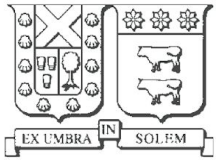
Departamento de Informática
Universidad Técnica Federico Santa María



Informe de Proyecto – INF-225-2018-1-CSJ
Proyecto "Intendo de Software"
2018-08-29

Integrantes:

Nombres y Apellidos	Email	ROL USM
Gustavo Andrés Barrios Araya	gustavo.barrios@sansano.usm.cl	201573510-7
Esteban Emilio Tapia Manzano	esteban.tapiam@sansano.usm.cl	201573511-5
Daniel Esteban Tapia Ramírez	daniel.tapiara@sansano.usm.cl	201573532-8

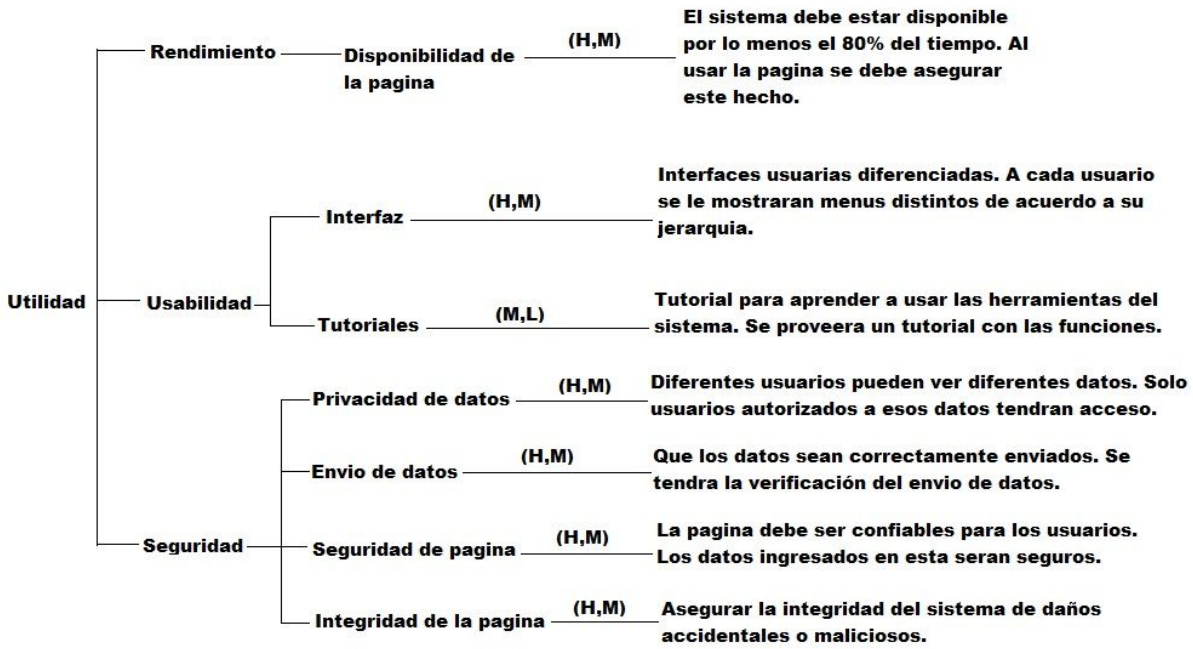
Contenido del Informe a Entregar

1. Requisitos clave (Final)

Req. funcional	**Descripción y medición (máximo 2 líneas)**
Petición de insumos en la obra	Se podrá revisar las peticiones hechas por cada obra, para tener un seguimiento ordenado de las obras.
Visualización de inventario por parte del bodeguero	Se podrán ver los materiales y la cantidad de cada uno para tener en orden la cantidad de materiales restantes.
Envío de insumos a obra o bodega	Se confirmará la recepción por parte de los bodegueros de obra, para tener seguimiento adecuado de estos.
Confirmación de la correcta entrega de materiales	Se confirmará si los materiales recepcionados son los correctos, para no contener errores en cuanto a los materiales en obra.
Implementación del sistema LAUDUS	Se podrá utilizar el sistema de forma más simple, se creará una ventana más simple para su utilización.
Envió de notificación por falta de materiales	Se podrá notificar al encargado de compra de la falta de ciertos materiales para su posterior compra.
Notificación de atraso de materiales	Se informará al momento de tener poco stock en algún material o plazo de entrega atrasado. Se abrirá una notificación a los usuarios con más cargo.
Cotización de los materiales a solicitar	Se muestran las cotizaciones de los proveedores para tomar el mejor de estos, tanto en calidad como precio.

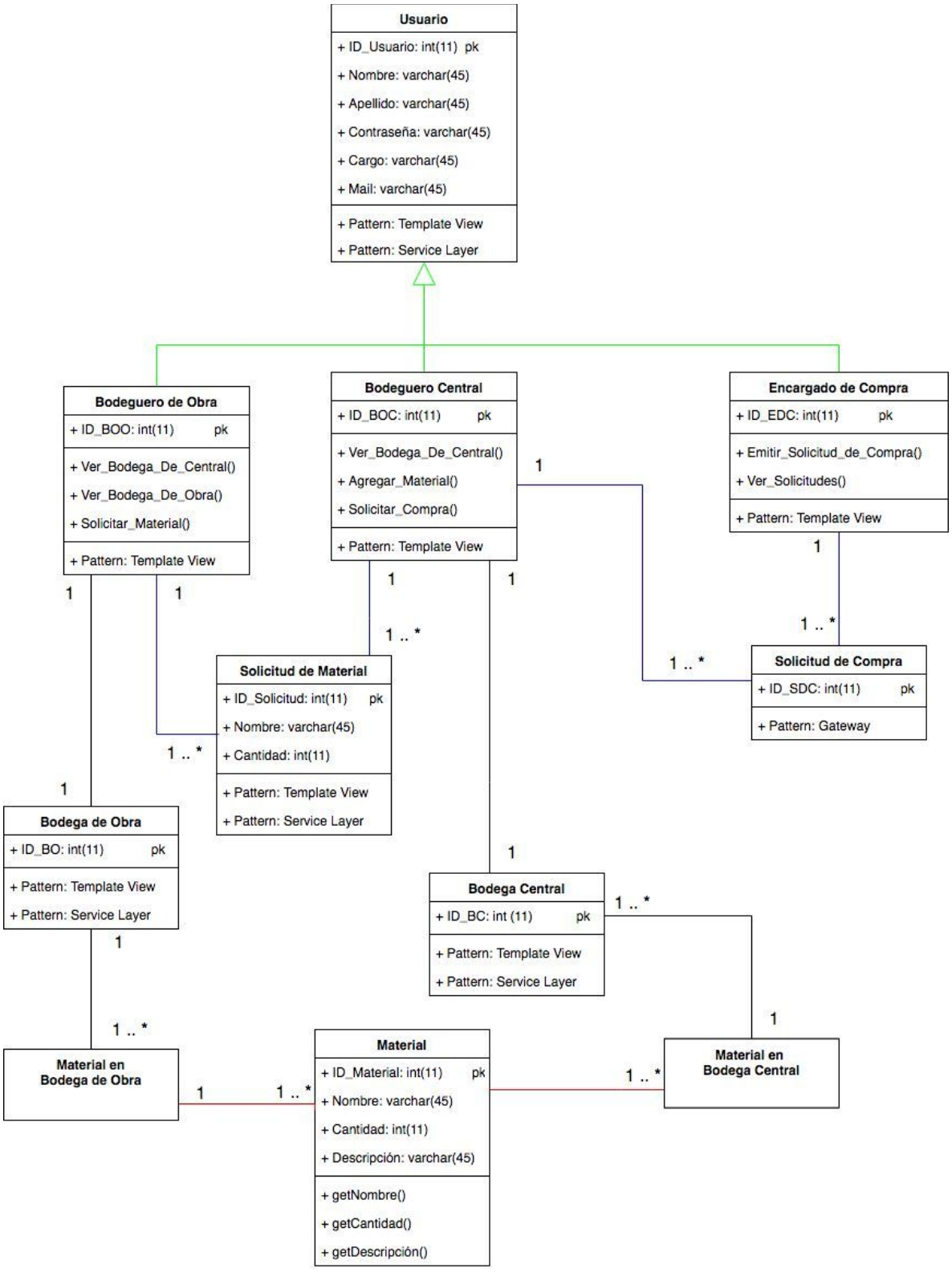
Los requisitos siguen siendo los mismos que los anteriores entregables, no se encuentra necesario agregar o quitar más de los que ya hay, ya que se considera que dentro de estos se encuentran los necesarios.

2. Árbol de Utilidad (Final)



Dentro del árbol de utilidad se detalló más sobre las principales formas de medir cada objetivo a lograr.

3. Modelo de Software (Final)



Donde el color verde es View, el rojo es Model y el azul es de Controller.

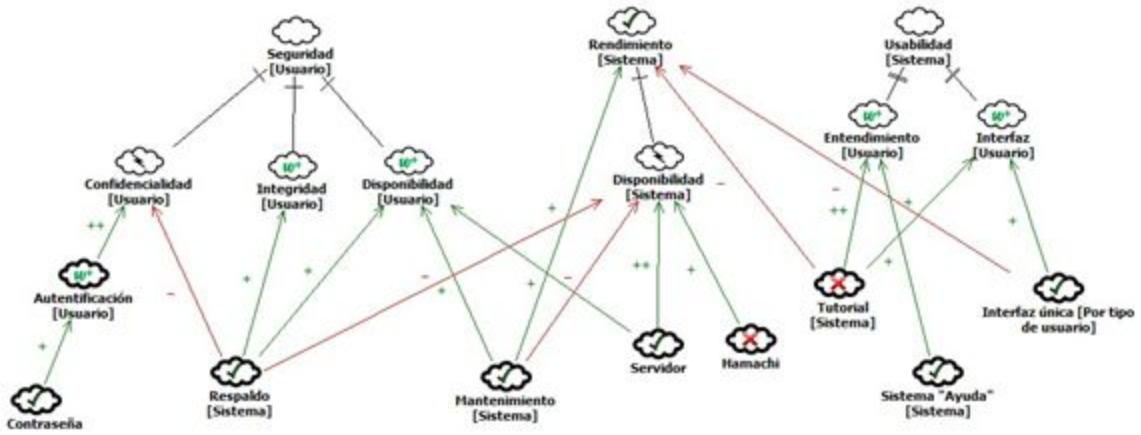
El modelo de software que se propone construir por nuestro equipo es un sistema para automatizar ciertos aspectos realizados dentro de la empresa que son tediosos y difíciles de realizar por humanos pero que con la ayuda de nuestro sistema ahorrará tiempo y trabajo. De esta manera el sistema ayudará a cumplir los requisitos propuestos anteriormente. La forma más eficiente de hacerlo es mediante un modelo iterativo incremental donde se testea cada cierto tiempo y se procede a agregar nuevas cosas mediante feedback con usuarios.

Intención	**Patrón de Diseño**	**Razonamiento**
Ya que angular es un framework que se basa en el modelo MVC, este tiene que ir como uno de los patrones de diseño.	Model-View-Controller (MVC)	El patrón MVC es la cabeza del stack que se utiliza para este proyecto, además que facilita el trabajo por ser un patrón muy utilizado, entonces investigar
Ya que no se usa ORM, y las sentencias SQL con SELECT, INSERT y UPDATE a lo mucho, se cree que es el más adecuado.	Active Record	Por el hecho de no utilizar sentencias SQL demasiado complejas, se nos facilita más este patrón ya que cumple con hacer querys sencillas.
Mantener una sesión de usuario constante en el tiempo o por lo menos hasta que desloguee	Client/Server/Database Session State	Ya que se tienen un montón de usuarios con distintos cargos, se necesita una forma de separar las vistas, funciones e incluso ciertos permisos.
Al estar utilizando el framework de angular 2, este tiene implementado este patrón por defecto.	Page Controller	Al tener el framework de Angular y este tenerlo implementado, sería poco apropiado no usarlo, de hecho se ha facilitado el trabajo con este.

Poder manejar operaciones en una sola ubicación mayoritariamente, el framework angular 2 nos facilita esto.	Service Layer	Es casi la misma idea de Page Controller, ya que de por si están implementados es cosa de usarlos, además hace que las funciones no queden repartidas por todo el proyecto, sino que quedan en una parte accesibles por todos.
Poder alterar vistas sin tener muchas modificaciones, ni html's para cada usuario o página.	Template View	Como se dijo, al tener muchos usuarios se puede dar el caso de tener que crear muchas html's para cubrirlos a todos, pero con este patrón solo nos enfocamos en crear una plantilla estándar y solo poner datos encima correspondiente a cada usuario.
Facilitar el manejo de la API de LAUDUS para el entendimiento mejor del grupo desarrollador	Gateway	Próximamente cuando se implemente la API de LAUDUS, se necesitará una serie de funciones entendibles por los desarrolladores para poder facilitar la comunicación del programa con LAUDUS.

Para este entregable se agregaron los patrones más a simple vista, donde se trata de destacar la mayoría los patrones dentro del modelo de software.

4. Trade-offs entre tecnologías (final)



Decisión	Softgoal	Evaluación	Razonamiento
Contraseña	Autentificación	+	Una contraseña establece una forma de realizar una autenticación de forma efectiva. Simple en comparación con otras formas de autentificación.
Autentificación	Confidencialidad	++	Una autentificación permite diferenciar a los distintas personas que ingresan al sistema. Autorizando al usuario a acceder al sistema si cumple con el proceso establecido.
Respaldo	Integridad	+	Un respaldo a los datos del sistema otorga una acción a realizar para recuperar el estado del sistema frente a una eventual complicación con el servidor.

Respaldo	Disponibilidad para el usuario	+	Un respaldo del sistema también facilita su disponibilidad frente a los usuarios debido a que al permitir una mayor capacidad de reacción frente a un problema. Aumenta su facultad para seguir manteniéndose en línea.
Respaldo	Confidencialidad	-	Tener varios copias de los datos del sistema genera que exista una mayor oportunidad para obtener la información de los usuarios.
Respaldo	Disponibilidad del sistema	-	El respaldo por su parte también afecta a la disponibilidad del sistema debido al tiempo que se necesita para realizarlo. Tiempo en el cual el sistema no se encontrará disponible.
Servidor	Disponibilidad	++	Teniendo un servidor donde se pueda almacenar datos y el software, se podría evitar pérdidas locales.
Servidor	Disponibilidad	+	Como se tiene un servidor para la empresa, los datos de este pueden ser accesados por el usuario a cargo de la vista de estos datos.
Hamachi	Disponibilidad	+	Utilizando la aplicación LogMeIn Hamachi se puede crear un servicio de red privada permitiendo a los usuarios acceder a un computador donde se levantará la página web. Sin embargo, esta alternativa no es segura y es poco profesional.

Tutorial	Entendimiento	++	Como tener un tutorial es básicamente un recorrido completo al software, ayudaría mucho al usuario a entender este aunque sea nuevo.
Tutorial	Interfaz	+	Al tener una explicación previa al uso del software, facilita el encontrar cierto implementos que tal vez no sean tan obvios de encontrar.
Tutorial	Rendimiento	-	Al necesitar tener que explicar paso a paso cada implementación, se pierde tiempo, se gasta espacio y además afecta en la velocidad de creación del software.
Mantenimiento	Disponibilidad	+	Al tener siempre un backup del sistema, se evitaría la pérdida de ciertos datos que podrían ser eliminados por accidente o alguna catástrofe.
Mantenimiento	Rendimiento	+	En el caso de que haya alguna forma de desconexión, se puede trabajar de manera offline con el backup previamente hecho, de tal manera no perder trabajo.
Mantenimiento	Disponibilidad	-	En el caso de que la mantención sea en periodos muy pequeños (cada dos dias), este podria pasar abajo lo que conlleva a un atraso en las ejecuciones de la empresa.

Sistema "Ayuda"	Entendimiento	+	Al igual que en la mayoría de los programas, en el caso de que no se entienda algo del producto o no se pueda encontrar cierta utilidad, el crear este sistema facilita al usuario en encontrarlo o usarlo.
Interfaz única de usuario	Interfaz	+	Al tener interfaces distintas, se tiene una clara diferencia de que vista pertenece a cada uno, por lo tanto, afectaría menos al grado de error y permisos de cada usuario.
Interfaz única de usuario	Rendimiento	-	Al tener interfaces para cada tipo de usuario, este podría afectar negativamente, ya que se necesita revisar y decidir muchas vistas en el caso de que hayan muchos tipos.

No se encontró que haya problema en el modelo de los trade-offs, por lo que no fue necesario un cambio.

5. Deuda técnica incurrida

Tabla 5: Deuda técnica

Ítem deuda técnica	Razonamiento	Impacto
Poca o nula documentación	Poco tiempo en el orden de la documentación	Llevará al poco entendimiento de las tecnologías utilizadas
Conservación de sesiones modificables	Session state fue fácil de implementar	Sesiones totalmente inseguras, ya que es modificable desde el navegador.
ERP del cual no se tiene conocimiento	El ERP que se pide trabajar no se encuentra disponible para testing.	A estar usando una api externa que no tiene que ver con el ERP, al momento de implementarlo pueden ocurrir errores.

Ausencia de tutorial para el uso de las herramientas del software	Se pensó en la creación de un tutorial para la facilitación del uso de las herramientas del sistema.	Los usuarios no tendrán una visión clara de las herramientas disponibles del sistema.
No verificación de las cantidades a solicitar en materiales	Para disminuir la carga del trabajo optamos por no verificarlo.	Si una cantidad ingresada es mayor a la que se tiene en bodega podría ocurrir un error.