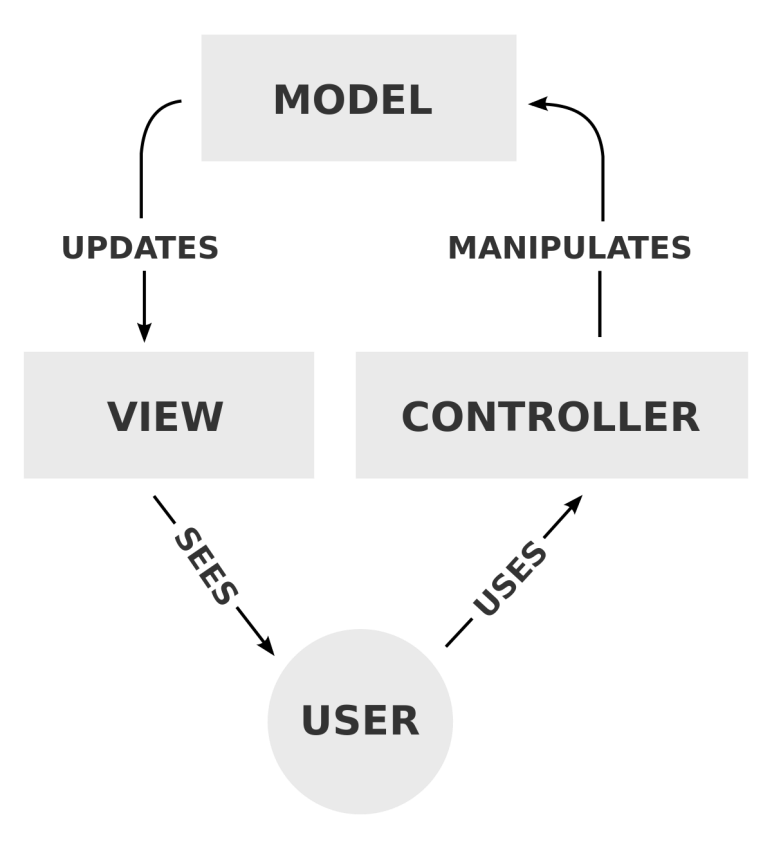
Así, el proceso de implementar cambios en cualquiera de estos tres componentes se ve drásticamente simplificado, siempre que se respeten las interfaces entre los tres.

Diagrama básico del patrón MVC

2.2.1 Vista

2.2.2 Controlador

2.2.3 Modelo