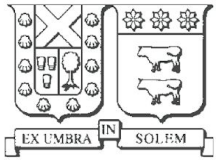
Departamento de Informática
Universidad Técnica Federico Santa María



Informe de Proyecto – INF-225-2018-1-CSJ
Proyecto "AbstractPattern"
2018-Agosto-26

Integrantes:

Nombres y Apellidos	Email	ROL USM
Claudia Hazard Valdés	claudia.hazard.14@sansano.usm.cl	201404523-9
Diego Montecinos Olivares	diego.montecinos.14@sansano.usm.cl	201473601-0
Daniel Pacheco Pacheco	daniel.pacheco.14@sansano.usm.cl	201404570-0

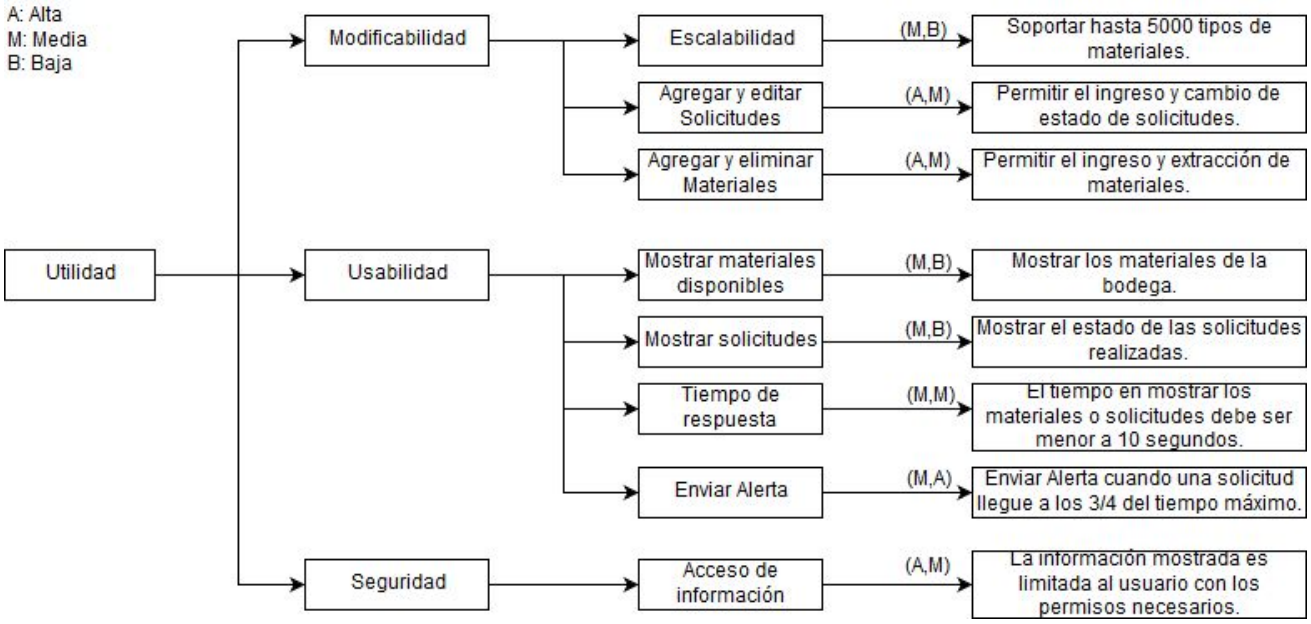
Índice

1. Requisitos clave (Final)

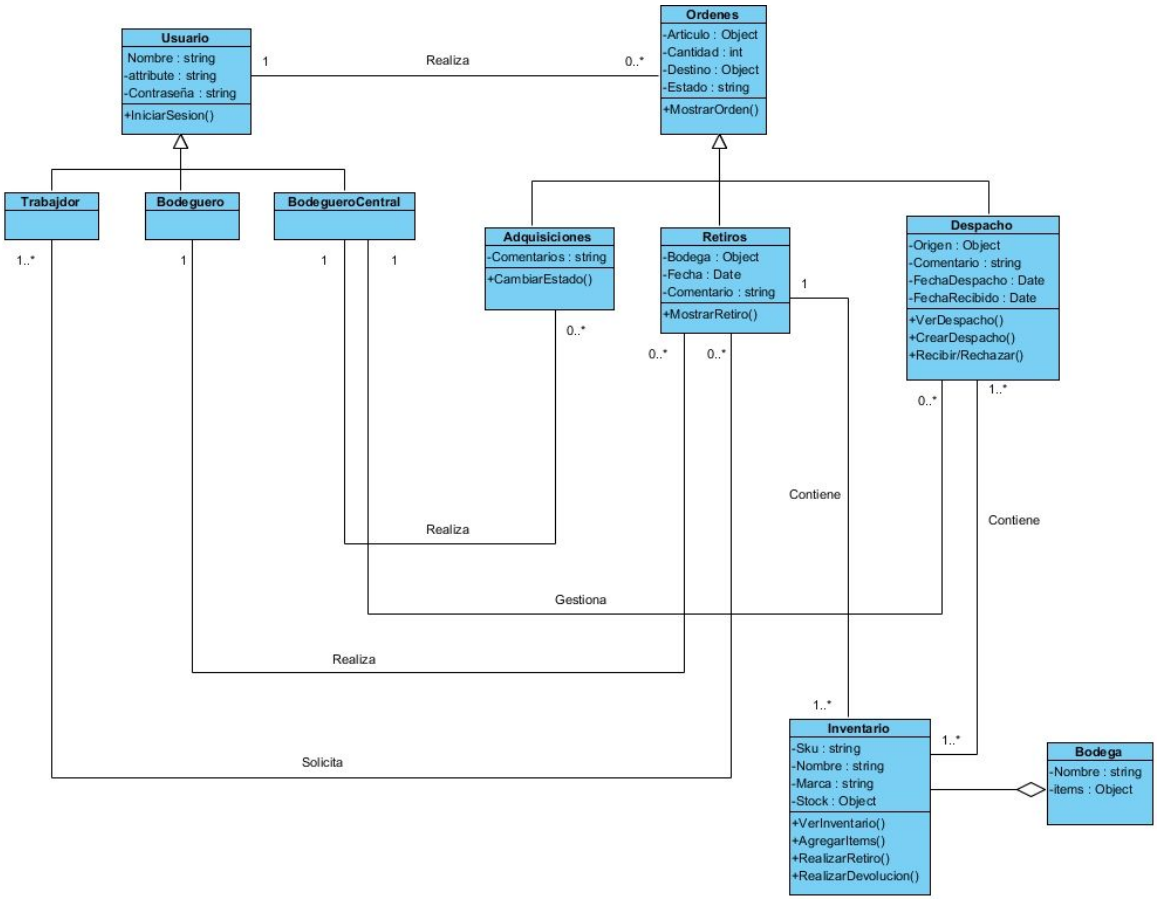
Req. funcional	**Descripción y medición**
Diferentes tipos de usuario	El sistema mostrará distintas características para cada tipo de usuario.
Ingresar solicitud de Materiales	Se podrá ingresar la solicitud de un material quedando esta con un identificador único asociado.
Mostrar materiales disponibles.	Se mostrarán los materiales disponibles en bodega.
Mostrar estado de solicitudes.	Se mostrará el estado de las solicitudes los cuales pueden ser entregado, pendiente, en espera y rechazada.
Almacenamiento de datos.	El sistema debe proveer de una base de datos que tenga la información de materiales y solicitudes.
Enviar alerta.	El sistema enviará una alerta al usuario bodeguero central cuando una solicitud lleve ¾ del tiempo máximo.
Ingresar material	Se podrá ingresar los materiales registrados en las bodegas.

Req. extra-funcional	**Descripción y medición.**
Tiempo de respuesta	No puede tardar más de 10 segundos en mostrar los datos de materiales o solicitudes.
Consistencia	Al agregar una solicitud o material, este debe actualizarse para todos los usuarios.
Disponibilidad	Que la aplicación esté online sobre el 90% del tiempo
Escalabilidad	Soportar hasta 5000 tipos de materiales.
Seguridad	Los datos son accedidos por los usuarios autorizados.

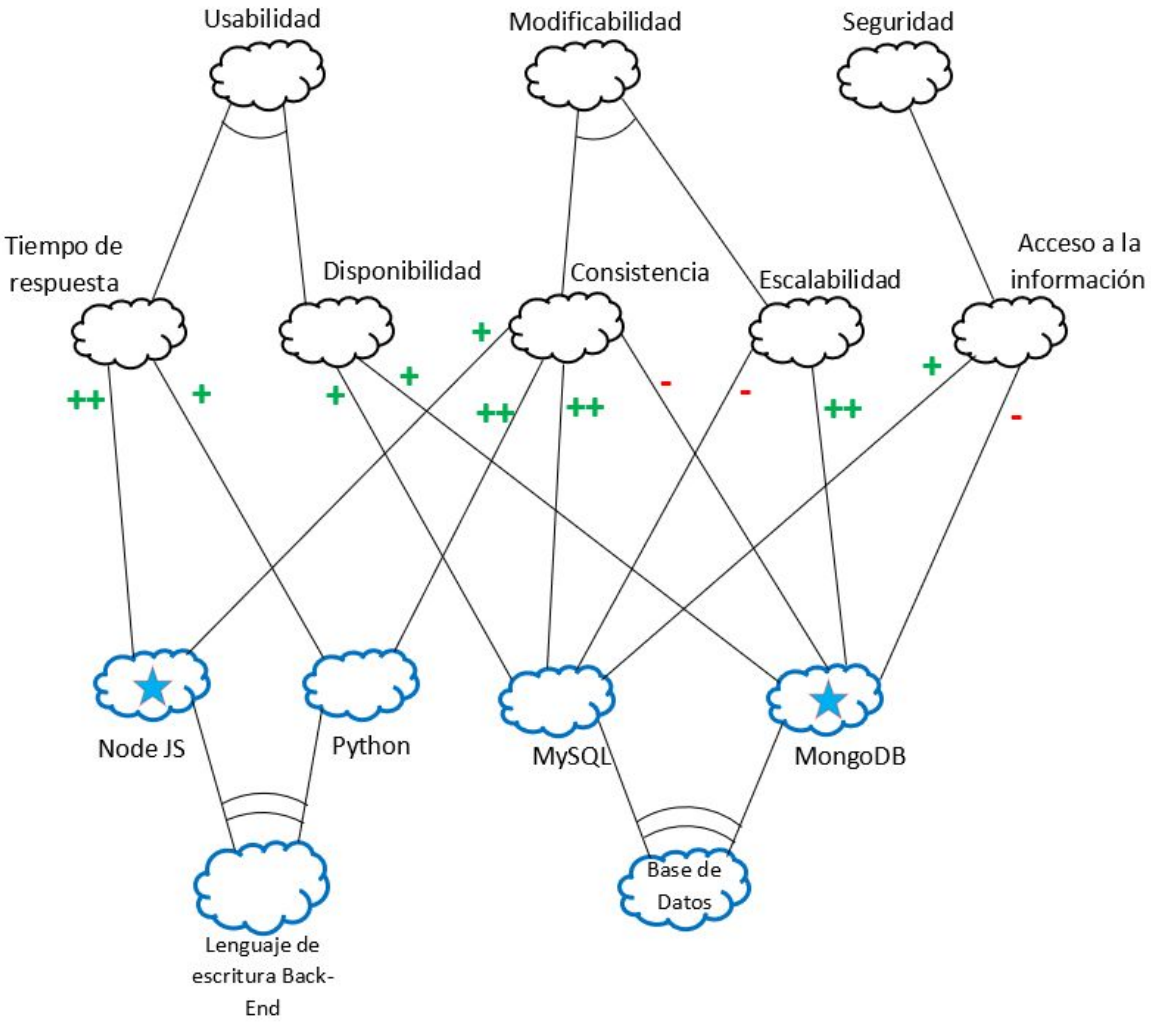
2. Árbol de Utilidad (Final)



3. Modelo de Software (Final)



4. Trade-offs entre tecnologías (final)



Trade-offs entre opciones tecnológicas

A través de la siguiente tabla se explica el porqué de las decisiones tomadas por el equipo AbstractPattern.

Decisión	**Softgoal**	**Evaluación**	**Razonamiento**
Node JS como lenguaje de escritura Back-End	Tiempo de respuesta	++	Al utilizar javascript en el backend realiza las operaciones directamente en el código de la máquina, por lo que resulta muy rapido.

Node JS como lenguaje de escritura Back-End	Consistencia	+	A pesar de que puede ser bastante consistente en cuanto al código para realizar modificaciones, lo extenso de este hace que sea más propenso a errores, además de tener que estar actualizando NodeJS para que funcione en algunos casos.
Python como lenguaje de escritura Back-End	Tiempo de respuesta	+	A pesar de no ser lento, no pasa directamente por el código de la máquina pues tiene algunos objetos determinados directamente, por lo que no resulta tan rapido.
Python como lenguaje de escritura Back-End	Consistencia	++	Al ser más simple de programar y en menos lineas realizar cambios en el código resulta más simple junto con cometer menos errores.
MongoDB como Base de datos	Disponibilidad	+	Presenta menos tiempo entre una falla primaria y su recuperación.
MongoDB como Base de datos	Consistencia	-	No provee transacciones atómicas para múltiples documentos.
MongoDB como Base de datos	Escalabilidad	++	Está diseñada para grandes cantidades de datos y rápido crecimiento de estos.
MongoDB como Base de datos	Acceso a la información	-	Por ser basada en objetos, los datos son almacenados de forma más compleja.
MySQL como Base de datos	Disponibilidad	+	Cuenta con necesidad de privilegio y sistema de seguridad para acceder a los datos.
MySQL como Base de datos	Consistencia	++	Soporta transacciones atómicas y privilegia la seguridad.
MySQL como Base de datos	Escalabilidad	-	Por su forma de modelar los datos, es lenta para adaptarse al crecimiento de los datos.
MySQL como Base de datos	Acceso a la información	+	Privilegia la seguridad y los password en el sistema.

Se decidió utilizar MongoDB como mejor base de datos debido a que los datos no tenderán a cambiar de estructura por lo que no se debería ver implicada la consistencia, los datos pueden crecer demasiado en algún momento por la gran cantidad de elementos y órdenes. En el caso de que los pc no sean muy potentes, utilizará menos espacio ya que es dinámico. Es bueno mientras no se requieren transacciones complejas.También se decidió utilizar NodeJS como mejor lenguaje de escritura back-end porque es una estructura conocida, ordenada y bastante rápida, por lo que puede funcionar para entregar resultados en un tiempo limitado.

5. Deuda técnica incurrida

A través de la implementación del software se decidió debido a ciertas razones e implicando ciertos costos realizar las siguientes deudas técnicas:

Tabla 5: Deuda técnica

Ítem deuda técnica	Razonamiento	Impacto
Manejo de la BD	Implementación eficaz	Baja escalabilidad, difícil de migrar a otra tecnología.
Automatización completa de testing	Automatizar sólo las tareas más importantes.	La prueba de otras tareas no será testeada al cambiar el código.
Error en los comentarios de los menús.	No es un error grave, hay otros errores más importantes	Genera que express deje de funcionar.
Imposibilidad de quitar artículos ya listados	No es un error en si, solo es por usabilidad.	No es un error, pero podría ralentizar el trabajo de quien está listando materiales.
Falta comentar el código	Trabajo contra el tiempo, se priorizo el avance rápido.	El código es más difícil de entender.
Interfaz de software	Enfoque en la funcionalidad del software.	Mayor capacitación para aprender a utilizar el software.

HAR/