

IR-Board: Requirements Management Platform

Documento de Especificación de TFG

Summary of Chapters

1. Descripción general	1
1.1. Título	1
1.2. Resumen	1
2. Decisiones arquitectónicas	2
3. Funcionalidades de alto nivel	3
3.1. Gestión de proyectos	3
3.2. Gestión de stakeholders	3
3.3. Gestión de requisitos	3
3.4. Gestión de usuarios	3
3.5. Gestión de documentos asociados y modelado (Integración Draw.io)	4
3.6. Control de variantes	4
3.7. Control de versiones	4
4. Requisitos no funcionales	5

1. Descripción general

1.1. Título

IR-Board: Requirements management platform

1.2. Resumen

Se pretende crear una herramienta que permita dar soporte a los procesos involucrados en la gestión de requisitos: gestión de fuentes e interesados, obtención y refinado de requisitos permitiendo la comprobación de los atributos de calidad, y dando soporte a distintas estándares y normas (guías del IEEE 830 y el estándar de la ISO 29148) Se incorporarán facilidades para incluir información de técnicas estáticas y dinámicas (como casos de uso, diagramas de flujo y tablas de decisión) y la aplicación de metodologías ágiles como las historias de usuario.

2. Decisiones arquitectónicas

- Debido a petición de los tutores, se decidió un sistema web sobre standalone.

Se elige React con Material UI debido a su conocimiento previo, además de una gran versatilidad y enfoque destinado a portabilidad y adaptabilidad. Se prefiere sobre la madurez de otros frameworks como Angular, o sobre bibliotecas de componentes como Ant Design o TailwindCss debido a dicho enfoque y simplificación del desarrollo.

- Se elige SpringBoot sobre Express.js por el entorno más seguro por defecto, y valorando que ambos tienen una buena integración con la tecnología de frontend elegida.
- A fin de facilitar su implantación en otros sistemas, se decide separar el frontend y backend en proyectos separados para facilitar su adaptación a otros entornos.
- Arquitectura web simulando trabajo colaborativo similar al word con los mutex a nivel de párrafo = requisito
- La asociación entre elementos a fin de marcar como pendientes de revisar debe ser bidireccional, y de forma que el patrón de publicador-suscriptor debe estar unido para cada entidad. Para evitar bucles infinitos, se puede hacer una revisión por separado de una modificación, que a efectos, aplica el mutex de la misma manera, y o bien valida o modifica aquello marcado. Se ha decidido ante la visión que, aunque se pueda entrar en un bucle de modificar todo, forma parte del proceso de refinado, y es en esencia, una buena práctica.

3. Funcionalidades de alto nivel

3.1. Gestión de proyectos

- Creación de proyecto
- Borrado de proyecto
- Modificado de proyecto
- Asignación de proyecto a usuarios
- Vista general de proyecto
 - % de requisitos aprobados
 - Requisitos pendientes de revisión

3.2. Gestión de stakeholders

- Añadido de stakeholder
- Borrado de stakeholder
- Modificado de stakeholder
 - Flagging de requisitos asociados a revisar.

3.3. Gestión de requisitos

- Visión de requisito
 - Si otro usuario está editando el requisito, otros usuarios lo verán en modo lectura, con un indicador de quién lo está editando.
- Borrado de requisito
 - Debe exigirse una confirmación
 - Flagging de otros que lo referencien
- Añadido de requisito
 - El guardado del requisito lo envía al servidor y pasa a ser modificado de un requisito existente.
 - Linking con stakeholder (Observer pattern)
 - Linking con otros requisitos
- Modificado de requisito
 - Mutex de edición, otros usuarios no puede editar este requisito hasta su liberación.
 - Se libera el requisito tras un timeout de inactividad.
 - Se libera el requisito si se intenta modificar otro requisito diferente.
 - Linking con stakeholder (Observer pattern)
 - Linking con otros requisitos
 - Al guardado con cambios, debe marcarse el material asociado como pendiente de revisar
- Revisión de material marcado como pendiente de revisar
 - Se permite o bien modificar el requisito o marcarlo como correcto.
- Filtrado
 - Según orden natural
 - Según orden

3.4. Gestión de usuarios

- Añadido de usuario

- Niveles de permiso: (Admin - (proyectos, gest. usuarios, asignado gestor proyecto), gestor de proyecto - (asignar usuarios como ingeniero requisitos), ingeniero requisitos - (gestión de requisitos de medio - bajo), Usuario stakeholder (solo permiso de visión))
- Borrado de usuario
- Modificado de usuario
- Asignación de usuarios
 - Admin puede asignar a proyectos.
 - Gestor de proyecto puede asignar como ing. requisitos de sus proyectos.
 - Puede asignar permisos de acceso a editar según que niveles de la estructura (requisitos de alto - medio - bajo).
 - Puede asignar permisos de gestor de archivos (añadir, borrar, editar)
 - (Opcional) Generar un grupo con conjunto de permisos (copiar a linux)

3.5. Gestión de documentos asociados y modelado (Integración Draw.io)

- Visión de documento
 - Visión de asociados
- Añadido de documento
- Asociado de documento
 - A otro documento
 - A uno/s requisito/s
- Modificación
 - Flagging de documento al cambiar un requisito/funcionalidad asociada
- Borrado
- Modelado de diagramas mediante integración de Draw.io

3.6. Control de variantes

Muy ambicioso solo si voy bien de tiempo. Tenerlo en cuenta aunque no lo implemente, reutilización de requisitos. Definición de plantillas o proyectos abstractos. En la industria se tira por copia y pega.

3.7. Control de versiones

Aparte de UUID interno y orden externo (del usuario). Número de versión del PROYECTO. Semantic versioning Major-Minor.

- Guardado de versión
- Recargado de versión

4. Requisitos no funcionales

- RN-01 Persistencia Normativa: La base de datos debe permitir la trazabilidad exigida por la ISO 29148 (id, descripción, prioridad, estado, origen, justificación).
- RN-02 Control de Concurrencia: El sistema debe implementar un mecanismo de bloqueo a nivel de requisito (mutex) para evitar condiciones de carrera durante la edición simultánea.
- RN-03 Disponibilidad y Web: Acceso multiplataforma mediante navegador, garantizando un diseño responsive gracias a Material UI.
- RN-04