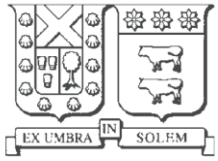
Departamento de Informática
Universidad Técnica Federico Santa María



Informe de Proyecto – INF-225-2018-1 - CSJ
Proyecto “Metalgreyteamon”
03/08/2018

Integrantes:

Nombres y Apellidos	Email	ROL USM
Nicolás Acevedo	nicolas.acevedoy@sansano.usm.cl	201573512-3
Francisco Farías	francisco.fariasm@sansano.usm.cl	201573601-4
André Vigneau	andre.vigneau@sansano.usm.cl	201573572-7

Contenido del Informe:

1. Requisitos clave (Actualizado)

No se agregaron nuevos requisitos funcionales. Sin embargo, se modificó el título o descripción en varias de ellas. Los cambios hechos se especifican a continuación:

- Al requisito "login a la plataforma" se le cambió el nombre a "Posesión de cuenta propia por usuario", ya que se consideró que la acción de un login no refleja el valor correspondiente a la actual descripción de aquel requisito.
- Se cambia "bodeguero central" y "bodega central" por "encargado de bodega central" en todos los requisitos. Esto porque refleja de mejor manera la participación del actor y le da consistencia con lo ya expuesto, por ejemplo, con los casos de uso.

Con respecto a los requisitos no funcionales, se modificaron los ya existentes, y se agregaron nuevos. Los cambios hechos se especifican a continuación:

- Se especificó un tiempo estimado de 5 minutos máximo para el requisito "actualización en tiempo real", ya que es muy poco realista pensar en un sistema de respuesta inmediata en una etapa tan temprana de desarrollo con más énfasis en funcionalidad que en optimización.
- Se especificó quién será el encargado de la aprobación de cuentas de usuario, otorgándole dicha facultad al encargado de adquisiciones, ya que según el criterio del equipo, es quien tiene una mayor jerarquía dentro de los actores, quienes son los futuros usuarios del software.
- Se agregó el requisito "capacitación necesaria para uso de software", que indica la necesidad de una breve capacitación o instructivo de uso del sistema, el cual no tomará más de 1 día.
- Se modificó la descripción del requisito "tiempo de creación de usuarios" para hacerlo más específico.
- Se añadió el requisito "uptime del servidor", ya que es crucial tener una aplicación que esté disponible casi el 100% del tiempo (y ojalá en su totalidad), para así evitar consecuencias negativas, por ejemplo, retrasos en tareas requeridas.
- Se agregó el requisito "escalabilidad del sistema", porque las aplicaciones serán de uso masivo y eso no debe afectar su rendimiento eficiente y óptimo, y, por ende, tener un sistema expedito.
- Se agregó el requisito "restricción de accesos", ya que se pensó en la importancia de que los usuarios de distintos niveles jerárquicos de la organización de la empresa tengan acceso sólo al contenido acorde a esa jerarquía, y no a toda la información (éstas corresponden a políticas de la empresa).
- Se añadió el requisito "información de errores", ya que se considera importante mostrarles a los usuarios detalles de los errores que le aparezcan en las vistas por cualquiera sea el motivo (y hacerles saber dicho motivo). Esto mejorará la experiencia usuaria y ayudará a que el sistema sea más sencillo de usar.

Tabla 1: Requisitos funcionales (actualizados)

Req. funcional	Descripción y medición
Posesión de cuenta propia por usuario	Cada actor (futuro usuario) debe poseer una cuenta para acceder a la aplicación, con su respectiva vista.
Creación de usuarios	Cada persona deberá poder crear su propio usuario, especificando el cargo que tenga.
Envío de solicitud de pedido	El encargado de obra debe poder enviar una solicitud de pedido de materiales al encargado de bodega central.
Estado solicitudes de pedido bodega central	El encargado de bodega central debe poder ver el estado de todas las solicitudes de pedido de material que haya hecho.
Reenvío de solicitud a bodeguero de obra	El encargado de bodega central debe poder reenviar al Enc. de bodega de obra las solicitudes de pedido recibidas.
Estado de solicitudes de pedido recibidas.	El Encarg. de bodega de obra debe poder informar el estado de una solicitud de pedido recibida desde la bodega central.
Reenvío de solicitud a encargado de adq.	El Enc. de bodega central debe poder reenviar al encargado de adquisiciones las solicitudes de pedido recibidas.
Cotizar materiales	El encargado de adquisiciones debe poder cotizar material a través del sistema LAUDUS, el cual le enviará las opciones.
Generar Orden de Compra	El encargado de adquisiciones debe poder enviar una OC al proveedor que corresponda mediante el sistema LAUDUS.
Informar compra a bodega central.	El encargado de adquisiciones debe poder informar al enc. de bodega central que la orden de compra de materiales se efectuó.

Tabla 2: Requisitos extra-funcionales (actualizados)

Req. extra-funcional	Descripción y medición
Actualización en tiempo real	Las actualizaciones en el estado de las solicitudes de pedido deben poder verse reflejadas en <=5min.
Tiempo de creación de usuarios	La creación de cuenta le tomará al usuario <=30s. en promedio.
Aprobación de usuarios	El encargado de adquisiciones deberá aprobar la creación de nuevas cuentas.
Capacitación necesaria para uso de software	Se necesita <=1 día de capacitación para poder utilizar el software correctamente.
Uptime del servidor	El servidor estará disponible 99.9% de las ocasiones en

	que necesite ser ocupado.
Escalabilidad del sistema	El sistema deberá funcionar correctamente y sin caídas con <=10 usuarios simultáneos.
Restricción de accesos	El acceso por parte de un usuario no autorizado a datos no permitidos no debe ocurrir.
Información de errores	Deben existir mensajes o descripciones de los errores que puedan generarse por cualquier motivo.

2. Árbol de Utilidad (Actualizado)

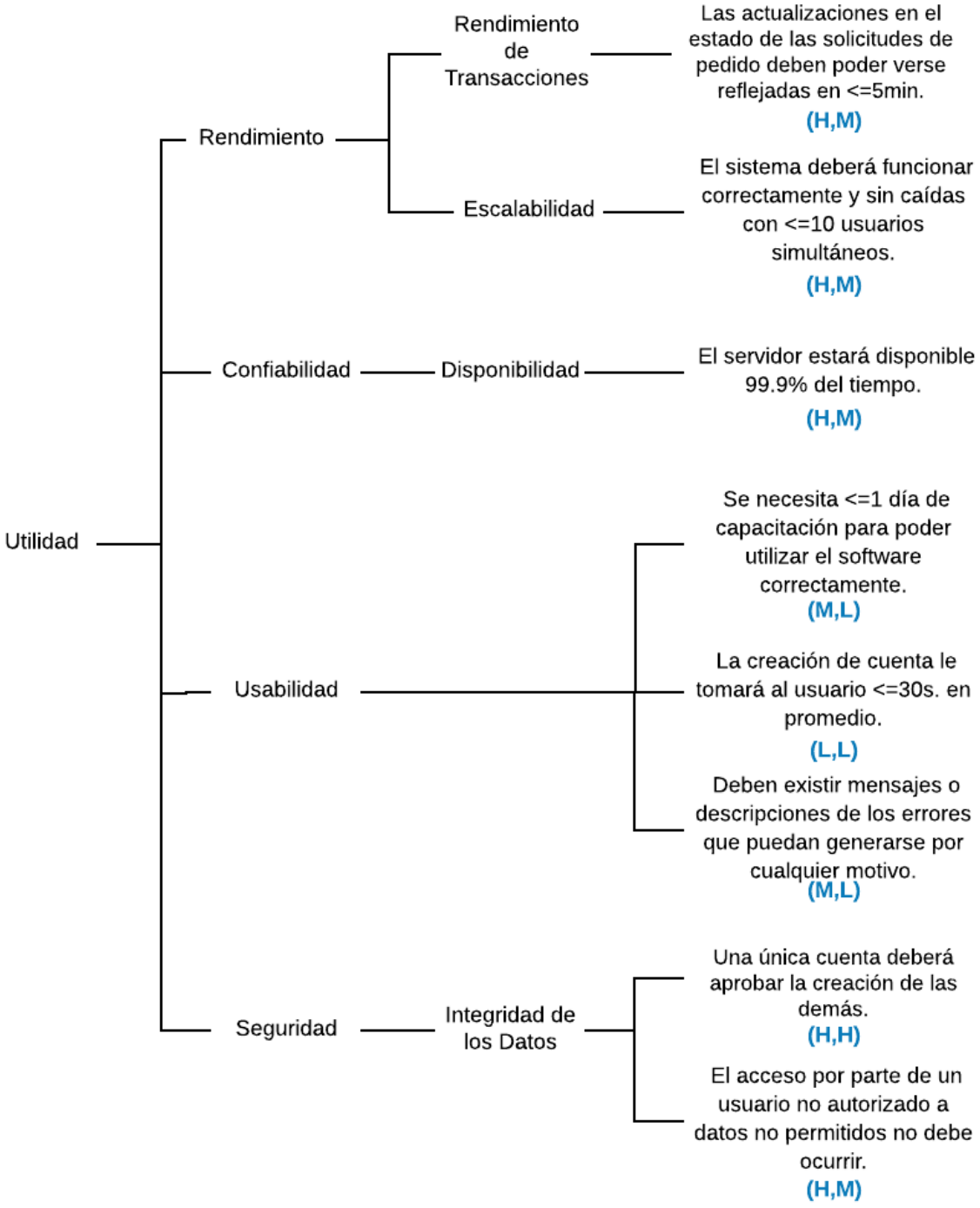


Ilustración 1: Árbol de Utilidad (Actualizado)

3. Modelo de Software

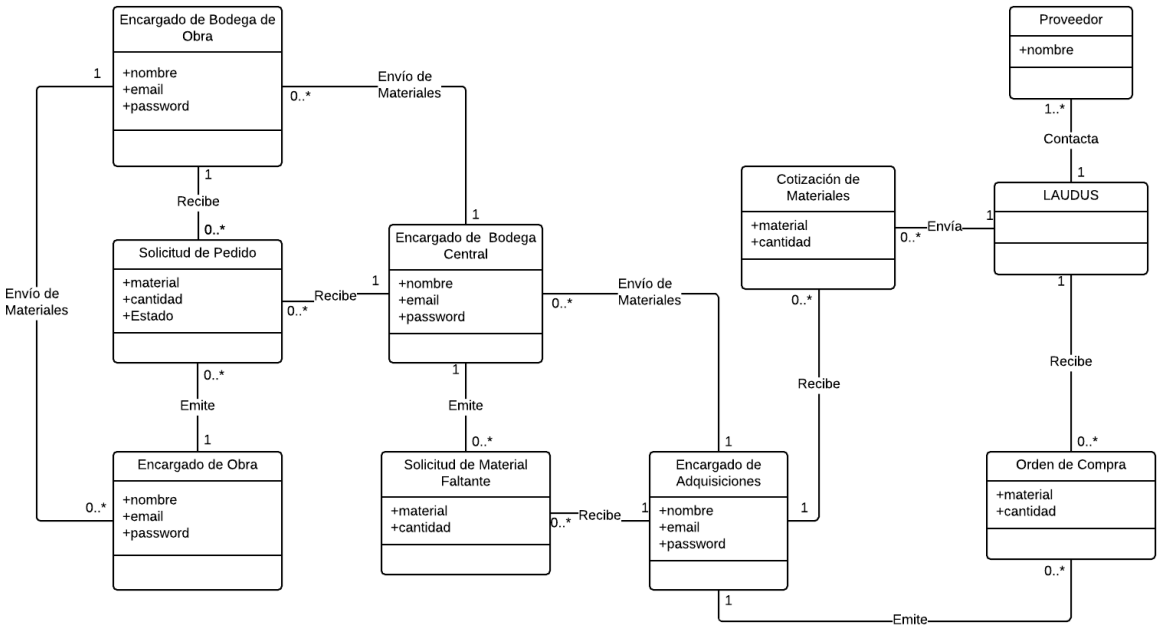


Ilustración 2: Modelo de software

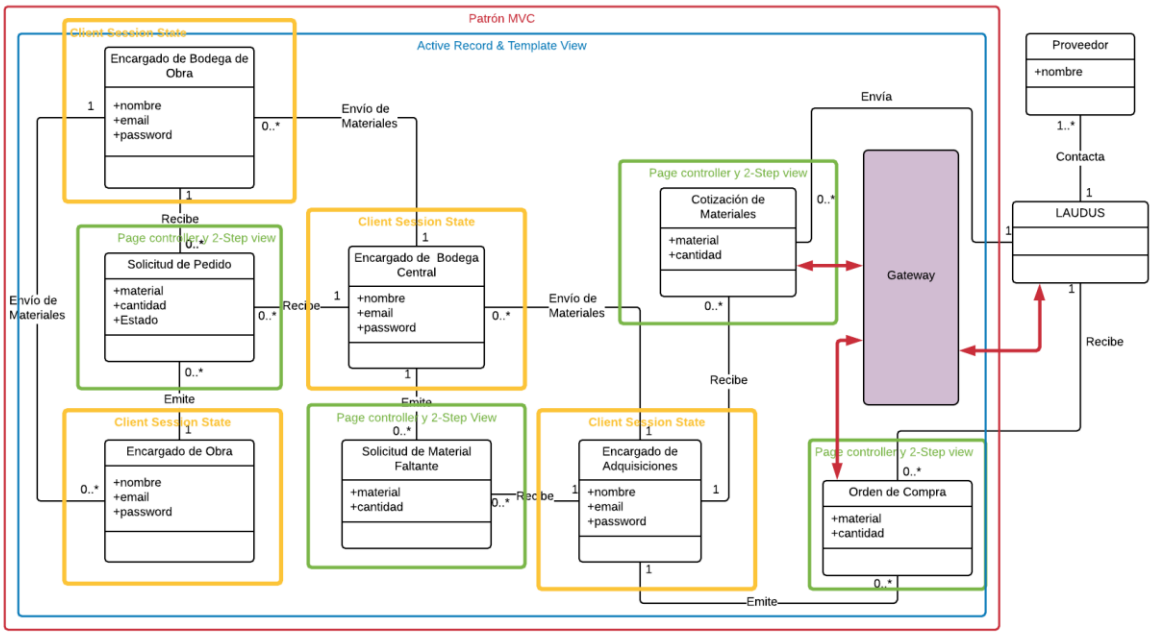


Ilustración 3: Modelo de software indicando patrones de diseño a utilizar

Tabla 3: Selección de Patrones

Intención	**Patrón de Diseño**	**Razonamiento**
Crear una aplicación web con IU amigable y que refleje actualizaciones en los datos dinámicamente.	Model-View-Controller (MVC)	MVC es muy utilizado para desarrollo de WebApps, ya que al separar el modelo (BD y data persistente), el controlador (funcionalidades y parseo de datos) y la vista, se evita tener códigos desordenados donde la vista y el controlador, por ejemplo, se mezclen en un solo archivo. También, se mantiene un orden general en el espacio de trabajo, para evitar modificar funcionalidades por accidente. Este patrón, además, es automáticamente generado por el framework que escogimos, Ruby on Rails.
Administrar, parsear y manipular datos de formulario y peticiones.	Page Controller	Es buena idea utilizar una clase por cada vista para parsear los datos ingresados por el usuario a través de la misma, para así mantener cierta modularidad y evitar confusiones y cruce de datos accidental. Viene implementado con el framework escogido.
Gestión de persistencia de datos de manera sencilla y abstracción de la BD.	Active Record	Para una mejor gestión de datos y menos complicaciones en el código, Active Record es ideal, ya que no se necesita usar sentencias SQL para operaciones en la BD, si no que utiliza métodos para objetos que representan los datos, y que se encargará de validar antes que se hagan persistentes. Simplifica la comunicación con los datos y la lectura/desarrollo del

		código al no necesitar escribir SQL.
Renderización de vistas a partir de datos no-estáticos. Libre modelado de vistas HTML.	Template View	Renderiza datos obtenidos, por ejemplo, de la BD en una vista HTML, y así se disminuye la cantidad de código (utilizando por ejemplo iteraciones), y además permite renderizar datos no estáticos (variables) en una vista de manera sencilla. Gracias al framework, es muy sencillo de implementar.
Diseño de vistas a partir de un modelo base o "plantilla". Flexibilidad de la vista para distintas pantallas.	2-Step View	Se busca una apariencia uniforme para cada vista de la aplicación, por ende, se genera primero un HTML con los datos y formatos básico, y sobre este, mediante el uso de Template View, se renderizan los demás elementos (textos, botones, etc) según la vista que desee mostrarse, para mantener así el estilo y formato y evitar duplicidad de código. Viene por defecto implementado con el framework escogido.
Mantener sesiones iniciadas o recordadas opcionalmente. Disminuir tiempo de requests al no tener que verificar datos una y otra vez. Uso de cookies.	Client Session State	Se almacenarán datos muy pequeños en el ordenador del cliente para que, si así lo desea, pueda mantener iniciada/recordada su sesión en su ordenador. Esto disminuirá los tiempos de respuesta y mejorará el rendimiento del sistema. Gracias al framework escogido, esto es fácilmente lograble mediante el uso de cookies, familiares para los usuarios por su masivo uso en muchos sitios web.
A futuro: generar datos/recibir	Gateway	Pensando en etapas futuras de

datos con formatos compatibles o similares a los usados con LAUDUS.		desarrollo, se podría querer recibir los datos enviados por LAUDUS, o al menos generar OC con un formato similar al pedido por éste. Para ello se buscará implementar una clase Gateway, que transforme los datos enviados a través de la vista de la App, y organizarlos de manera que LAUDUS sea capaz de recibirlos válidamente.

4. Trade-offs entre tecnologías

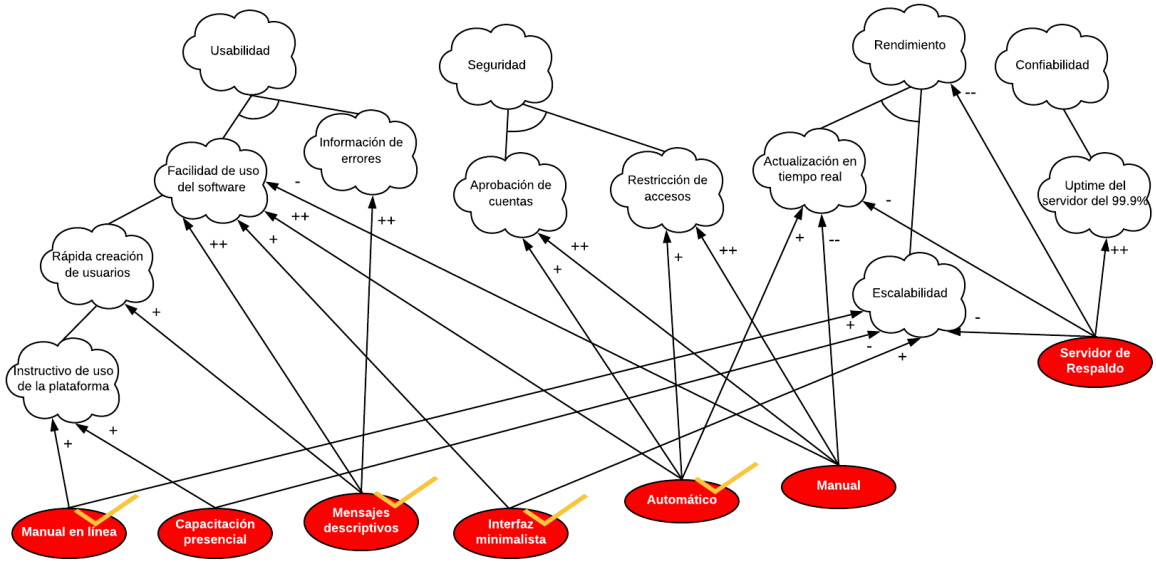


Ilustración 4: Softgoal Interdependency Graph (SIG)

Tabla 4: Trade-offs entre opciones tecnológicas

Decisión	Softgoal	Evaluación	Razonamiento
Manual en línea	Instructivo de uso de la plataforma	+	Explica a los usuarios la forma en que el software debe ser usado, sin embargo, la lectura no asegura una comprensión total de como se debe hacer ese uso.
Manual en línea	Escalabilidad	+	Conforme la aplicación se actualiza, el manual puede ser fácilmente actualizado de la misma manera, lo

			cual facilita el aprendizaje de uso de la plataforma.
Capacitación Presencial	Instructivo de uso de la plataforma	+	Explica a los usuarios acerca de cómo utilizar el software, pero requiere tiempo para todos, lo que significa menos tiempo de producción para la empresa.
Capacitación presencial	Escalabilidad	-	Sería necesario hacer una nueva capacitación cada vez que se sumen nuevos usuarios al sistema.
Mensajes descriptivos	Rápida creación de usuarios	+	Gracias a estos mensajes, se hace más fácil entender cuáles son los datos que se piden.
Mensajes descriptivos	Facilidad de uso del software	++	A causa de estos mensajes la plataforma se vuelve más entendible y fácil de usar.
Mensajes descriptivos	Información de errores	++	El usuario estará al tanto de los errores que él mismo cometió, o de los campos que no llenó. En caso de tener un error un poco más técnico, lo sabrá también, y podrá avisar a su supervisor.
Interfaz minimalista	Facilidad de uso del software	+	Mientras más simple sea la interfaz, menos es el esfuerzo que debe invertir un usuario en entenderla.
Interfaz minimalista	Escalabilidad	+	Al usar interfaces sencillas, se disminuirán los recursos utilizados por usuario, lo que permitirá que el sistema funcione igual de óptimo.
Automático	Facilidad de uso del software	++	Mientras menos tareas se deleguen al usuario, más sencillo es utilizar el software
Automático	Aprobación de cuentas	+	Habrá un controlador encargado de verificar, según datos proporcionados por la empresa, si un usuario está correctamente creado o si los datos no corresponden, por lo que se rechazaría la solicitud. Es más seguro pero menos eficiente.
Automático	Restricción de acceso	+	Reduce el trabajo del usuario y es eficiente pero el software podría

			incurrir en errores.
Automático	Actualización en tiempo real	+	Le da autonomía al software.
Manual	Facilidad de uso del software	-	Al estar en manos de alguien, será negativo para el software tener que esperar la respuesta del encargado, y hará que todo funciona más lento.
Manual	Aprobación de cuentas	++	Al ser manual, ésta debe ser vigilada por un empleado y no una máquina que puede hacer omisiones por patrones.
Manual	Restricción de acceso	++	Al hacer que los accesos sean dados de manera manual, se consigue que todos los permisos sean autorizados por algún nivel de jefatura, lo que reduce al mínimo los errores.
Manual	Actualización en tiempo real	--	El depender de alguien nunca actualizará las cosas en tiempo real, ya que debe esperarse a que dicho encargado apruebe cada nueva solicitud.
Servidor de respaldo	Escalabilidad	-	Al hacer una actualización en el software, esta debe ser replicada en ambos servidores, por lo que implica un costo mayor.
Servidor de respaldo	Uptime del servidor del 99.9%	++	El tener un servidor de respaldo garantiza un uptime mayor, ya que éste funcionará incluso si ocurre un fallo del servidor principal.
Servidor de respaldo	Actualización en tiempo real	-	Cada vez que algo se actualice, se deberá actualizar también en el respaldo, por lo que el sistema funcionará más lento al estar realizando estas operaciones extra.
Servidor de respaldo	Rendimiento	--	El sistema debe estar actualizando constantemente el respaldo, y se mantendrá ocupado en estas operaciones, consumiendo recursos extra.