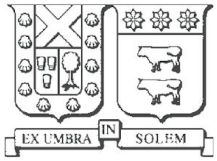
Departamento de Informática
Universidad Técnica Federico Santa María



Informe de Proyecto – INF-225-2018-1-CSJ
Proyecto "BF"
2018-08-05

Integrantes:

Nombres y Apellidos	Email	ROL USM
Florencia Cuzmar Khayatte	florencia.cuzmar@sansano.usm.cl	201573503-4
Benjamin Jorquera Jorquera	benjamin.jorquera.14@sansano.usm.cl	201473521-9
Juan Carlos Bustamante	juan.bustamante.12@sansano.usm.cl	201204757-9

Objetivo de proyecto

Con el fin de facilitar la administración y la gestión de recursos en el sistema de bodegas de la empresa GPI. El proyecto mejorará los tiempos de solicitud y despacho de materiales.

1. Requisitos clave (Actualizado)

Tabla 1: Requisitos funcionales (actualizados)

Req. funcional	Descripción y medición
Crear solicitud de material	Los clientes internos, los supervisores y bodegueros deben ser capaces de realizar una o más solicitudes de material.
Edición de solicitud	Los bodegueros y los supervisores deben poder editar las solicitudes de material hechas por clientes internos. De ser necesario los clientes internos deben poder actualizar sus solicitudes antes de que estas sean aprobadas por la jefatura.
Ver solicitudes	Los usuarios deben poder ver las solicitudes realizadas por ellos mismos y sus estados, mientras que la jefatura, supervisores y bodegueros deben poder ver todas las solicitudes
Aprobar/Rechazar solicitudes	La jefatura debe poder aprobar o rechazar las solicitudes realizadas para que sean atendidas por un supervisor (encargado de adquisición)
Crear órdenes de compra	Los encargados de adquisición deben poder realizar ordenes de compra a partir de solicitudes de material
Aprobar/Rechazar órdenes de compra	La jefatura debe poder aprobar o rechazar las órdenes de compra realizadas
Ver ordenes de compra	El bodeguero debe ser capaz de revisar las órdenes de compra y sus estados
Administrar usuarios	Usuario administrador del sistema puede administrar los usuarios existentes y sus permisos

Tabla 2: Requisitos extra-funcionales (actualizados)

Req. extra-funcional	Descripción y medición
Consistencia	Al agregar o editar una solicitud o material, este se debe actualizar para los usuarios correspondientes.
Usabilidad	El tiempo de respuesta al mostrar las tablas debe ser, en promedio, menor a 10s.
Seguridad	Los datos tienen que ser accedidos solo por los usuarios autorizados.
Consistencia	No permitir a más de un usuario editar la misma entrada al mismo tiempo.
Modificabilidad	En caso de necesitar agregar nuevos campos a alguna de las tablas, esto puede hacerse de manera flexible.

2. Árbol de Utilidad (Actualizado)

Prioridades: A: Alta, M: Media, F: Baja

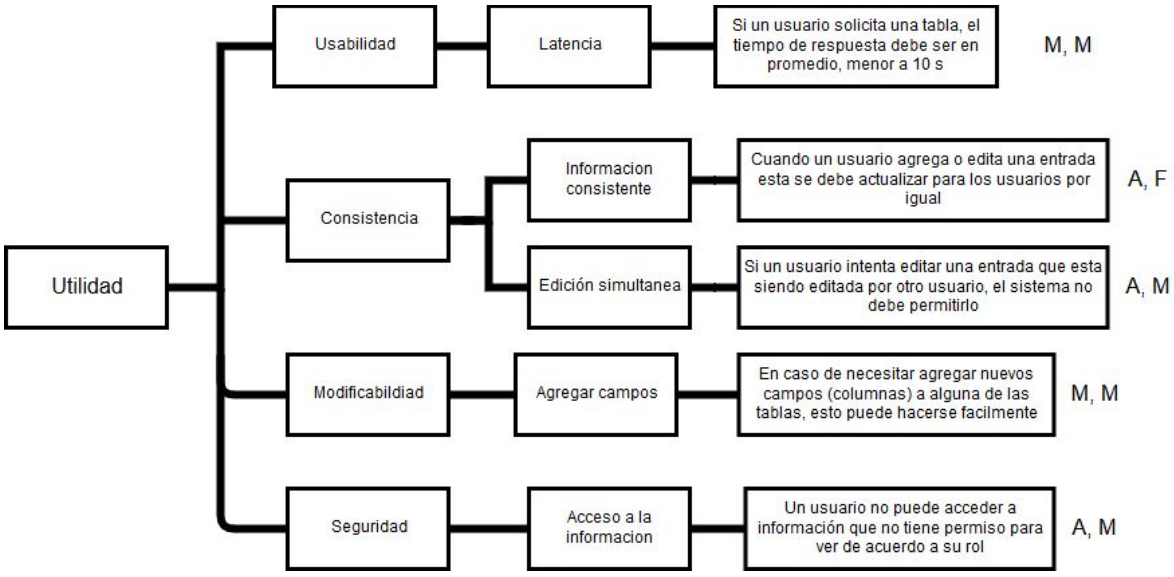


Ilustración 1: Árbol de utilidad actualizado

3. Modelo de Software

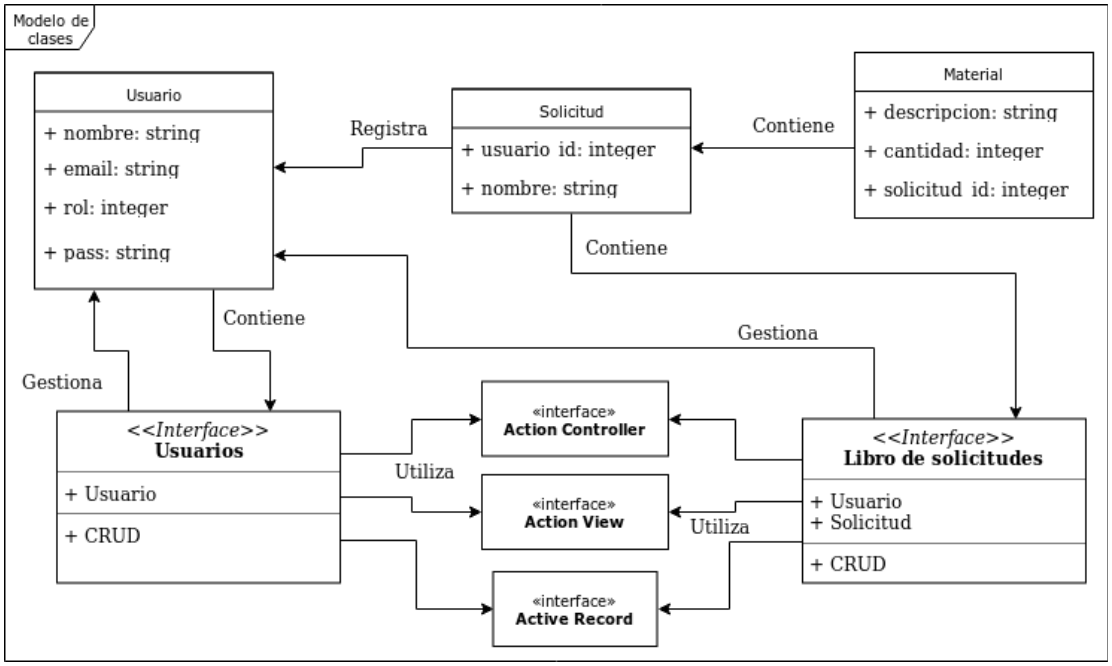


Ilustración 2: Modelo de Software

Documentación:

http://www.agilemodeling.com/style/classDiagram.htm

http://thinkingonthinking.com/ruby-and-patterns/

https://guides.rubyonrails.org/action_view_overview.html

https://guides.rubyonrails.org/form_helpers.html

https://guides.rubyonrails.org/active_record_basics.html

https://guides.rubyonrails.org/action_controller_overview.html

http://blog.returnly.com/3-ruby-design-patterns-we-use-every-day

Tabla 3: Selección de Patrones

Intención	**Patrón de Diseño**	**Razonamiento**
Utilizar un Framework (Ruby On Rails)	MVC	Desarrollo basado en Patrones de Arquitectura para aplicaciones empresariales, además del conveniente uso de las gems como librerías del lenguaje.
Desarrollar una aplicación RESTful con panel dinámico	Action Controller	Utilizar un intermediario entre los modelos y las vistas. Action Controller se preocupa de comunicarse con la base de datos y realizar operaciones CRUD cuando sea necesario. Este patrón realiza la mayor parte del trabajo preliminar para dar sentido a las solicitudes del cliente y producir un resultado apropiado.
Entrada de datos a través de formularios	Action View	Éste patrón proporciona *view helpers* muy útiles para generar formularios sencillos que procesen la entrada de datos del cliente y compilar una respuesta.
Integrar una base de datos	Active Record	Implementar la lógica del negocio utilizando un patrón que permita la creación y el uso de objetos con almacenamiento persistente y mapeo relacional.
Sesiones de usuario	Memorization	El patrón de memorización nos permite ejecutar consultas o cálculos costosos una vez por sesión, y luego reutilizar los resultados tantas veces como sea necesario después (CanCan Ability).

4. Trade-offs entre tecnologías

AND: Arco simple, OR: Arco doble

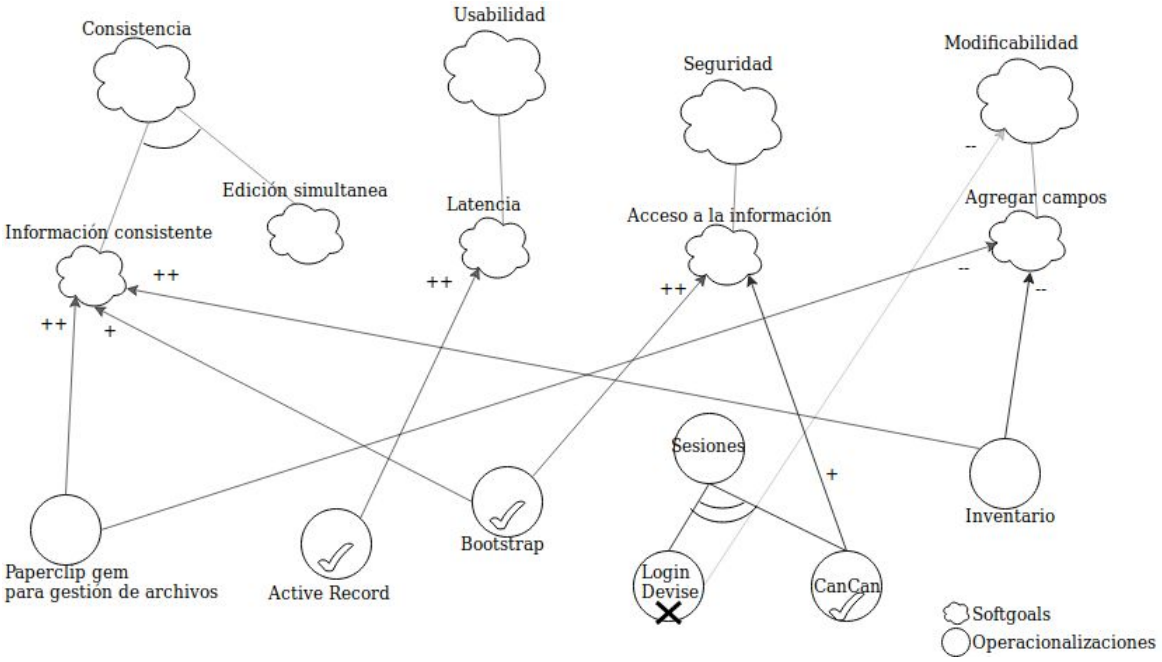


Ilustración 3: Softgoals

Tabla 9: Trade-offs entre opciones tecnológicas

Decisión	Softgoal	Evaluación	Razonamiento
Utilizar CanCan para autorización	Seguridad	+	CanCan es fácil de entender y configurar, y funciona bien para proyectos que no requieren de demasiados roles y habilidades
Implementar un sistema de inventario	Modificabilidad	-	Por un lado es organizado, por otro es complicado manejar más modelos
Interfaz amigable	Usabilidad	++	Bootstrap hace que sea más fácil crear una interfaz estética y amigable
Adjuntar y descargar archivos	Consistencia	-/+	La idea es mejorar el proceso interno de mensajería entre los clientes e incorporar los requisitos de orden de compra.