**Arquitectura TIMELESS**



**Luis Fernando Lee Rodríguez**

**Juan David Castillo Laverde**

**Juan José Gómez Arenas**

**Decanatura Facultad de ingeniería**

**Carrera de Ingeniería de Sistemas**

**FUNDAMENTOS DE INGENIERIA DE SOFTWARE**

**PONTIFICIA UNIVERSIDAD JAVERIANA**

**BOGOTÁ D.C**

**2023**

**Objetivos de arquitectura**

A continuación, se presentan los objetivos de arquitectura para TIMELESS. Estos objetivos se han definido para guiar el proceso de diseño y desarrollo del software, asegurando que se logren los resultados esperados. Los objetivos de arquitectura se pueden evidenciar en los requisitos no funcionales, buscando la seguridad, escalabilidad y rendimiento del sistema entre otras cosas.

De esta manera, para determinar qué objetivos se podían realizar de manera más rápida, se realizó una comparación entre los requisitos no funcionales y los casos de uso para encontrar los casos de uso arquitectónicos, es decir los cuales se desarrollarán en primer lugar y se priorizarán. Esto se puede evidenciar en la Fig. 1, para verificar al CU o R-NF especifico, referirse al documento de visión general.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Caso de uso Requisito | CU-001 | CU-002 | CU-003 | CU-004 | CU-005 | CU-006 | CU-007 | CU-008 |
| R-NF-001 |  | X |  |  | X |  |  |  |
| R-NF-002 | X | X |  |  | X |  |  |  |
| R-NF-003 | X |  | X |  |  | X |  | X |
| R-NF-004 | X | X | X | X | X | X | X | X |
| R-NF-005 | X | X | X | X | X | X | X | X |
| R-NF-006 |  | X |  |  | X |  |  |  |
| R-NF-007 |  |  |  | X |  |  | X |  |
| R-NF-008 |  | X |  |  | X |  |  |  |
|  | 4 | 6 | 3 | 3 | 6 | 3 | 3 | 3 |

*Fig. 1. Matriz Caso de uso/Requisito no funcional*

Como se puede evidenciar, dentro de la matriz se obtuvieron dos casos de uso arquitectónicos principales, (CU-002) y (CU-005). No obstante, estos no cumplen con cada uno de los requisitos, por lo cual, para asegurar que se cumplan se tomó la decisión de desarrollar el (CU-001), el cual cumple con el (R-NF-003) y posteriormente el (CU-004) asegurando que cada uno de los requisitos se desarrollen dentro del SW.

**Vista Lógica**

En esta sección del documento, nos enfocaremos en la vista lógica de nuestro software, donde describiremos los elementos funcionales que componen nuestro sistema, sus interacciones y relaciones, así como las responsabilidades y funciones de cada uno de ellos. Además, explicaremos cómo se lleva a cabo el procesamiento de datos y la lógica subyacente que rige el comportamiento del sistema.

En primer lugar, de manera que se entiendan las interacciones y conexiones entre algunos componentes del sistema, se realizó el siguiente diagrama MVC.

Gráfico, Gráfico de cajas y bigotes

Descripción generada automáticamente

*Fig. 2. Diagrama MVC con componentes del sistema*

En este diagrama se evidencian tres tipos de componentes, de Modelo, vista y controlador. Dentro de la vista hay diversas pantallas con las que interactúa el usuario. Estas se utilizan dependiendo de la acción a realizar, ya sea de inicio sesión, registro, añadir actividad o ver las actividades. Posteriormente, estas se conectan con el tipo controlador. Cada una de las pantallas requiere de campos para validar información, por lo cual los controladores se conectan a la mayoría de las vistas, al tener que garantizar una conexión a la base de datos, hacer consultas para acceder a la información requerida y cada pantalla tiene su respectivo controlador. Dentro del modelo se almacena la información del usuario de la aplicación, ya sea de usuario, de sus actividades o de la información de estas. Para esto, se utiliza la base de datos, la cual permite un fácil control del acceso a cada uno de los registros.

Con respecto a la secuencia de interacción entre los diversos casos de usos, se encuentran los siguientes:

Diagrama

Descripción generada automáticamente

*Fig. 3. Diagrama de secuencia: Registrar*

En este diagrama se evidencia como el usuario inicialmente solicita registrarse, por lo cual lo primero que se hace es cambiar a la respectiva pantalla en donde se realiza el registro, para posteriormente establecer la conexión a la base de datos.

Posteriormente se realiza una inserción de los datos del usuario, ya habiendo validado que estos fueran validos para la base de datos y desplegar un mensaje de registro exitoso.

Diagrama

Descripción generada automáticamente con confianza media

*Fig. 4. Diagrama de secuencia de agregar evento fijo*

Dentro de la Fig. 4, es posible evidenciar como se añade una actividad a la base de datos, cabe resaltar que el usuario ya esta registrado. Para esto se inicia sesión, estableciendo la conexión a la base de datos. Luego se muestra la pantalla principal y se selecciona la opción de agregar evento fijo, se rellenan los datos y se hace la inserción en la base de datos. Este mismo proceso se realiza en la Fig. 5, donde se agrega una ActividadXEntrega

Gráfico de cajas y bigotes

Descripción generada automáticamente con confianza media

*Fig. 5. Diagrama secuencia ActividadXEntrega*

Este diagrama de secuencia se realiza las mismas acciones que el anterior, no obstante, cambian los parámetros que se envían y además también cambia los modelos, vistas y controladores que realizan la acción.

**Vista de componentes**

Los componentes del software se pueden evidenciar el la Fig. 6.

**Diagrama, Esquemático

Descripción generada automáticamente**

*Fig. 6. Diagrama de componentes y su distribución*

Entre estos componentes se puede ver la base de datos, donde se administra la información, registrar usuario, el cual permite acceder la información cuando se quiere registrar a un usuario, de manera que se procesan y se busca la manera de conectar con la base de datos. El componente de inicio de sesión tiene los valores que se ingresan dentro de los formularios de la interfaz, guardando en variable y conectándose a la base de datos. Cada uno de estos componentes tiene su respectiva conexión a la base de datos para hacer inserciones o para extraer información, lo cual lo hace por medio de MySqli. El único componente con alguna particularidad es el de Fullcalendar, el cual crea un objeto de este tipo, y le provee el servicio al componente de Ver Horario, creando un objeto de este componente.

**Vista de despliegue**

La manera en que se distribuyen cada uno de los componentes, se puede evidenciar en la Fig. 6. En la figura se muestra la distribución de los componentes, entre el pc local y el servidor, donde el servidor es el encargado de gestionar los datos, enviarlos y garantizar su seguridad.

**Vista de procesos**

Con la finalidad de permitir la escalabilidad, actualmente se cuenta con una distribución como fue mencionada anteriormente, donde la base de datos que es la encargada de extraer e insertar información esta localizada en un servidor, por lo cual permite que se tenga una buena capacidad de respuesta y administración. No obstante, en caso de que se presenten situaciones en las que la capacidad de procesamiento del servidor no sea suficiente para manejar todas las solicitudes, se requerirá una solución para escalar horizontalmente el sistema. Esto se puede lograr distribuyendo la base de datos en un número mayor de máquinas, lo que permite que los procesos se distribuyan y se ejecuten en múltiples servidores. De esta manera, se garantiza que el rendimiento se mantenga para cada uno de los procesos del sistema, sin que se vea afectada la capacidad de respuesta, por lo cual garantizaremos que el usuario mantenga su interés dentro de la aplicación.