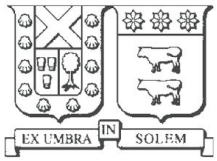
Departamento de Informática
Universidad Técnica Federico Santa María



Requisitos de Software
Proyecto: “Sistema de Ticket para gestión Administrativa”

Integrantes:

Nombres y Apellidos	Email	ROL USM
Juan Pablo León León	juan.leonl.14@sansano.usm.cl	201473047-0
Rodolfo Jaramillo Guzmán	rodolfo.jaramillo.14@sansano.usm.cl	201473024-1
Felix Urtubia Carrasco	felix.urtubia.13@sansano.usm.cl	201303050-5

Índice

Desarrollo Del Prototipo

Se encuentra en la página de github del equipo dentro del directorio untitled, mostrado en la presentación 2 al cliente.

Selección de patrones de diseño.

Intención	Patrón de Diseño	Razonamiento
Se desea mostrar en el Diagrama de Clases la notificación en el Sistema cuando un usuario crea un ticket,se modifica o un operador envía una solicitud para modificar el estado del ticket y avisar al usuario correspondiente	Observer	Seleccionamos este patrón de diseño ya que diferentes usuarios deben recibir notificaciones de distintas acciones o registros, siendo más fácil controlar estos así. También ofreciendo flexibilidad al usuario por si no desea recibir notificaciones de cierto evento.
En el diagrama de clases se desea mostrar la interacción modelo-vista-controlador a través de paquetes separados, que permiten el entendimiento y un desarrollo más ordenado, permitiendo escalabilidad. También se desean mostrar los cambios en el modelo hechos por un usuario a todos los demás, cambiando la vista.	Modelo - Vista - Controlador	Se utiliza este patrón de diseño para separar lo que es datos de la vista, siendo ambos conectados por el controlador, esto se hace para facilitar el desarrollo y el mantenimiento del software
	Template View	Se desea hacer más entendible el código del sistema, facilitando el desarrollo y también reutilizar el código que se pueda, para así reducir tiempos de desarrollo y aumentar la cohesión del sistema.

Creación del diagrama de clases

El diagrama de clases se encuentra en la página de github del grupo, el patrón MVC se encuentra separado en packages o paquetes dentro del modelo y el patrón observer dentro de un paquete, el cual se encuentra a su vez dentro del modelo.

Diagramas de Secuencia

Se encuentran en github del grupo.

Análisis de Trade-off

Funcionalidad:

Mostrar estadísticas generales en el dashboard de cada usuario según su perfil, mostrar estadísticas de avance en operador, de tickets y estado de operadores para supervisor, de tickets generales para jefe y de usuarios y sistema para administrador.

1. ¿Qué cambios se deben realizar en el sistema para agregar una confirmación al realizar acciones relevantes?

2. Opciones:

- O1: Agregar vistas para cada perfil de usuario y una función dentro de ellas.
- O2: Agregar un controlador más para poder generar todas estadísticas y decidir qué y a quién mostrar de acuerdo al usuario que ha iniciado sesión dentro de una misma vista.

- O3: Generar templates en html que vayan cambiando de acuerdo del usuario.

3. Criterio:

- Disponibilidad
- Confiabilidad
- Mantenibilidad
- Interoperabilidad
- Escalabilidad
- Rendimiento

Criterio\Opciones	O1	O2	O3
Disponibilidad	-	+	++
Confiabilidad	-	+	+
Mantenibilidad	--	++	+
Interoperabilidad	-	+	-
Escalabilidad	--	++	+
Rendimiento	-	-	-

- La opción 1, compromete demasiado todos los aspectos del sistema a pesar de cumplir con lo que se busca, por lo tanto, no es una opción viable. El mayor problema que se presenta es la mantenibilidad y la escalabilidad por la cantidad de archivos y líneas de código a utilizar, si se llega a cambiar algo que es transversal a todas las vistas, hay que cambiarlo de uno en uno,

arriesgándose a cometer fallas.
- La opción 2, compromete levemente el rendimiento, pero favorece en gran medida a los demás criterios, el gran beneficio vendría a ser la mantenibilidad y escalabilidad de la opción, ya que todos los componentes críticos para generar información se encuentran en el mismo lugar.
- La opción 3, presenta una tendencia favorable hacia la disponibilidad, lo cual el cliente busca, pero no ofrece las mismas condiciones de escalabilidad y mantenimiento que la opción 2.

En conclusión, la opción 2 es la opción seleccionada por el equipo, porque a pesar de que comprometer levemente el rendimiento del sistema, se favorecen bastante los demás aspectos, entre ellos la confiabilidad que es algo que busca el cliente, esta se favorece del patrón M-V-C a implementar en el sistema, generando estadísticas de acuerdo al modelo y cambiarlas junto con este, el otro gran beneficio para los desarrolladores es la mantenibilidad y la escalabilidad, generando un código más limpio, entendible y con capacidad de crecer si es necesario sin llegar a ser difícil de entender y manejar.