

ReactivePlan for JIRA

Memoria del Proyecto

Autor: Eduardo Pertierra Puche
Director: Carles Farré Tost
Profesor de GEP: Joan Sarda Ferrer
Especialidad: Ingeniería del Software
Fecha: 25/6/2019

Índice

| | |
|--|----|
| ReactivePlan for JIRA | 1 |
| Resumen | 8 |
| Resum | 8 |
| Summary | 8 |
| 1. Contexto | 9 |
| 1.1 Introducción | 9 |
| 1.2 Stakeholders | 9 |
| 1.2.1 Realizador del Proyecto | 9 |
| 1.2.2 Director del Proyecto | 9 |
| 1.2.3 Gestores de Proyecto | 10 |
| 1.2.4 Desarrolladores | 10 |
| 1.2.5 Atlassian | 10 |
| 2. Estado del Arte | 11 |
| 2.1 Contexto | 11 |
| 2.2 ReactivePlan | 11 |
| 2.2.1 Arquitectura de ReactivePlan | 12 |
| 2.3 JIRA | 12 |
| 2.3.1 Funcionamiento | 12 |
| 2.4 Portfolio for JIRA | 13 |
| 2.4.1 Portfolio for JIRA vs Replan | 13 |
| 3. Alcance | 14 |
| 3.1 Definición del Alcance | 14 |
| 3.2 Obstáculos | 14 |
| 3.2.1 Documentación de JIRA muy extensa | 14 |
| 3.2.3 Curva de aprendizaje del desarrollo de plugins en JIRA | 14 |
| 3.3 Riesgos | 14 |
| 3.3.1 Diferencias radicales entre JIRA y Replan | 14 |
| 3.3.2 Restricción Temporal | 15 |
| 4. Metodología de Trabajo | 16 |
| 4.1.1 Análisis y Adaptación Conceptual de Replan a JIRA. | 16 |
| 4.1.2 Análisis y Aprendizaje del Desarrollo de Plugins en JIRA | 16 |
| 4.1.3 Integración de ReactivePlan a JIRA mediante un Plugin | 16 |
| 4.1.4 Pruebas al Plugin de ReactivePlan para JIRA | 16 |
| 4.2 Seguimiento | 17 |

| | |
|--|----|
| 4.3 Validación | 17 |
| 5. Planificación Temporal | 18 |
| 5.1 Duración del Proyecto | 18 |
| 5.2 Recursos | 18 |
| 5.2.1 Recursos Humanos | 18 |
| 5.2.2 Recursos Materiales | 19 |
| 5.2.3 Software a Emplear | 19 |
| 5.3 Descripción de Tareas | 19 |
| 5.3.1 Tarea 0: Estudio Previo | 20 |
| 5.3.2 Tarea 1: Análisis y Especificación de Requisitos | 20 |
| 5.3.3 Tarea 2: Desarrollo del Plugin en JIRA | 20 |
| 5.3.3 Tarea 3: Pruebas del Plugin y Documentación | 21 |
| 5.4 Diagrama de Gantt | 22 |
| 5.5 Valoración de alternativas y plan de acción | 22 |
| 5.5.1 Caso 1: No da tiempo a implementar algún requisito | 22 |
| 5.5.2 Caso 2: Baja por enfermedad | 22 |
| 5.5.3 Caso 3: Falta o no disponibilidad de recursos | 22 |
| 5.5.4 Caso 4: Problema no valorado | 23 |
| 5.6 Desviaciones respecto a la planificación inicial | 23 |
| 5.6.1 Desviación 1: Prolongación de la primera fase de Desarrollo. | 23 |
| 5.6.1.1 Desviación 1: Repercusión Temporal | 23 |
| 5.6.1.2 Desviación 1: Repercusión Económica | 23 |
| 5.6.1.3 Desviación 1: Repercusión de Objetivos | 23 |
| 6. Análisis de Sostenibilidad | 24 |
| 6.1 Introducción | 24 |
| 6.1.1 Cuestionario de Estudiantes de Ingeniería Informática | 24 |
| 6.2 Gestión Económica | 24 |
| 6.2.1 Introducción de los Costes | 24 |
| 6.2.2 Costes Indirectos | 24 |
| 6.2.3 Costes Directos por actividad | 25 |
| 6.2.4 Imprevistos | 25 |
| 6.2.5 Contingencia | 26 |
| 6.2.6 Costo Total | 26 |
| 6.2.7 Control de Gestión | 26 |
| 6.2.8 Análisis del Impacto Económico | 27 |
| 6.3 Análisis del impacto Medioambiental | 28 |

| | | |
|---------|---|----|
| 6.3.1 | Impacto ambiental del proyecto | 28 |
| 6.3.2 | Minimización del impacto ambiental | 28 |
| 6.3.3 | Vida útil del proyecto a nivel ambiental | 28 |
| 6.4 | Análisis del impacto social | 28 |
| 6.4.1 | Aportación Personal del desarrollo del proyecto | 28 |
| 6.4.2 | Resolución actual del problema y mejora social | 29 |
| 6.4.3 | Necesidad del Proyecto | 29 |
| 7. | Análisis de Alternativas | 30 |
| 7.1 | Herramienta Replan | 30 |
| 7.1.1 | Replan Controller | 30 |
| 7.1.2 | Replan Optimizer | 30 |
| 7.1.3 | Comparación Replan Controller frente a Replan Optimizer | 31 |
| 7.1.4 | Herramienta Escogida | 31 |
| 7.2 | Integración en JIRA | 31 |
| 7.2.1 | JIRA Rest API | 31 |
| 7.2.2 | Atlassian JIRA - Server 8.1.0 API | 32 |
| 8. | Representación de una planificación en JIRA | 33 |
| 8.1 | Conceptos | 33 |
| 8.1.1 | Project | 33 |
| 8.1.1.1 | Version | 33 |
| 8.1.1.2 | Component | 33 |
| 8.1.2 | Issue | 33 |
| 8.1.2.1 | Issue Type | 35 |
| 8.1.2.2 | Priority | 35 |
| 8.1.2.3 | Resolution | 36 |
| 8.2 | Planificación | 36 |
| 9. | Representación de una planificación en Replan Optimizer | 37 |
| 9.1 | Resource | 37 |
| 9.2 | Feature | 37 |
| 9.3 | Calendar | 37 |
| 9.4 | Skills | 37 |
| 9.5 | PriorityLevel | 37 |
| 10. | Comparación Conceptual JIRA - Replan | 38 |
| 10.1 | Modelo Conceptual de Replan | 38 |
| 10.2 | Modelo Conceptual de Replan Optimizer | 39 |
| 10.2.1 | Modelo Conceptual Request | 39 |

| | |
|---|----|
| 10.2.2 Modelo Conceptual Solution | 40 |
| 10.3 Modelo Conceptual de JIRA | 41 |
| 10.4 Comparación de Elementos | 41 |
| 10.5 Comparación de atributos | 42 |
| 10.5.1 Project - Project | 42 |
| 10.5.2 Version - Release | 42 |
| 10.5.3 Issue - Feature | 43 |
| 10.5.4 ProjectRole - Skill | 43 |
| 10.5.4 User - Resource | 43 |
| 10.5.5 Calendar | 43 |
| 11. Especificación del Sistema | 44 |
| 11.1 Requisitos Funcionales | 44 |
| RF 11.1.1 Realizar Planificación Automática | 44 |
| RF 11.1.2 Seleccionar Proyecto y Versión a Planificar | 44 |
| RF 11.1.3 Visualizar Issues a Planificar | 44 |
| RF 11.1.4 Previsualizar Plan | 44 |
| RF 11.1.5 Persistir Plan | 44 |
| 11.2 Requisitos no Funcionales | 45 |
| 11.2.1 Requisitos de Apariencia | 45 |
| 11.2.2 Requisitos de Usabilidad | 45 |
| 11.2.3 Requisitos de Escalabilidad | 45 |
| 11.3 Casos de Uso | 46 |
| 11.3.1 Diagrama de Casos de Uso | 46 |
| CU 1 Seleccionar Proyecto y Versión | 47 |
| CU 2 Visualizar Issues a Planificar | 47 |
| CU 3 Previsualizar Plan | 48 |
| CU 4 Persistir Plan | 48 |
| 12. Diseño del Sistema | 49 |
| 12.1 Arquitectura de un Plugin en JIRA | 49 |
| 12.2 Módulos de ReactivePlan for JIRA | 50 |
| 12.2.1 Web Item | 50 |
| 12.2.2 Servlet | 50 |
| 12.3 Diagramas de Clases de la Solución | 51 |
| 12.3.1 Entidades | 51 |
| 12.3.2 Lógica | 53 |
| 12.3.3 Conversores | 54 |

| | |
|--|----|
| 12.3.4 API Handlers | 55 |
| 12.3.5 Servlet | 55 |
| 12.3 Diagrama de Paquetes de la Solución | 56 |
| 12.4 Diagrama de Navegación | 57 |
| 12.5 Diseño interno de la capa de Presentación | 58 |
| 12.5.1 Diseño interno del CU1 y CU2 | 58 |
| 12.5.2 Diseño interno del CU3 | 59 |
| 12.5.3 Diseño interno del CU4 | 60 |
| 12.6 Diseño de llamadas: Diagramas de Secuencia | 61 |
| 12.6.1 CU1 - Diagrama de Secuencia | 61 |
| 12.6.2 CU2 - Diagrama de Secuencia | 62 |
| 12.6.3 CU3 - Diagrama de Secuencia | 63 |
| 12.6.4 CU4 - Diagrama de Secuencia | 63 |
| 12.7 Tecnologías a Emplear | 64 |
| 12.7.1 Back-End | 64 |
| 12.7.2 Front-End | 64 |
| 12.8 Patrones de Diseño Empleados | 64 |
| 12.8.1 Singleton | 64 |
| 12.8.2 Modelo Vista Controlador | 64 |
| 13. Implementación del Sistema | 65 |
| 13.1 Atlassian SDK | 65 |
| 13.2 Replan Optimizer | 65 |
| 13.3 IntelliJ IDEA 2019.1 | 65 |
| 13.4 POSTMAN | 66 |
| 13.5 Github y Github Desktop | 66 |
| 13.6 Andamiaje del Proyecto | 66 |
| 13.6.1 Andamiaje JIRA | 67 |
| 13.6.2 Andamiaje REPLAN | 67 |
| 13.6.3 Andamiaje del Front-End | 68 |
| 13.7 Librerías Externas Empleadas | 68 |
| 14. Pruebas | 69 |
| 14.1 Pruebas Funcionales | 69 |
| 14.2 Pruebas de Sistema | 70 |
| 14.2.1 Definición del conjunto de pruebas. | 70 |
| 14.2.1.1 Caso de Prueba 1. Planificación de Todo el Proyecto | 72 |
| 14.2.1.2 Caso de Prueba 2. Planificación de la versión 1.0 | 74 |

| | |
|---|----|
| 14.2.1.3 Caso de Prueba 3. Planificación de la versión 2.0 | 76 |
| 14.3 Errores Detectados a través de las diferentes pruebas. | 77 |
| 15. Despliegue | 78 |
| 16. Legislación Sobre el proyecto | 79 |
| 16.1 Aspectos Relevantes del Proyecto | 79 |
| 16.2 Aplicación de la GDPR por parte de JIRA Atlassian. | 79 |
| 17. Conclusión | 80 |
| 17.1 Cumplimiento de Objetivos | 80 |
| CES1.1: Desarrollar, mantener y evaluar sistemas y servicios software complejos y/o críticos. | 80 |
| CES1.2: Dar solución a problemas de integración en función de las estrategias, de los estándares y de las tecnologías disponibles. | 80 |
| CES1.3: Identificar, evaluar y gestionar los potenciales riesgos asociados a la construcción de software que se pudiesen presentar. | 81 |
| CES1.4: Desarrollar, mantener y evaluar servicios y aplicaciones distribuidas con soporte de la red. | 81 |
| CES1.7: Controlar la calidad y diseñar pruebas en la producción de software. | 81 |
| CES2.1: Definir y gestionar los requisitos de un sistema de software. | 81 |
| 17.2 Trabajo Futuro | 81 |
| 18. Conocimientos Empleados | 82 |
| 18.1 Programación | 82 |
| 18.1.1 Programación Orientada a Objetos | 82 |
| 18.1.2 Estructura de Datos | 82 |
| 18.2 Tecnologías | 83 |
| 18.2.1 Tecnologías Web | 83 |
| 18.2.2 Aplicaciones y Servicios Web | 83 |
| 18.3 Análisis Funcional y Diseño | 83 |
| 18.3.1 Modelado y Diseño de Software | 83 |
| 18.3.2 Ingeniería de Requisitos | 84 |
| 18.3.3 Interfaces de Usuario | 84 |
| 18.4 Gestión Proyectos | 84 |
| 18.4.1 Gestión de Proyectos | 84 |
| 18.5 Pruebas de Software | 84 |
| 18.5.1 Mantenimiento y Pruebas de Software | 84 |
| Glosario | 85 |
| Bibliografía | 86 |

Resumen

Este trabajo de fin de grado se centra en el diseño y desarrollo de un plugin para la plataforma de desarrollo colaborativo JIRA. El propósito de este plugin es extraer los datos que se presenten en esta plataforma y ofrecérselos a Replan (una herramienta de planificación automática) de tal forma que este pueda planificar las distintas versiones de los proyectos que se realicen. Además de eso, este trabajo también implica un profundo estudio de la herramienta JIRA para poder desarrollar dicho plugin.

Resum

Aquest treball de fin de grau es centra en el disenny i desenvolupament d'un plugin per la plataforma de desenvolupament col·laboratiou JIRA. El propòsit d'aquest plugin és extreure les dades que es presenten en aquesta plataforma i oferir-los a Replan (una eina de planificació automàtica) de tal manera que aquest pugui planificar les diferents versions dels projectos que es realitzin. A més d'això, aquest treball també implica un profund estudi de l'eina JIRA per poder esenvolupar aquest plugin.

Summary

This final project focuses on the design and development of a plugin for the collaborative development platform JIRA. The purpose of this plugin is to extract the data presented in this platform and offer it to Replan (an automatic planning tool) so that it can plan the different versions of the projects that are carried out. Besides that, this work also involves a deep study of the JIRA tool to be able to develop such a plugin.

1. Contexto

1.1 Introducción

Este documento hace referencia a un Trabajo Final de Grado de la Facultad de Informática de Barcelona (Universidad Politécnica de Cataluña) llamado *ReactivePlan for Jira*.

Se trata de un proyecto de la especialidad *Ingeniería del Software*. Este proyecto ha sido propuesto por el departamento de *Ingeniería de Servicios y Sistemas de la Información* (ESSI).

En este departamento y a través de diversos proyectos Europeos como *SUPERSEDE* [1], se ha desarrollado una herramienta software llamada *ReactivePlan* [2] cuya finalidad es reforzar la conexión entre el desarrollo de software y las actividades de gestión de proyectos a través de fomentar lo denominado *Continuous Software Release Planning (CSRP)*, permitiendo al usuario de esta herramienta mediante una breve configuración, crear un plan para su proyecto de forma automática así como replanificarlo en caso de que sea necesario (p. ej si una tarea se finaliza antes de tiempo o se tarda más de lo debido).

Cabe destacar que el desarrollo de software es todo el proceso que se encarga de la creación software desde la especificación de los requisitos hasta la implementación y las pruebas, mientras que la gestión de proyectos decide quién y cuándo realiza cada tarea a la hora de desarrollar éste.

Lo que se pretende a través de este proyecto es facilitar el uso de la herramienta *ReactivePlan* mediante su integración con JIRA.

JIRA es una popular herramienta de software comercializada y desarrollada por *Atlassian* [3] empleada principalmente para llevar a cabo la gestión de proyectos ágiles. Una de las mayores ventajas de JIRA es que permite la adición de extensiones (third-party extension) para ampliar su funcionalidad.

1.2 Stakeholders

Los stakeholders de este proyecto son todas aquellas personas, empresas o instituciones interesadas en que el proyecto salga adelante debido a que obtendrán algún tipo de beneficio en el caso de que el proyecto resulte exitoso.

1.2.1 Realizador del Proyecto

Es aquel que se encarga tanto de realizar el desarrollo del proyecto así como ésta propia documentación y lograr que el proyecto se realice con éxito.

Es, probablemente la parte más interesada en el proyecto puesto que es aquel que lo desarrolla, está implicado totalmente en él y, en este caso al ser un *Trabajo Final de Grado*, obtendrá una calificación en función de la calidad resultante del proyecto.

1.2.2 Director del Proyecto

Profesor del departamento de *Ingeniería de Servicios y Sistemas de la Información* (ESSI) que se encargará de dirigir tanto el TFG como de guiar y supervisar todos aquellos aspectos que desarrolle el realizador de este proyecto.

1.2.3 Gestores de Proyecto

Miembro de un equipo de desarrollo de software encargado de gestionar los proyectos y sus lanzamientos.

Cualquier Gestor de Proyectos podrá resultar beneficiado si este proyecto resulta exitoso puesto que disminuirá considerablemente el tiempo que debe emplear en gestionar y desarrollar los diversos planes de proyecto.

1.2.4 Desarrolladores

Miembro de un equipo de desarrollo de software que realiza las tareas técnicas.

Esto proyecto interesa a los desarrolladores ya que cambiará la forma de ver los proyectos para un desarrollador de software automatizando la tediosa tarea de organizar a un equipo de desarrollo.

1.2.5 Atlassian

La empresa que desarrolla JIRA.

Le interesa el proyecto puesto que si resulta exitoso podría resultar en que su plataforma atraiga a más desarrolladores popularizando su herramienta.

2. Estado del Arte

2.1 Contexto

Decidir qué se ha de implementar, cuando y por qué es una actividad que corresponde al gestor de proyectos en cada proyecto, y esta actividad es conocida como *Software Release Planning*. En cualquier empresa de software ésta es una actividad crucial ya que influirá directamente en que el producto se lance a tiempo o no, así como en el bienestar de los desarrolladores y la flexibilidad de cara a reaccionar a todos los problemas que puedan surgir durante el desarrollo de software.

Para ello actualmente, existen diversas herramientas (Microsoft Project, WorkFront, Trello, JIRA...) que permiten organizar tareas de una forma simple, sin embargo no tienen la capacidad de planificar todo el desarrollo de un producto software desde el planteamiento de las historias de usuario hasta su lanzamiento a través de algoritmos genéticos, como es el caso de *ReactivePlan*. Además de ese factor, ninguno tiene la capacidad de replanificar un proyecto automáticamente en función de los cambios de éste a tiempo real.

Esto da lugar a que la adaptación de *ReactivePlan* a diversas herramientas de gestión de proyectos resulte algo totalmente innovador, y es por ello que en el año 2017 en la Facultad de Informática de Barcelona se desarrolló una primera integración de *ReactivePlan* con la plataforma de gestión de proyectos *Trello* [4] en el trabajo de fin de grado titulado "*Anàlisi de l'activitat dels desenvolupadors en plataformes de suport al desenvolupament col·laboratiu*" [5]. Trabajo en el que podemos encontrar una primera investigación sobre el CSRP y su integración con *Trello*.

Y es por ello que esta investigación es totalmente innovadora al tratar de integrar plenamente *ReactivePlan* con JIRA.

2.2 ReactivePlan

ReactivePlan surge de la necesidad de automatizar en la medida de lo posible la gestión de proyectos, y es por ello que para dar dicho soporte automático desde el proyecto SUPERSEDE se comenzó a desarrollar ésta herramienta en el año 2015.

ReactivePlan (Replan en su abreviación) es una herramienta de gestión de proyectos cuya finalidad es facilitar de un gestor de proyectos en una empresa de desarrollo de software. Para ello, esta herramienta tratará de planificar los releases de un proyecto de forma eficiente teniendo en cuenta los recursos humanos, temporales y el número de tareas del mismo.

Acto seguido, Replan facilitará diversas planificaciones donde se podrá encontrar las fechas de comienzo y finalización de cada tarea así como la persona a la que está asignada dicha tarea.

Finalmente lo que obtendremos será una planificación de principio a fin de un proyecto de software.

Cabe destacar que Replan también tiene la posibilidad de modificar esta planificación a mediados del proyecto en el caso de que estos plazos sean poco asequibles, difíciles de cumplir o haya algún factor externo que demore el proyecto, siendo éste uno de sus puntos fuertes.

2.2.1 Arquitectura de ReactivePlan

Actualmente la herramienta ReactivePlan cuenta con tres componentes, un **Dashboard**, un **Controller** y un **Optimizer**.

- **Dashboard:** Es una aplicación web con una interfaz gráfica cuyo objetivo es el de proporcionar acceso a nivel de usuario a todas las funcionalidades de Replan. Este componente permite al gestor de proyecto configurar los recursos humanos así como las diversas tareas que pueda contener un proyecto de software.
- **Controller:** Servicio web encargado de dar soporte, gestión y almacenamiento a todos los elementos del dominio de ReactivePlan. Éste servicio web consta de una API REST bien definida mediante la cuál se pueden realizar todas las operaciones correspondientes a la gestión de un proyecto en concreto. Cuando el *controller* obtiene toda la información necesaria para gestionar un proyecto también tendrá la opción de planificarlo, y para ello conectará con el optimizer.
- **Optimizer:** Servicio web encargado de la ejecución del algoritmo genético que transforma toda la información proveída por el *controller* en uno o muchos planes tangibles de cara a que el gestor de proyectos pueda realizar la propuesta de llevarlos a cabo.

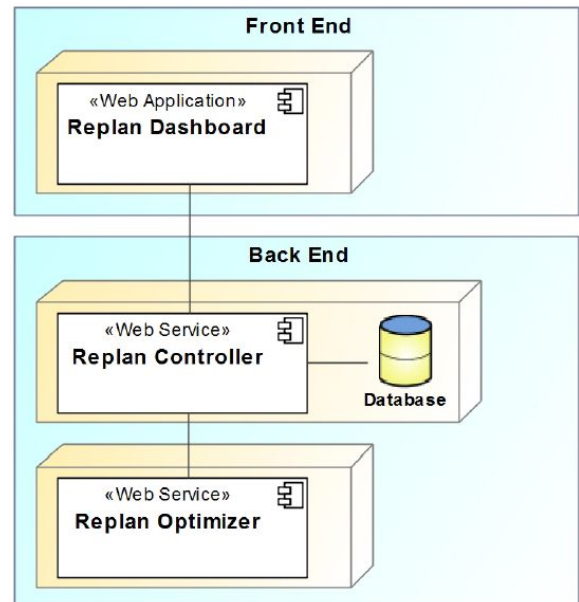


FIG 1. ARQUITECTURA DE REACTIVEPLAN [7]

Cabe decir que debido a que este proyecto será de cara a la integración con JIRA se omitirá totalmente el uso del dashboard y se centrará en el uso o interpretación del *Controller* u *Optimizer*.

2.3 JIRA

JIRA es una herramienta en línea para la administración de tareas de un proyecto, el seguimiento de errores e incidencias y para la gestión operativa de proyectos. La herramienta fue desarrollada por la empresa australiana Atlassian. Inicialmente Jira se utilizó para el desarrollo de software, sirviendo de apoyo para la gestión de requisitos, seguimiento del estado de desarrollo y más tarde para la gestión de errores. Jira puede ser utilizado para la gestión y mejora de los procesos, gracias a sus funciones para la organización de flujos de trabajo [6].

2.3.1 Funcionamiento

El funcionamiento básico de Jira consiste en crear, buscar, y cambiar el estado de los asuntos en los que uno está trabajando. Veamos ahora cómo organizar este trabajo:

- Los asuntos se agrupan en lo que se conoce como **proyectos**. A más alto nivel, los proyectos se pueden agrupar en **categorías** y a más bajo nivel un proyecto puede dividirse en lo que se conocen como **componentes** o **versiones**. Los componentes actúan como etiquetas de los asuntos. Las versiones representan hitos de entrega.
- Un asunto pertenece siempre a un proyecto. Puede pertenecer a varios componentes a la vez e incluso estar asociado a varias versiones, pero pertenece a un único proyecto.

En esta figura podemos observar las relaciones descritas previamente entre asuntos y proyectos.

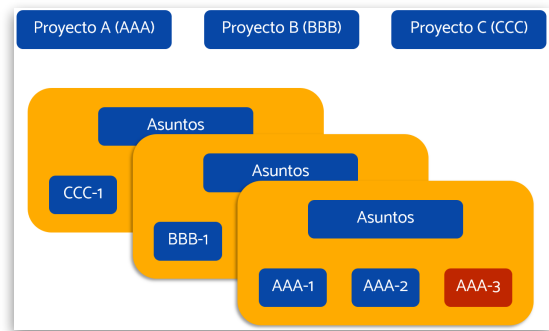


FIG 2. RELACIÓN ENTRE PROYECTOS Y ASUNTOS EN JIRA [3]

2.4 Portfolio for JIRA

Portfolio for Jira es una herramienta que permite a los equipos construir planes y monitorizar el progreso en el **Software de Jira**. Es un Add-On para JIRA que provee una integración directa con el sistema de **JIRA**, tal y como se pretende hacer en el plugin de Replan. También permite realizar una visualización rápida de cualquier plan creado así como jerarquías de trabajo y planificaciones automáticas. Cabe destacar que esta herramienta no se va a utilizar a lo largo del proyecto pero sí se ha de tener en cuenta su existencia para construir una herramienta que se distinga claramente de la actual herramienta de **Atlassian**.

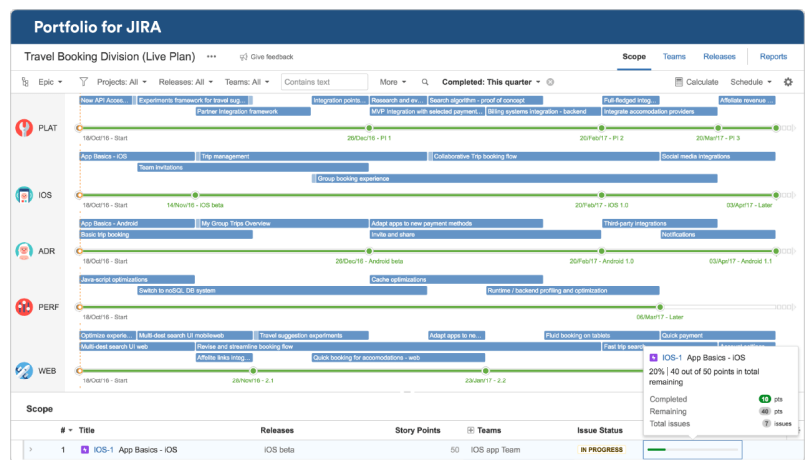


FIG 3. PLANIFICACIÓN EN PORTFOLIO FOR JIRA [3]

2.4.1 Portfolio for JIRA vs Replan

Mientras que **Portfolio for JIRA** trata de ofrecer múltiples funcionalidades de cara al **desarrollo ágil de software**, siendo una de sus características la organización automática de proyectos, lo que se pretende con el **Plugin de Replan** será centrarse en esta organización automática de forma elegante y sencilla de cara al usuario así como su correcta integración con **JIRA**, permitiendo realizar planificaciones de forma rápida y sencilla, adaptándolas directamente a tu proyecto en JIRA sin necesidad de configuraciones o interfaces extra más que la que provee la propia aplicación, mientras que en **Portfolio** se requieren múltiples campos y acciones extra que puede llevar al usuario a que le resulte muy complejo utilizar la herramienta de la planificación automática. De hecho, los propios desarrolladores de este **Portfolio**, reconocen que su herramienta de planificación automática es *algo compleja* de usar y no la recomiendan para usuarios novatos.

3. Alcance

3.1 Definición del Alcance

Lo que se pretende en este proyecto es la realización de un plugin de JIRA que permita integrar ReactivePlan plenamente con esta plataforma y realizar la gestión de los proyectos automáticamente, para ello, lo que se pretende con el proyecto es la creación de un componente de plugin de JIRA que permita lo siguiente:

- Realizar una planificación total y completa estableciendo los parámetros necesarios en JIRA.
- Permitir al usuario establecer los parámetros que contenga ReactivePlan pero no JIRA y viceversa a través de la interfaz del plugin.
- Visualizar el resultado de las planificaciones ofrecidas por ReactivePlan desde la propia plataforma de JIRA, ya sea mediante un diagrama de Gantt u otro tipo de herramientas.
- Adaptar en la medida de lo posible los conceptos de JIRA a ReactivePlan y viceversa con el fin de poder ofrecer la mejor planificación posible.
- Que el propio plugin actualice el proyecto según lo establecido en la planificación.
- En el caso de haber un imprevisto en el proyecto (como que no se realiza una tarea a tiempo o se realiza antes de tiempo), que el plugin sea capaz de reaccionar y reorganizar todo el proyecto pudiendo incluso atrasar o adelantar la fecha de entrega.

3.2 Obstáculos

3.2.1 Documentación de JIRA muy extensa

Además de resultar JIRA una plataforma muy compleja, tanto su esquema conceptual como su documentación resultan muy extensas y puede ocasionar problemas a la hora de entender cómo desarrollar un plugin o incluso resultar incompleta. Esto puede retrasar sustancialmente el proyecto o incluso limitar la implementación de ciertas funcionalidades que se pretenden desarrollar.

3.2.3 Curva de aprendizaje del desarrollo de plugins en JIRA

El desarrollo de plugins en JIRA es una tarea a la que se dedican desarrolladores profesionales y su curva de aprendizaje podría ser tan compleja que se tarde más de lo debido en cumplir los plazos del proyecto. Esto podría acarrear que el proyecto no se cumpla en el plazo propuesto debido a esta dificultad.

3.3 Riesgos

3.3.1 Diferencias radicales entre JIRA y Replan

JIRA y Replan, a pesar de ser herramientas de gestión de proyectos no comparten la mayoría de atributos en sus esquemas conceptuales pudiendo resultar en que alguna de las funcionalidades de Replan no se pueda integrar en JIRA o tengan que integrarse parcialmente. Dicho de otra forma, al no ser JIRA y Replan aplicaciones de software similares, podría resultar excesivamente complejo integrar una herramienta con otra tal y como se pretende hacer en este proyecto.

Estas diferencias se ubican a nivel de implementación y de cómo cada plataforma está montada, pudiendo resultar en que, la integración completa de Replan sea inviable.

3.3.2 Restricción Temporal

Debido a que este proyecto es un Trabajo de Fin de Grado cuenta con una seria restricción temporal sobre cuánto tiempo se debe emplear en desarrollarlo, y echando un ojo a las dimensiones del proyecto, éste tiempo podría no ser suficiente.

4. Metodología de Trabajo

Debido a la duración y evaluación de este proyecto se ha optado por llevar a cabo un enfoque ágil facilitando una retroalimentación constante por parte del director del proyecto.

Este proyecto se tratará de desarrollar de forma iterativa e incremental haciendo a cada iteración un producto entregable y tratando de mejorar las entregas previas, para ello dividiremos el proyecto en cuatro fases o entregas. Cabe destacar que la documentación se irá realizando en paralelo al progreso de las fases en lugar de realizar todo el proyecto y por último una memoria.

4.1.1 Análisis y Adaptación Conceptual de Replan a JIRA.

En esta fase se tratará de poner en común los conceptos de ambas plataforma mediante el desarrollo de esquemas conceptuales y la comparación de los mismos con tal de aclarar hasta qué punto se puede integrar una herramienta como ReactivePlan en JIRA así como tratar de establecer con qué componente de ReactivePlan se hará esta integración (Optimizer o Controller), así como la evaluación de la necesidad de crear un componente intermedio que se encargue de realizar la *traducción* de éstos datos con el fin de desarrollar el proyecto de la forma más eficiente posible.

4.1.2 Análisis y Aprendizaje del Desarrollo de Plugins en JIRA

En esta fase se valorará cuáles son las ventajas y limitaciones sobre el desarrollo de plugins en JIRA con el fin de saber hasta qué punto se puede llegar con el plugin. Inicialmente lo que se desea es poder evaluar si se puede, a la par que conectar ReactivePlan, desarrollar una pequeña interfaz gráfica en el propio plugin de forma que quede totalmente integrado.

Por otro lado se requerirá de tiempo para familiarizarse con el desarrollo de plugins en esta plataforma así como realizar diversas pruebas con el fin de poder dar un producto sólido en la siguiente fase.

4.1.3 Integración de ReactivePlan a JIRA mediante un Plugin

Esta será la fase de desarrollo e integración del Plugin. Lo que se pretende es lograr que JIRA contacte con ReactivePlan y muestre la planificación a seguir según lo que Replan haya considerado oportuno. El alcance de esta fase dependerá totalmente del estudio previo aunque la idea final es lograr integrar plenamente ReactivePlan realizando ésta gestión automática y ofreciendo también una interfaz donde se pueda visualizar todo aquello que corresponde a ReactivePlan.

4.1.4 Pruebas al Plugin de ReactivePlan para JIRA

En esta última fase se realizarán pruebas principalmente de integración con el fin de verificar que el plugin ha sido correctamente desarrollado y no de errores en el futuro, y así poder dar el proyecto por finalizado exitosamente.

4.2 Seguimiento

El seguimiento del proyecto se realizará a través de la plataforma en la que se va a trabajar, JIRA con el fin de familiarizarse con ésta lo más pronto posible. Para ello se definirán una serie de tareas con una fecha límite que deberán cumplirse y entregas periódicas por cada fase del proyecto.

Por otro lado, de cara a realizar el seguimiento del código se utilizará alguna de las plataformas basadas en Git.

4.3 Validación

Con el fin de poder validar cada una de las fases principalmente, se tendrán reuniones con el director del proyecto para que éste pueda dar su visto bueno a cada una de las fases. En el caso de que alguna de las fases no se realice satisfactoriamente o falten requisitos por cumplir se evaluará si se puede añadir a la siguiente fase o si queda descartado debido a las limitaciones del proyecto.

También se pretenden establecer una serie de criterios de satisfacción para todos y cada uno de los **requisitos** a implementar o desarrollar a lo largo del proyecto los cuáles deberán verificarse para poder verificar que éste ha sido correctamente elaborado.

Por último, la validación final de cara al software a desarrollar en este proyecto se realizará mediante pruebas de software que deberá pasar. En el caso de que no se pasen las pruebas no se podrá decir que la funcionalidad ha sido validada de modo que se deberá modificar el requisito a cumplir, las pruebas o depurar los errores que vayan surgiendo en esta parte del desarrollo.

5. Planificación Temporal

Con el fin de poder realizar el proyecto a tiempo así como realizar una correcta gestión de recursos y tareas se procede a llevar a cabo una planificación temporal de la forma más precisa posible donde se organizarán esas tareas en intervalos de tiempo junto a los recursos que se requieran para llevarlas a cabo.

Por otro lado, en este apartado también se pretende elaborar un plan de acción en el caso de que hubiese algún imprevisto o no se pudiese cumplir algún plazo de tal forma que este proyecto quede finalizado en la fecha propuesta.

5.1 Duración del Proyecto

Según la información que se dispone en la página web de la Facultad de Informática de Barcelona [8] este proyecto que conforma un Trabajo de fin De Grado equivale a 15 créditos académicos.

Cada crédito equivale a 48 horas de trabajo individual por parte del alumno [9], de modo que la duración total del proyecto deberá rondar en torno a las 450 horas.

Estas 450 horas se pretenden repartir en jornadas de 6 horas diarias, resultando en 90 días de trabajo.

Por otro lado, se debe elaborar un informe de cara a la asignatura GEP que corresponde a 3 créditos académicos lo cual añade 90 horas en total al proyecto, resultando en 8 jornadas más de trabajo.

Esto resulta a un total de 98 días de trabajo en el proyecto.

De modo que si se tiene en cuenta que este proyecto fue comenzado el día **2 de Febrero de 2019**, se pretende que el proyecto quede totalmente finalizado en torno al **28 de Mayo de 2019** con el fin de ser presentado en **la convocatoria de Julio de 2019** ante un tribunal.

5.2 Recursos

Entre los recursos que se disponen de cara a hacer el proyecto diferenciaremos tres; *recursos humanos, recursos materiales y software*.

5.2.1 Recursos Humanos

Los recursos humanos serán aquellos individuos que desarrollen o colaboren de forma directa en el proceso de desarrollo del proyecto.

Desarrollador del Proyecto: Esta persona se encargará de realizar todo el trabajo de cara al proyecto, desde la documentación hasta el código con el fin de que el proyecto salga adelante. Su trabajo deberá ser de en torno a 40 horas semanales (incluidas reuniones).

Director del Proyecto: Esta persona se encargará de orientar al desarrollador así como de tratar de resolver sus dudas. Su trabajo será de en torno a 1 o 2 horas semanales según la necesidad que tenga el desarrollador de reunirse con éste.

5.2.2 Recursos Materiales

Con el fin de desarrollar este proyecto se requerirá de un **ordenador con acceso a internet**. En este caso, se desarrollará con uno con las siguientes características:

Modelo: Macbook Pro (Retina 13 Pulgadas, principios de 2015)

Procesador: 2,9 GHz Intel Core i5

Memoria: 8GB 1867 MHz DDR3

Gráficos: Intel Iris Graphics 6100 1536 MB

5.2.3 Software a Emplear

La siguiente tabla muestra todo el software que se va a emplear durante el proyecto.

| Software | Tipo | Finalidad |
|--------------|---------------------------------|---|
| Dropbox | Plataforma de almacenamiento | Almacenar toda la documentación y archivos utilizados durante el desarrollo del proyecto. |
| IntelliJ IDE | IDE de programación JAVA | Ayudar al desarrollo, depuración de errores y gestión del proyecto a realizar. |
| Pages | Herramienta de edición de texto | Realizar todos los documentos requeridos de cara al proyecto. |
| Apple Mail | Cliente de correo electrónico | Facilitar la comunicación entre el desarrollador y el director del proyecto. |
| GitHub | Gestor de Repositorios On-Line | Permitir la gestión del código en la nube así como sus diferentes versiones. |
| MagicDraw | Herramienta de diseño UML | Realización de todos los esquemas que aparezcan en la documentación. |
| Safari | Explorador Web | Navegación por la web, pruebas al plugin. |
| Postman | Herramienta de desarrollo | Realizar pruebas directamente de cara a las diferentes API a emplear. |

5.3 Descripción de Tareas

En este apartado se pretende desglosar las tareas que se deben llevar a cabo durante el proyecto con el fin de poder estimar con precisión el tiempo asignado a cada tarea. Para ello, se reutilizarán las fases mencionadas en el apartado 4. La enumeración de las tareas corresponde a la secuencia lógica en la que serán elaboradas.

5.3.1 Tarea 0: Estudio Previo

Tiempo estimado: 75 horas.

Durante esta tarea correspondiente a la asignatura de GEP se pretende realizar un análisis del contexto y la viabilidad del proyecto. Así como otros estudios necesarios. Su duración será de en torno a un mes y al no tener dependencia con ninguna otra tarea, se podrán realizar con otras tareas en paralelo.

Las subtareas serán las siguientes:

- Definición del Proyecto
- Estado del arte
- Planificación Temporal
- Gestión económica y de sostenibilidad
- Introducción al entorno de trabajo

5.3.2 Tarea 1: Análisis y Especificación de Requisitos

Tiempo estimado: 112,5 horas.

Esta tarea corresponde a la fase del apartado: 4.1.1 Análisis y Adaptación Conceptual de Replan a JIRA.

En esta tarea se pretende estudiar las diferencias y parecidos entre JIRA y Replan para ver qué casos de uso se pueden implementar como un plugin en JIRA y cuáles no, tratando de definir los requisitos iniciales así como los casos de uso que se realizarán durante el proyecto.

En esta tarea podemos distinguir algunas subtareas principales:

- Estudio de la API de Replan
- Estudio de la API de JIRA
- Aprendizaje de la usabilidad de JIRA
- Estudio del modelo conceptual de Replan
- Elaboración del modelo conceptual de JIRA
- Asociación de conceptos de Replan a conceptos de JIRA (y viceversa)
- Análisis de representación de conceptos de Replan en JIRA
- Definición de requisitos de ReactivePlan
- Definición de casos de uso de ReactivePlan.
- Lograr que esta tarea se vea bien reflejada en la documentación del proyecto.

Esta tarea no tiene dependencia con ninguna tarea previa.

5.3.3 Tarea 2: Desarrollo del Plugin en JIRA

Esta tarea corresponde a los apartados 4.1.2. y 4.1.3, y debido a esto tendrá dos etapas, una primera etapa de aprendizaje sobre el desarrollo de plugins en JIRA y una siguiente etapa que corresponderá a la integración del plugin de Replan en JIRA.

Primera Fase

Tiempo Estimado: 62,5 horas

Durante esta fase se pretende aprender sobre el desarrollo de plugins en JIRA, para ello se pretenden realizar diversos tutoriales así como realizar pruebas sobre la API con el fin de conocer los conceptos básicos sobre el desarrollo en esta plataforma.

Se pueden distinguir las siguientes subtareas:

- Puesta a punto de un proyecto de plugin de JIRA.
- Aprendizaje de los diversos componentes de un plugin en JIRA.
- Realización de los diversos tutoriales de desarrollo de plugins en JIRA.
- Pruebas sobre la API de JIRA, tanto como la API REST como la API Java.
- Lograr que esta fase se vea bien reflejada en la documentación del proyecto.

Esta fase no tiene ninguna dependencia con ninguna fase o tarea anterior.

Segunda Fase

Tiempo Estimado: 181,25 horas.

Durante esta fase se pretende desarrollar todo lo correspondiente al proyecto en sí, será la fase que conlleve más trabajo puesto que será aquella en la que haya que desarrollar todo el plugin completo.

Se pueden observar las siguientes subtareas:

- Puesta a punto del proyecto del plugin de JIRA.
- Puesta a punto de la herramienta Replan.
- Integración de Replan en JIRA.
- Elaboración de una interfaz gráfica para poder utilizar el plugin de Replan.
- Desarrollo de los casos de uso propuestos en las etapas anteriores.
- Lograr que esta fase se vea bien reflejada en la documentación del proyecto.

Esta será la fase más larga y compleja ya que se procederá a desarrollar el plugin totalmente, y depende tanto de la primera fase de la tarea 2 como de la tarea 1.

5.3.3 Tarea 3: Pruebas del Plugin y Documentación

Tiempo estimado: 18,75 horas.

Durante esta tarea se pretende probar todo aquello que se ha desarrollado previamente con el fin de verificar el correcto funcionamiento del plugin para dar por finalizado el proyecto.

También se pretende que estas pruebas ayuden a actualizar y verificar la documentación para asegurarse de que está completa. Será la fase final antes de dar por terminado el proyecto.

Se pueden distinguir las siguientes subtareas:

- Desarrollo de pruebas unitarias para el plugin
- Desarrollo de pruebas de integración para el plugin
- Establecer casos de Prueba de interfaz
- Corrección de bugs
- Revisión de la documentación

Esta tarea se podrá llevar a cabo en paralelo con las anteriores aunque no se podrá dar por finalizada hasta que estén las anteriores también finalizadas.

5.4 Diagrama de Gantt

Este diagrama de Gantt es orientativo y los plazos finales podrían variar. Tampoco se tendrán en cuenta las subtareas para la planificación puesto que no se pueden estimar con demasiada precisión cuánto tiempo requerirá cada una.

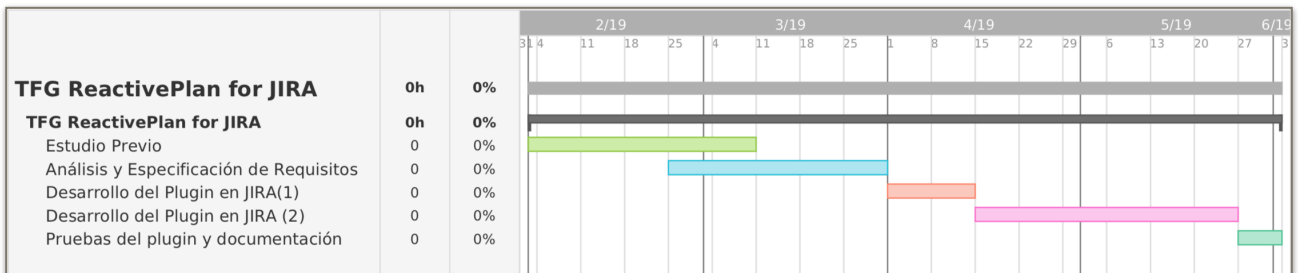


FIG 4. DIAGRAMA DE GANTT REPRESENTANDO LA DISTRIBUCIÓN DE TIEMPO DE LAS TAREAS CARA AL PROYECTO.

Se puede observar en el diagrama que el Estudio Previo y el Análisis y la Especificación de Requisitos se realizarán en paralelo para dar el proyecto por finalizado en el plazo estimado.

5.5 Valoración de alternativas y plan de acción

Puesto que durante la realización de este proyecto pudiera surgir algún imprevisto se va a realizar una propuesta de plan de acción con el fin de poder dar por finalizado el proyecto en la fecha estimada. Para ello, se van a valorar diversas situaciones que puedan ocurrir y cómo reaccionar ante ellas. Puesto que la metodología que se va a emplear de cara al desarrollo es ágil se podrán reaccionar rápidamente ante estos cambios.

5.5.1 Caso 1: No da tiempo a implementar algún requisito

Por cada requisito, se realizará una estimación del tiempo que va a tardar en implementarse. En el caso de que ese tiempo se exceda, se dará una prórroga de, como máximo, **una semana** en la que lo primero que se hará será reevaluar el tiempo estimado así como si hay que realizar una modificación sobre este requisito afín de que se pueda implementar en el tiempo establecido.

En el caso de que se estime que el requisito en cuestión no se va a poder implementar, se descartará totalmente.

5.5.2 Caso 2: Baja por enfermedad

En el caso de que el desarrollador no pueda trabajar en el proyecto debido a una enfermedad, **éste deberá suplir las horas no trabajadas por día durante los fines de semana**, los cuales no se han contemplado de cara a realizar el proyecto.

Si los fines de semana no fuesen suficientes debido a una baja demasiado larga, tras la recuperación se establecerá **un horario más amplio cada día**.

5.5.3 Caso 3: Falta o no disponibilidad de recursos

En este caso se pactará con el director del proyecto qué alternativas hay respecto a los recursos.

En el caso de que sean recursos que se puedan obtener rápidamente, se pospondrá el proyecto durante un máximo de **dos días** para conseguirlos.

5.5.4 Caso 4: Problema no valorado

En el caso de que surja un problema no valorado en este apartado, se tratará de concertar una reunión con el director del proyecto tratando de buscar una solución que afecte lo menos posible a la duración del proyecto. Pese a esto, no se puede calcular cuánto ni cómo afectará la proyecto.

5.6 Desviaciones respecto a la planificación inicial

En este apartado se procederá a medida avanza el proyecto a mostrar las desviaciones respecto a la planificación inicial, explicando el por qué de su aparición así como las diferentes repercusiones que éstas hayan podido tener en el proyecto tanto a nivel económico como de implementación.

5.6.1 Desviación 1: Prolongación de la primera fase de Desarrollo.

Debido a la elevada complejidad de **JIRA**, así como su *extensa documentación* (ver apartado de Obstáculos), comenzar la **Primera Fase de Desarrollo**, resultó más complejo de lo esperado y ha desembocado en una desviación de la planificación inicial, aumentando el tiempo requerido para comprender la complejidad de esta herramienta así como el desarrollo de Plugins para la misma.

Esta desviación, se ha producido debido a la valoración de distintas **Tecnologías** para implementar el Plugin de la forma más correcta y eficaz posible (ver apartado de Análisis de Alternativas), así como la dificultad para aprender a desarrollar plugins debido a la escasa documentación de cara al desarrollo de plugins respecto a la extensa documentación de la propia herramienta.

5.6.1.1 Desviación 1: Repercusión Temporal

Debido a lo mencionado anteriormente, esta fase ha requerido el doble de tiempo, siendo **125 horas** en lugar de **62,5**, atrasando la planificación temporal **15 días**.

No se procederá a realizar ningún nuevo *Diagrama de Gantt* respecto a la planificación, puesto que lo único que varía es que esta fase terminó el **1 de Mayo**, retrasando la planificación temporal en 15 días. Es decir, en lugar de terminar el proyecto el **28 de Mayo**, se estima que termine en torno al **15 de Junio**.

5.6.1.2 Desviación 1: Repercusión Económica

Debido a la duplicación de horas en esta fase, se deberán pagar 62,5 horas extra del **Programador Java**, resultando en **750€** en total, como se habían previsto **500€** para gastos por tiempo extra, el coste del proyecto aumentará en un total **250€** (750-500).

5.6.1.3 Desviación 1: Repercusión de Objetivos

Los objetivos del proyecto no se verán afectados ya que se elaboró el plan **hasta con un mes de margen** respecto a la entrega final, elaborando un plan temporalmente flexible que se pudiese cumplir.

6. Análisis de Sostenibilidad

En esta sección se procederá a realizar un análisis de la sostenibilidad del proyecto. Para ello se tratará de analizar el aspecto económico, social y medioambiental de este proyecto basando todas las conclusiones y reflexiones en la denominada *matriz de sostenibilidad* [9].

6.1 Introducción

A pesar de ser el análisis de sostenibilidad una tarea que se debe realizar ante cualquier proyecto software como este, resulta una tarea compleja puesto que, además de no ser un tema especialmente tratado a la hora de estudiar un grado técnico como es el caso de la *Ingeniería del Software*, se basa en gran parte en la reflexión personal así como en conceptos no tratados en las carreras técnicas de las que cada autor de un proyecto deberá informarse a través de fuentes externas en su mayoría.

6.1.1 Cuestionario de Estudiantes de Ingeniería Informática

Con el fin de poder realizar una autoevaluación sobre los conocimientos de sostenibilidad, se ha propuesto la realización de una encuesta para los estudiantes de ingeniería informática que estén realizando proyectos de fin de grado de tal forma que estos puedan reflexionar sobre sus conocimientos.

La sostenibilidad de un proyecto, ya sea económica social o medioambiental es algo que nos afecta a todos directa o indirectamente. Tener estos factores en cuenta a la hora de realizar un proyecto tecnológico como es el de este caso, claramente contribuirá a desarrollar la sociedad tal y como está. Desafortunadamente, en las carreras *TIC*, la sostenibilidad no es una materia de estudio y los conocimientos adquiridos por el estudiante a lo largo de estas son o muy básicos o, en ciertos aspectos de éstas, desconocidos a pesar de la verdadera importancia que tienen.

6.2 Gestión Económica

Este apartado está orientado a realizar un análisis de los costes, presupuestos y todo aquello relacionado con el aspecto económico del proyecto.

6.2.1 Introducción de los Costes

En este apartado se procederá a analizar todos los costes del proyecto, basándonos en los recursos mencionados en apartados anteriores junto con nuevos costes a contemplar que no son intrínsecos al proyecto.

6.2.2 Costes Indirectos

Estos costes corresponden a aquellos que no repercutirán de forma directa a la producción del proyecto.

El transporte será la T-Jove de Zona 1 de la ciudad de Barcelona, cuyo coste total es de 105€ por tres meses de duración.

| Concepto | Coste por Unidad [€] | Unidad | Unidades Aproximadas |
|-----------------------|----------------------|---------------|----------------------|
| Transporte | 33 | Meses | 4,5 |
| Conexión a Internet | 15 | Meses | 4,5 |
| Luz | 15 | Meses | 4,5 |
| Impresión del Trabajo | 0,01 | Folio impreso | 2.000 |
| Total | 303,5 € | | |

6.2.3 Costes Directos por actividad

Estos costes influyen directamente en la producción del proyecto, de modo que se calculará el coste por hora. En el caso de este proyecto serán exclusivamente recursos humanos. Para poder realizar una estimación del coste con precisión, se procederá a desglosar el trabajo del desarrollador del proyecto en los diversos roles que este va a ejercer a lo largo del proyecto.

Para calcular estos costes se utilizará como referencia el salario promedio en la ciudad de Barcelona de cada rol [10].

| Rol | Tarea | Salario [€/h] |
|---------------------------|---|---------------|
| Analista Funcional | Obtener los requisitos, CU y documentar el proyecto. | 15 |
| Programador JAVA | Programar cada caso de uso referente al proyecto. | 12 |
| Gestor de Proyecto | Lograr que el proyecto finalice a tiempo así como realizar la documentación correspondiente al mismo. | 25 |
| Total | 7800 € | |

Para elaborar la estimación sobre el **costo total** de los costes directos se tendrán en cuenta las 450 horas mencionadas previamente en la estimación del tiempo y se asumirá que todos los roles trabajarán por igual, realizando la siguiente media aritmética:

$$\sum (\text{Sueldo de Cada Rol} * 450) / 3$$

Dando lugar al **costo total** de **7800 €**.

6.2.4 Imprevistos

Con tal de ofrecer el análisis de imprevistos más completo posible se procederán a analizar los diferentes imprevistos que podrían afectar la economía durante el desarrollo del proyecto.

- **Avería del ordenador:** En el caso de que el ordenador con el que se está trabajando el proyecto se averíe se reservarán unos **200€** para arreglarlo, o, en el caso de que este no se

pueda reparar, obtener un ordenador básico para poder finalizar el proyecto. Esto tiene una probabilidad del 1% de que ocurra.

- **Duración extra del proyecto:** Por si el proyecto no pudiese finalizarse en la fecha estimada, se tendrán **500€** de margen (para recursos humanos y un mes más de gastos mensuales), ofreciendo hasta **dos semanas más de posible trabajo extra**. Esto tiene una probabilidad del 30% de que ocurra.
- **Material Extra:** Para el caso de que el proyecto requiera algún tipo de material que inicialmente no se ha tenido en cuenta (servidores dedicados, el uso de material de oficina etc...) se tendrán **50€** de margen para obtener estos recursos. Esto tiene una probabilidad del 20% de que ocurra.

| Concepto | Coste [€] |
|-----------------------------|--------------|
| Avería del ordenador | 200 |
| Duración extra del proyecto | 500 |
| Material Extra | 50 |
| Total | 750 € |

6.2.5 Contingencia

Se reservará una parte del presupuesto para la contingencia del proyecto. El valor de esta parte será de un 10% de la suma de los costes directos e indirectos, es decir una cantidad total de **810,35 €**.

6.2.6 Costo Total

Una vez se han analizado todos los aspectos principales sobre la gestión económica del proyecto se puede estimar el costo total de acuerdo a la siguiente tabla.

| Concepto | Coste [€] |
|-------------------|----------------|
| Costes Directos | 7800 |
| Costes Indirectos | 303,5 |
| Imprevistos | 750 |
| Contingencia | 810,35 |
| Total | 9360,85 |

6.2.7 Control de Gestión

El control de la parte económica del proyecto se llevará a cabo apuntando los costes reales que está teniendo el proyecto a medida que va avanzando y realizando el incremento de éstos.

Por otro lado, se apuntarán el número de horas empleadas en cada fase y tarea con el fin de poder realizar también el control de las horas empleadas para poder ofrecer un costo final lo más ajustado posible.

Los incrementos serán triviales, siendo todos los incrementos de la siguiente forma:

(Valor estimado - valor real del coste).

De este modo se podrán calcular sencillos incrementos y ver cuál ha sido el coste final del proyecto. Para eso usaremos la siguiente tabla:

| Fase | Horas Estimadas | Horas Reales | ΔHoras | Coste Estimado [€] | Coste Real [€] | Rol Trabajado |
|---|-----------------|--------------|--------------|--------------------|----------------|--------------------|
| Estudio Previo | 75 | 90 | -15 | 1.875 | 2.250 | Gestor de Proyecto |
| Análisis y Especificación de Requisitos | 112,5 | 70 | 42,5 | 1687,5 | 1.050 | Analista Funcional |
| Desarrollo del Plugin en Jira (1) | 62,5 | 62,5 | 0 | 750 | 750 | Programador JAVA |
| Desarrollo del Plugin en Jira (2) | 181,25 | 200 | -18,75 | 2.175 | 2.400 | Programador JAVA |
| Pruebas del Plugin y Documentación | 18,75 | 59 | -40,25 | 218,25 | 885 | Analista Funcional |
| Total | 450 | 481,5 | -31,5 | 6.705,75 | 7335 | |

6.2.8 Análisis del Impacto Económico

Con el fin de realizar un análisis lo más completo posible la gestión económica de este proyecto, también se tendrá en cuenta el impacto económico del mismo.

Reflexión sobre el coste total

El coste total del proyecto puede resultar algo elevado pero esto se debe a la envergadura que tiene y, al tiempo requerido y necesitado por parte de los recursos humanos del mismo, ya que, al ser una tarea tan compleja como la implementación de un plugin. A pesar de esto, no se requieren muchos materiales ni recursos para lograr el desarrollo satisfactorio del mismo abaratando mucho los costes.

Dicho de otro modo, para que este proyecto vaya a delante, en el caso de que ya se tenga el material básico (un ordenador, luz y conexión a internet), sólo se deberían tener en cuenta los recursos humanos cualificados para llevarlo a cabo resultando en una puesta en producción bastante sencilla así a nivel económico como material.

Vida útil

Como se menciona en el estado del arte de este proyecto, será totalmente satisfactorio hasta que aparezca una empresa que desarrolle el mismo software mejor que éste. En el caso de que se inviertan suficientes recursos económicos y tiempo en el desarrollo de un proyecto como la introducción de este plugin para JIRA se podría hacer competencia a nivel global resultando en grandes beneficios económicos.

Mejora Económica

Este proyecto además de ofrecer beneficio económico al desarrollador del mismo, permitirá una mejora económica para las empresas ofreciendo menos carga de trabajo a los actuales gestores de proyectos y pudiendo enfocar a estos en otras tareas solamente a través de un plugin. Respecto a otros proyectos de la misma envergadura, no se ha podido ofrecer un costo más barato puesto que lo más costoso del proyecto han sido los recursos humanos pero no se puede escatimar en estos, ya que son los que llevarán el proyecto hacia adelante.

6.3 Análisis del impacto Medioambiental

En esta sección se procederá a analizar el impacto medioambiental de este proyecto.

6.3.1 Impacto ambiental del proyecto

En el caso de este proyecto, el único impacto ambiental que tendrá es el del uso de dispositivos electrónicos para la realización del mismo así como los folios que se impriman sobre la documentación de éste.

6.3.2 Minimización del impacto ambiental

Como se ha mencionado anteriormente, el impacto ambiental será muy reducido. De cualquier modo, para reducir este posible impacto se tendrá en cuenta el consumo energético tratando de no realizar tareas innecesarias que consuman electricidad de más así como el correcto cuidado de los dispositivos electrónicos a utilizar con el fin de que duren el mayor tiempo posible.

6.3.3 Vida útil del proyecto a nivel ambiental

Este proyecto no pretende obtener ningún tipo de vida útil a nivel ambiental ni lo va a lograr.

6.4 Análisis del impacto social

En este apartado se procederá a evaluar el impacto social de la creación de este proyecto.

6.4.1 Aportación Personal del desarrollo del proyecto

A nivel personal, la puesta en producción de este proyecto contribuirá a la mejora de la autogestión, así como el aprendizaje de cómo desarrollar un proyecto sin un equipo. La capacidad de desenvolverse en situaciones complejas sin tener ningún tipo de ayuda es de los aprendizajes más relevantes en el proyecto.

A nivel técnico, el aprendizaje de nuevas tecnologías y documentación son factores importantes sobre el desarrollo personal realizado durante este proyecto.

6.4.2 Resolución actual del problema y mejora social

Actualmente, la gestión de proyectos se lleva a cabo a través de herramientas básicas de gestión de proyectos, resultando en que los gestores de proyectos en lugar de tener tiempo para desglosar tareas y ayudar en las tareas de equipo tengan que enfocarse en realizar diversos planes con el fin de que el trabajo en equipo se desarrolle de la mejor forma posible. La principal mejora social de este proyecto es la liberación de tiempo del gestor de proyectos pudiendo enfocarlos en otras tareas así como una mayor agilización de los equipos de desarrollo, en general.

6.4.3 Necesidad del Proyecto

Como en cualquier producto tecnológico (y más aún de software), antes de crearse no existe una necesidad más allá de la de mejorar nuestras vidas, y, en este caso ahorrar tiempo al gestor de proyectos a la hora de realizar su trabajo y abaratar costos para las diversas empresas, así como ayudar a pequeños equipo.

La necesidad básica y trivial de este proyecto se resume a cualquier necesidad tecnológica que es: facilitar nuestras vidas.

"Muchas veces la gente no sabe lo que quiere hasta que se lo enseñas." (Steve Jobs)

7. Análisis de Alternativas

En este apartado se procederá a valorar las diferentes alternativas que se han ido teniendo en cuenta de cara a la correcta realización del proyecto así como aquellas que finalmente se han escogido de cara a desarrollar **ReactivePlan for Jira**.

7.1 Herramienta Replan

Tal y como se menciona en el punto 2.2.1 del Análisis del Contexto de este documento, para integrar **JIRA** con la herramienta de Replan, se dudaba entre emplear dos capas, del Back-End de esta herramienta ya que, aunque la planificación final acabase resultando la misma, cada una tenía sus beneficios e inconvenientes. En este apartado se procederá a valorar ambas herramientas de cara al proyecto y por qué se ha escogido la herramienta final.

7.1.1 Replan Controller

Este controlador de **Replan** se encarga de generar el modelo y las entidades de **Replan**, así como asegurar su **persistencia** de cara a desarrollar una planificación a través del **replan optimizer**. Cuenta con una **API Rest** para comunicarse con éste, además de contar con sesiones para gestionar correctamente los datos y evitar que se corrompan.

Esta opción resulta demasiado compleja ya que en el caso de **Replan for Jira**, no se necesita persistir datos, ya que esto lo hace **Jira**, además de que cuenta con demasiados métodos [11] que podrían dificultar un **desarrollo simple** y producir un mayor **acoplamiento** entre la herramienta desarrollada durante el proyecto y **Replan**.

7.1.2 Replan Optimizer

El optimizador de **Replan** es la herramienta encargada de computar el *algoritmo genético* y enviar como respuesta una planificación concreta dado un conjunto de **Recursos e Incidencias**.

Esta herramienta cuenta con una sola llamada [12] la cual se encarga de gestionar el modelo enviado en la petición y devolver una planificación en el *formato de Replan*. Normalmente se accede a esta herramienta a través de su Controlador pero no es estrictamente necesario ya que se puede trabajar con ella.

El mayor inconveniente de utilizar esta herramienta es que, debido al uso de un *algoritmo genético*, y tener que convertir todos los parámetros de **JIRA** en el formato de esta herramienta requieren un tiempo computacional algo más elevado respecto al controlador, debido a la falta de persistencia.

7.1.3 Comparación Replan Controller frente a Replan Optimizer

| | Replan Optimizer | Replan Controller |
|------------------------|------------------|-------------------|
| Persistencia | ✗ | ✓ |
| Desarrollo Simple | ✓ | ✗ |
| Documentación Sencilla | ✓ | ✗ |

7.1.4 Herramienta Escogida

Teniendo en cuenta el alcance que se ha definido en el proyecto y que, lo que se pretende es desarrollar un Plugin de **JIRA** que sea capaz de planificar cómodamente un proyecto cualquiera se ha optado por conectar **JIRA** con el **Replan Optimizer**, ya que al tener éste una única llamada, es más sencillo realizar un *back-end* en el plugin para que se conecte a este Optimizer, traduciendo los parámetros para que sean entendibles por el **Optimizer**, y así, reduciendo el desacople entre **JIRA y Replan**, convirtiéndolo en un proyecto más fácilmente mantenible y reescalable. Siendo este plugin, entre otras cosas, **un controlador para el optimizer** que utiliza como persistencia los datos contenidos en **JIRA**.

7.2 Integración en JIRA

Con el fin de integrar la herramienta de **Replan** seleccionada en **JIRA**, primero debemos saber que de cara al desarrollo de plugins, todas las plataformas de **JIRA** funcionan de forma similar, de modo que no se diferenciarán las plataformas, sino el modo de conectarse a ellas.

7.2.1 JIRA Rest API

Afín de obtener los datos necesarios para poder realizar estas consultas y actualizarlos, JIRA cuenta con una **API Rest** [13] a la cuál se puede acceder como a cualquier otra API Rest, a través de un token de sesión.

Inicialmente, durante la **primera fase de desarrollo del proyecto**, se trató de emplear esta API para realizar las conexiones al propio servidor a través de una librería llamada **Rest Java Client for Jira** [14], librería la cuál se encargaría de realizar todas estas peticiones, ya que, inicialmente y, a simple vista parecía la opción más sencilla de implementar debido a que el manejo de una API Rest resulta más sencillo que comprender la compleja documentación de JIRA.

Finalmente, esta opción se descartó puesto que, además de ser realmente complejo realizar el inicio de sesión ya que se accedía desde el propio JIRA en lugar de desde una aplicación externa, aumentando la dificultad en la lógica de cada transacción. Por otro lado, también cabe destacar el hecho de que la librería que resultaba ser el conector en Java estaba desactualizada.

Sin embargo, es la mejor opción a considerar si se desea hacer un Plugin externo a JIRA.

7.2.2 Atlassian JIRA - Server 8.1.0 API

Estas librerías [15] son aquellas desarrolladas por **Atlassian** incluidas dentro de cualquier Plugin de JIRA (Atlassian SDK) , las cuales resultan especialmente complejas de utilizar debido al gran número de clases *irrelevantes* con las que se cuentan de cara a lo que se pretende hacer en **Replan**.

Inicialmente, estas librerías se descartaron debido a su elevada complejidad de cara al uso, sin tener una idea preconcebida sobre el desarrollo de la plataforma, así como la inexistencia de tutoriales para aprender a usar esta librería.

Más tarde, al ver que era la única opción viable de cara a realizar consultas complejas así como la obtención y manipulación de elementos claves como los **Issues**, se optó por realizar consultas en el **foro de desarrolladores de JIRA** [16] de cara al aprendizaje del desarrollo de plugins en esta plataforma lo cuál ha permitido el desarrollo de **ReactivePlan for JIRA**.

8. Representación de una planificación en JIRA

8.1 Conceptos

Con el fin de mostrar qué forma una planificación en JIRA se procederán a definir los diversos conceptos clave que dan lugar a esta planificación.

Debido a que la mayoría de la documentación de la herramienta JIRA se encuentra en inglés, no se realizarán las traducciones al castellano para evitar posibles errores de traducción e interpretación del sistema.

Todos los conceptos que aparecen en este apartados están redactados de acuerdo a la *Guía de Usuario de JIRA* [18].

8.1.1 Project

Un project en JIRA es una colección de *issues*, y se definirá de acuerdo a los requisitos de la organización.

Un project, además de *issues* podrán tener *versions* y *components*.

8.1.1.1 Version

Para algunos tipos de projects, particularmente aquellos de desarrollo de software, es útil poder asociar un *issue* con una *version* en particular.

Los issues tienen los campos que referencias a las versiones:

- **Affects Version(s)** — Esta es la versión/es a la que el issue en concreto afecta.
- **Fix Version(s)** — Esta es la versión/es donde el issue será resuelto.

Las versiones pueden estar en tres diferentes estados: **Released**, **Unreleased** o **Archived**. Las versiones también pueden tener una *Release Date*, y automáticamente se marcará como 'sobrepasada' si la versión permanece *Unreleased* tras esta fecha.

8.1.1.2 Component

Un componente de un proyecto es una agrupación lógica de issues a lo largo de un project. Cada proyecto podría tener varios o ningún componente, de acuerdo con las necesidades de la organización.

8.1.2 Issue

Las diferentes organizaciones utilizan JIRA para monitorizar diferentes tipos de issues. Dependiendo de cómo tu organización emplee JIRA un *issue* podría representar desde un error en el sistema, una tarea de un proyecto, una solicitud al servicio etc.

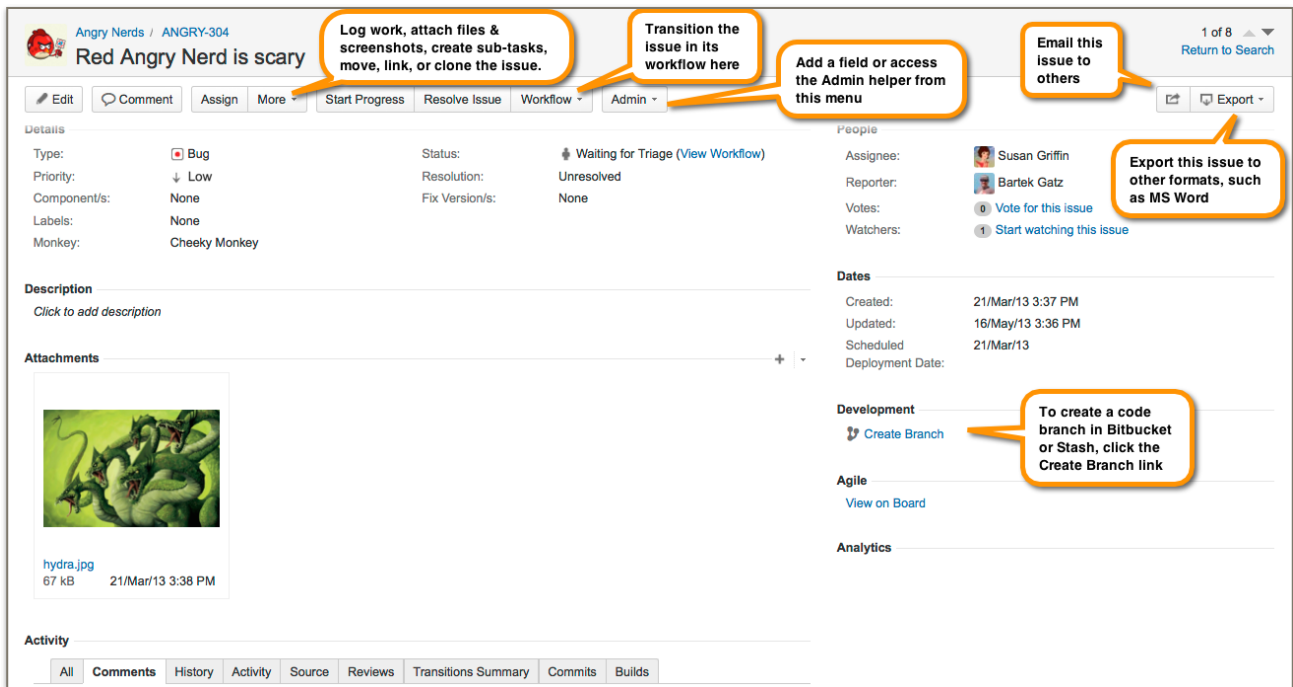


FIG 5. VISUALIZACIÓN DE UN ISSUE [18]

En la imagen superior podemos ver cómo se visualiza típicamente un issue en JIRA. Los campos mostrados en la imagen anterior serían los siguientes:

| Campo | Descripción |
|--------------------------|--|
| Project | El proyecto al que pertenece el issue. En este caso es "Angry Nerds". |
| Key | Es un identificador único para el issue en concreto. En este caso es ANGRY-304, las letras de la izquierda hacen referencia a la Key del proyecto. |
| Summary | Un breve resumen de una línea. |
| Type | El tipo de issue que es. |
| Status | El estado en el que se encuentra el issue respecto a su ciclo de vida. |
| Priority | La importancia del issue respecto a los otros existentes en el proyecto. |
| Resolution | El registro de la resolución del issue. Determina si el issue ha sido resuelto o no. |
| Affect Version(s) | Versiones del proyecto para las que el issue se manifiesta. |
| Fix Version(s) | Versiones del proyecto para las que el issue será resuelto. |
| Component(s) | Componentes del proyecto a los que el issue se refiere. |
| Label(s) | Campos a las que el issue se refiere. |
| Environment | Entorno ya sea de software o de hardware al que el issue se refiere. |
| Description | Una descripción detallada del issue. |
| Links | Conjunto de enlaces a otros issues. |
| Assignee | La persona que está actualmente asignada al issue. |
| Reporter | La persona que añadió el issue al sistema. |

| Campo | Descripción |
|-----------|--|
| Votes | Muestra el número de votos que tiene el issue. |
| Watchers | Muestra el número de personas que están observando el issue. |
| Due | La fecha en la que el issue debe ser finalizado. |
| Created | La fecha en la que se introdujo el issue en JIRA. |
| Updated | La fecha de la última edición del issue. |
| Resolved | La fecha en la que el issue fue resuelto (si lo está). |
| Estimate | La estimación original del tiempo total requerido para resolver el issue. |
| Remaining | El tiempo estimado restante , es decir el tiempo que queda estimado actualmente para resolver el issue. |

8.1.2.1 Issue Type

JIRA puede emplearse para realizar el seguimiento de muchos tipos de issues diferentes. Los issues por defectos están listados a continuación pero se tendrá en cuenta que el *administrador de JIRA* podrá crear tipos *personalizados* para que encajen en la organización.



Bug — Un problema que afecta, daña o impide la correcta funcionalidad del producto.



Improvement — Una mejora a un feature existente.



New Feature — Una nueva funcionalidad del producto.



Task — Una tarea que necesita ser realizada.



Custom Issue — Un tipo de issue personalizado.

8.1.2.2 Priority

La prioridad de un issue indica su importancia relativa. Las prioridades por defecto están listadas a continuación aunque se podrán añadir prioridades personalizadas para que convenga a la organización

Highest — La prioridad más alta. Este issue podría bloquear el progreso.

High — Indica que un issue tiene gran importancia. Deberá resolverse lo antes posible.

Medium — Indica que el issue tiene un impacto significativo.

Low — Indica que el issue tiene un impacto relativo menor.

Lowest — La prioridad más baja.

8.1.2.3 Resolution

Un issue puede ser resuelto de muchas maneras, sólo una de ellas será la de 'Resuelto'. Una resolución normalmente se establece cuando su estatus cambia. Se procederá a listar las resoluciones por defecto.

Fixed — Una solución para este issue ha sido implementada.

Won't Fix — Este issue no se va a solucionar.

Duplicate — El issue está duplicado.

Incomplete — No hay suficiente información para trabajar en este issue.

Cannot Reproduce — No se puede resolver en este momento.

Won't Do — El issue no se va a resolver.

8.2 Planificación

Para realizar una planificación, JIRA permite la creación de tableros a través de su *Plugin* de JIRA *Agile*.

Al haber empleado en este proyecto exclusivamente JIRA Core (afín de que sea maximice su compatibilidad con cualquier organización), se procederá a explicar cómo se muestra una planificación en JIRA Core.

A nivel básico, y teniendo en cuenta todo lo explicado anteriormente. De acuerdo a JIRA Core una planificación no será más que establecer un **Assignee** y **Due date** a un conjunto de issues en concreto. Aquellos que cumplan estas características serán dados por planificados. En el caso de que queramos organizar nuestros issues por sprint se deberá añadir el plugin de **JIRA Agile** el cual no entra en el desarrollo de este proyecto. [19]

9. Representación de una planificación en Replan Optimizer

Para poder aclarar cómo funciona y qué hace la herramienta de Replan Optimizer, es necesario primero, definir todos y cada uno de los conceptos que se emplean. Puesto que es una herramienta a la que se accede a través de una API Rest, no se dispone de una interfaz gráfica para mostrar cómo es una planificación, de modo que se procederá a describir cada uno de los elementos que tiene en cuenta una planificación en el Replan Optimizer.

9.1 Resource

Representa a los empleados. Cada resource tendrá asignado un calendar así como una serie de skills.

9.2 Feature

Representa las diversas tareas a completar. Cada feature requerirá una serie de skills, así como que tendrá una prioridad en concreto de cara a realizar la tarea. También tendrá una duración en concreto medida en horas.

Un feature a su vez, podrá depender de otro feature lo cuál quiere decir que hasta que el feature del que depende no se haya finalizado, éste no podrá iniciarse.

9.3 Calendar

Representa al conjunto de jornadas disponibles de un resource. El calendario puede tener diversos status, así como una hora de inicio, hora de final, semana, y día de la semana.

Los posibles status de un calendar serán los siguientes:

- **Free:** La jornada está libre y el resource en cuestión no tiene ninguna tarea asignada.
- **Used:** La jornada se ha utilizado para realizar una planificación y contiene un feature en cuestión.
- **Frozen:** El resource tenía una feature previamente asignada en este slot, de modo que nos e puede emplear.

9.4 Skills

Representan el conjunto de habilidades que puede tener el resource o feature en concreto. Su único valor es su nombre.

9.5 PriorityLevel

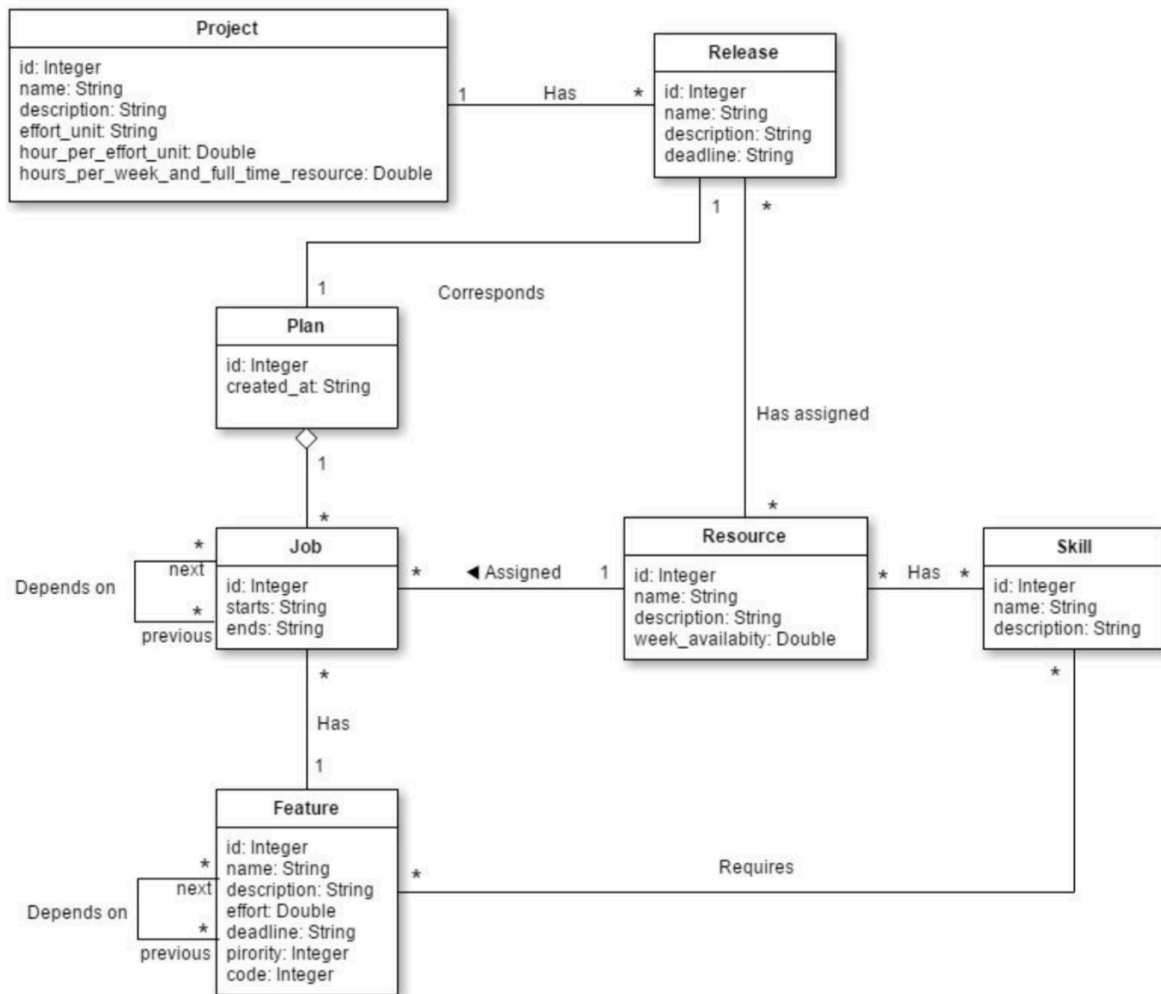
Representa el nivel de prioridad de una feature en concreto.

10. Comparación Conceptual JIRA - Replan

Una vez se han definido los conceptos básicos de qué es una planificación en cada una de las dos herramientas. Se procederá a realizar una comparación a nivel conceptual de ambas herramientas así como definir las diferentes transformaciones de datos que serán necesarias afín del correcto funcionamiento de la plataforma.

Para facilitar esta tarea, se hará uso de modelos conceptuales en formato UML.

10.1 Modelo Conceptual de Replan



RI:

- 1) Un Resource no pot estar assignat a un Job que tingui una Feature que requereixi unes Skills que el Resource no té
- 2) Un Resource només pot estar assignat a Jobs que formin part d'un Plan corresponent a una Release a la qual el Resource està assignat
- 3) start < deadline <= ends
- 4) Un Job no pot dependre d'ell mateix
- 5) Una Feature no pot dependre d'ella mateixa

FIG 6. MODELO CONCEPTUAL DE REPLAN [5]

Como se puede apreciar a lo largo de este modelo conceptual, la herramienta Replan es más extensa de lo que se puede emplear de cara al Replan Optimizer. De este modelo se debe sacar en claro el concepto de **Project** así como el de **Release**, puesto que serán los más importantes de cara a realizar la comparación conceptual en los siguientes apartados.

Como se puede apreciar, un **Project** tendrá un conjunto de **Releases** las cuales planificarán los diferentes **Features** que se mencionan previamente en el *optimizer*.

10.2 Modelo Conceptual de Replan Optimizer

Debido a que a Replan Optimizer se conectará a través de una API Rest, se ha procedido a diseñar dos modelos conceptuales, uno para la petición a la API y otro para la respuesta.

10.2.1 Modelo Conceptual Request

Este sería el modelo que se debe enviar con el fin de solicitar una planificación a la API. Como se puede apreciar, de cara a enviar esta solicitud no es necesario establecer ningún parámetro relacionado con el **Proyecto** o la **Versión** con la que queremos trabajar.

Este modelo ha sido desarrollado a través de la documentación de la API de Replan Optimizer.

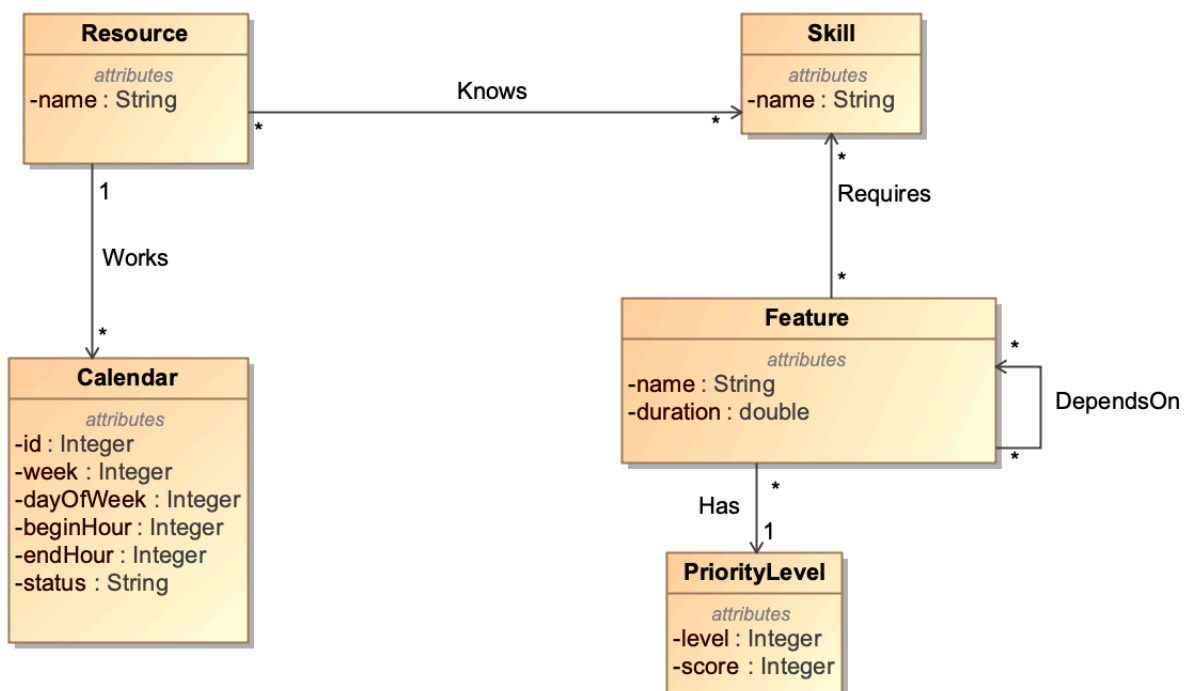


FIG 7. MODELO CONCEPTUAL DE REPLAN OPTIMIZER REQUEST

10.2.2 Modelo Conceptual Solution

Como podemos observar, aparece el elemento **PlanningSolution**, el cual nos muestra unos parámetros sobre cuál es la calidad de la solución, además de que las relaciones entre los diferentes elementos se ven modificadas.

Este modelo se ha desarrollado a través de la documentación de la API de Replan Optimizer.

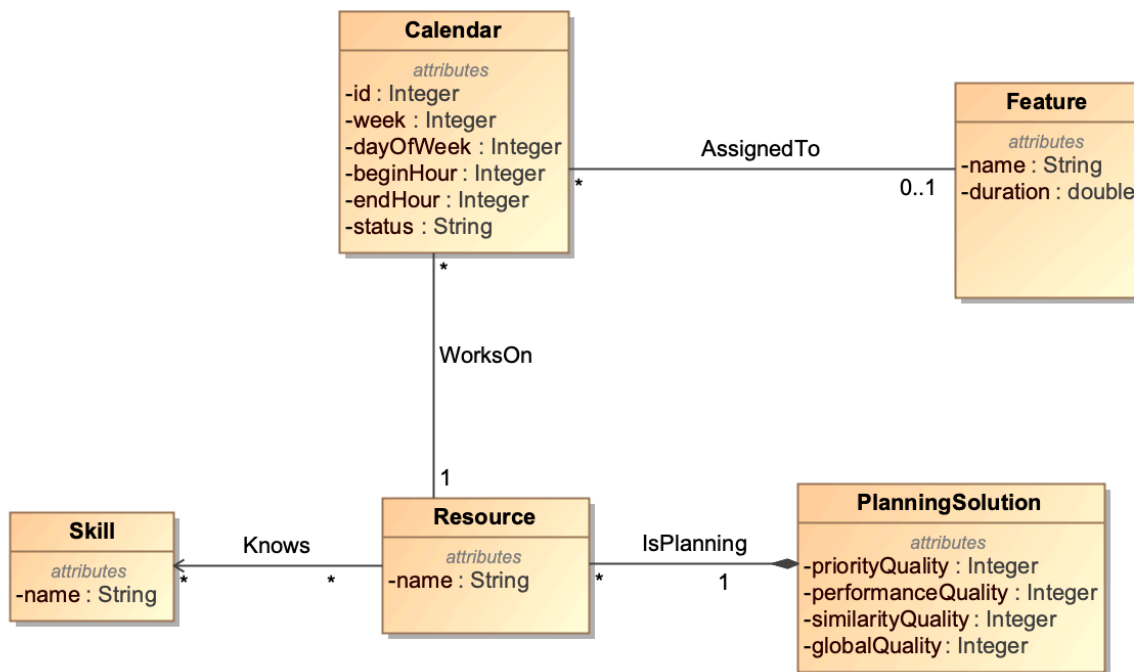


FIG 8. MODELO CONCEPTUAL DE REPLAN OPTIMIZER SOLUTION

10.3 Modelo Conceptual de JIRA

Debido a que JIRA es una herramienta muy extensa y con una amplia cantidad de clases así como funcionalidades, se ha procedido a elaborar un esquema conceptual que agrupe solamente los conceptos relevantes para realizar una comparación lo más completa posible.

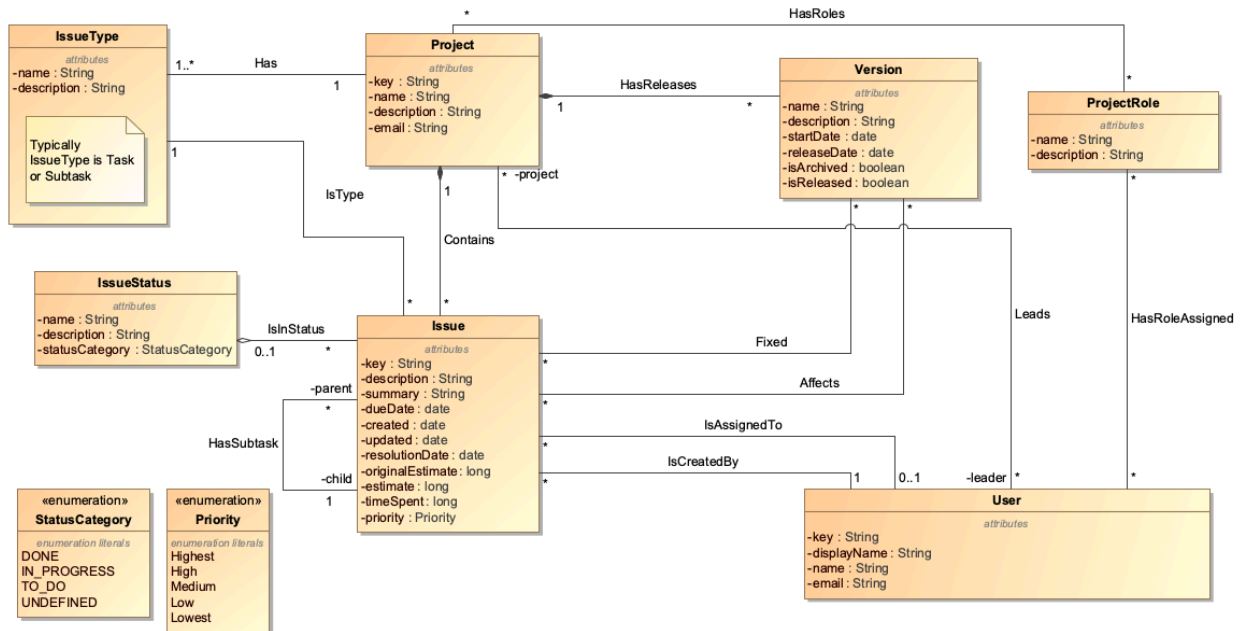


FIG 9. MODELO CONCEPTUAL DE JIRA

Como se puede apreciar a lo largo del modelo conceptual, JIRA es significativamente más complejo que REPLAN de modo que, se deberá adaptar toda la información que éste posee afín de que la conversión funcione correctamente.

Este modelo se ha desarrollado a través de la documentación de JIRA y se ha ido ampliando a lo largo de todo el proyecto con el fin de que se ajuste en lo máximo posible a los requisitos que se definen.

10.4 Comparación de Elementos

En este apartado se procederá a hacer una comparación entre elementos, tratando de establecer una relación entre los elementos de REPLAN y JIRA que aparecen en los diversos esquemas conceptuales.

La siguiente tabla muestra la relación entre elementos.

| JIRA | REPLAN | Descripción |
|----------------|----------------|---|
| Project | Project | Es el proyecto que engloba diversas funcionalidades a implementar. |
| Version | Release | Conjunto de funcionalidades dentro de un proyecto que serán publicadas en un momento determinado. |
| Issue | Feature | Funcionalidad en concreto a publicar. |

| JIRA | REPLAN | Descripción |
|--------------------|-----------------|---|
| ProjectRole | Skill | Habilidades que tiene cada usuario a lo largo de un proyecto. |
| User | Resource | Elemento que trabaja en una funcionalidad en concreta. |
| - | Calendar | Períodos en los que un resource está o no libre. También se le puede asignar una feature. |

10.5 Comparación de atributos

En este apartado se procederá a realizar una comparación de atributos por cada elemento con el fin de poder realizar las transformaciones pertinentes para lograr el correcto funcionamiento del plugin. Los atributos que no se vean reflejados en las tablas comparativas se asumirá que es debido a que no existe un equivalente en la plataforma o no son necesarios de cara al desarrollo del proyecto.

Para realizar esta comparación de atributos, en el caso de que existan enumeraciones se procederá a emplear únicamente los valores por defecto de JIRA.

10.5.1 Project - Project

| Atributo JIRA | Atributo Replan | Descripción | Transformación de Unidades |
|---------------|-----------------|----------------------------------|---|
| Key | ID | Es la clave única del proyecto. | Hay que convertir la Key de String a Integer . |
| Name | Name | Es el nombre del proyecto. | No es necesario. |
| Issue | Feature | Tarea a realizar en el proyecto. | Ver apartado Issue - Feature. |

10.5.2 Version - Release

| Atributo JIRA | Atributo Replan | Descripción | Transformación de Unidades |
|--------------------|--------------------|---|---|
| Name | Name | Es el nombre de la versión. También se puede utilizar como identificador. | No es necesario. |
| startDate | - | Fecha de inicio de la versión. | Es necesario para realizar la transformación de duración de versión a calendario. |
| releaseDate | - | Fecha de fin de la versión. | Es necesario para realizar la transformación de duración de versión a calendario. |
| Description | Description | Descripción de la versión. | No es necesario. |

10.5.3 Issue - Feature

| Atributo JIRA | Atributo Replan Optimizer | Descripción | Transformación de Unidades |
|----------------------|---------------------------|--|---|
| Key | Name | Es el nombre del feature. También se usa como identificador. | No es necesario. |
| originalEstimate | duration | Es la estimación original de la funcionalidad en concreto. | En el caso de JIRA la duración está en milisegundos mientras que en Replan está en horas. |
| OutGoingLinkedIssues | dependsOn | Se utiliza para realizar dependencias. | En el caso de JIRA la función de dependencia no existe, de modo que se asumirá que un Issue depende de otro si tiene un enlace saliente hacia éste. |

10.5.4 ProjectRole - Skill

| Atributo JIRA | Atributo Replan Optimizer | Descripción | Transformación de Unidades |
|---------------|---------------------------|--|----------------------------|
| Name | Name | Es el nombre de la skill. También se usa como identificador. | No es necesario. |

10.5.4 User - Resource

| Atributo JIRA | Atributo Replan Optimizer | Descripción | Transformación de Unidades |
|---------------|---------------------------|--|----------------------------|
| Name | Name | Es el nombre del usuario. También se usa como identificador. | No es necesario. |

10.5.5 Calendar

Debido a que el calendario no aparece de ninguna de las formas en JIRA, se procederá a hacer una breve descripción sobre cómo se pretende obtener a partir de la información que se tiene. Como se ha mencionado en el apartado de las versiones, los atributos StartDate y EndDate de una versión servirán para generar el calendario.

Para ello, se asumirá que las jornadas laborales son de **8 horas**, y tendrán 5 días a la semana, siendo el primer día el que empieza la versión.

También cabe decir que el calendario que se generará inicialmente tendrá todos sus días libres para poder asignar cualquier issue.

11. Especificación del Sistema

En esta sección se procederá a realizar la especificación del sistema que se va a proceder a implementar. Al ser un plugin para JIRA se asumirá que, de cara a definir este sistema el usuario tiene todos los permisos necesarios para editar cualquier tipo de issue así como que ya ha iniciado sesión, de tal forma que sólo habrá que centrarse en los aspectos más concretos de la especificación de ReactivePlan for JIRA.

11.1 Requisitos Funcionales

En esta sección se procederá a definir aquellos requisitos funcionales. Puesto que todos los requisitos funcionales están directamente relacionados con realizar una planificación sobre un proyecto no se realizarán distinciones entre tipos de requisitos funcionales ya que todos satisfacen la misma necesidad.

RF 11.1.1 Realizar Planificación Automática

El sistema debe permitir realizar una planificación automáticamente a un proyecto en concreto y con una versión determinada.

RF 11.1.2 Seleccionar Proyecto y Versión a Planificar

El sistema debe permitir seleccionar el proyecto y la versión a planificar.

RF 11.1.3 Visualizar Issues a Planificar

El sistema debe permitir visualizar cuáles son los *issues* que se van a planificar de cara una planificación automática.

RF 11.1.4 Previsualizar Plan

El sistema debe permitir previsualizar el plan que se obtenga de la planificación automática.

RF 11.1.5 Persistir Plan

El sistema debe permitir persistir el plan obtenido de la planificación automática.

11.2 Requisitos no Funcionales

En este apartado se tendrán en cuenta todos aquellos requisitos que no sean funcionales pero que deban tenerse en cuenta de cara a realizar la herramienta de ReactivePlan for JIRA con la máxima calidad.

11.2.1 Requisitos de Apariencia

RNF 11.2.1.1 Apariencia de JIRA - El sistema debe tener la apariencia según los criterios de diseño de JIRA.

RNF 11.2.1.2 Familiaridad con JIRA - El sistema debe resultar familiar a cualquier usuario que haya utilizado JIRA previamente.

11.2.2 Requisitos de Usabilidad

RNF 11.2.2.1 Aprendizaje Sencillo - El sistema debe ser fácil de usar, facilitando su aprendizaje.

RNF 11.2.2.2 Guiar Errores - El sistema debe hacer que sea complejo cometer un error de usuario. En el caso de que ocurra, el sistema debe guiar al usuario para solucionar estos errores.

11.2.3 Requisitos de Escalabilidad

RNF 11.2.3.1 Sistema Ampliable - El sistema debe permitir ampliar o modificar sencillamente todas sus funcionalidades.

RNF 11.2.3.2 Escalable - El sistema debe funcionar correctamente sea cual sea el número de usuarios que empleen la plataforma.

11.3 Casos de Uso

De cara a definir cómo se implementarán los diversos requisitos funcionales se ha optado por desarrollarlos a través de casos de uso ya que dentro de las diversas especificaciones de requisitos posibles es la que mejor especifica el funcionamiento del sistema así como las diversas acciones que deberán realizar tanto el usuario como el propio sistema.

11.3.1 Diagrama de Casos de Uso

Afin de poder mostrar cómo estos casos de uso se relacionan entre sí, se ha optado por hacer un diagrama de Casos de Uso. Como se puede observar en el diagrama, todos los casos de uso están incluidos en el caso de uso general de *Realizar Planificación Automática*. A la hora de especificar los casos de uso, se realizará individualmente obviando el caso de uso general.

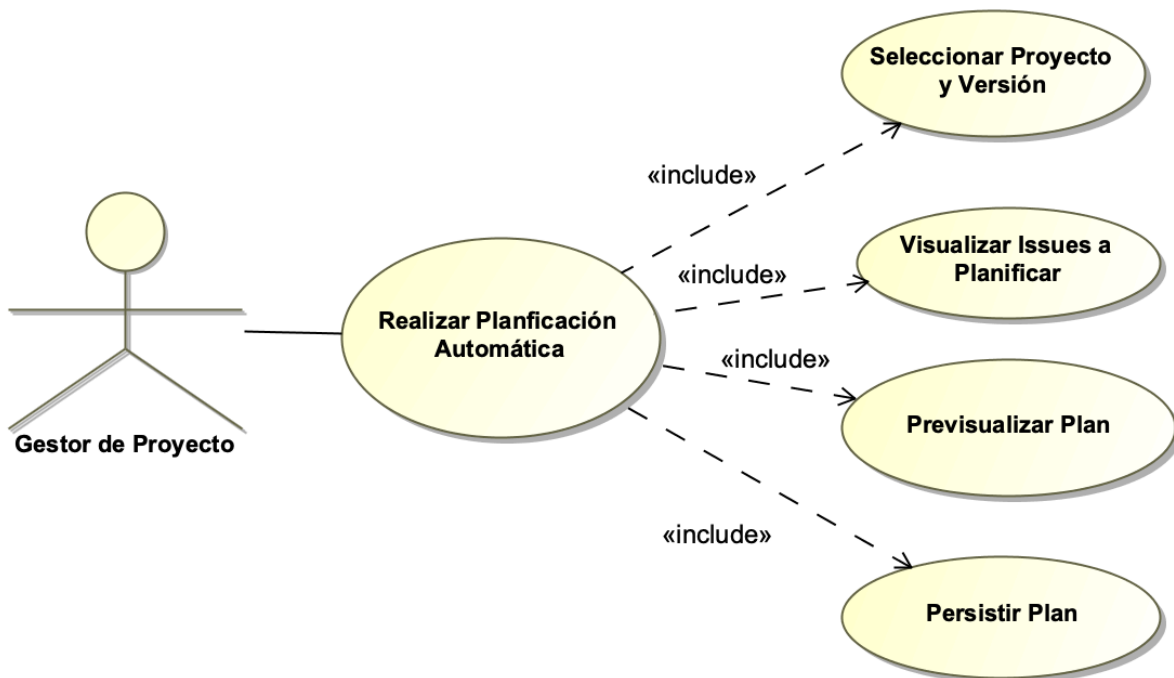


FIG 10. DIAGRAMA DE CASOS DE USO DE REPLAN FOR JIRA

CU 1 Seleccionar Proyecto y Versión

| | |
|-----------------------|---|
| Título | Seleccionar Proyecto y Versión |
| Descripción | El usuario debe poder seleccionar un proyecto y una versión para poder comenzar a planificar los diferentes issues. |
| Pre-Condición | El usuario se encuentra en la vista principal de RePlan |
| Post-Condición | |
| Autor | Gestor de Proyecto |
| Escenario Principal | <ol style="list-style-type: none">1. El usuario comienza a introducir el proyecto.2. El sistema muestra los diferentes proyectos que tiene almacenados.3. El usuario termina de introducir el proyecto o selecciona uno de los desplegados.4. El usuario comienza a introducir la versión.5. El sistema muestra las diferentes versiones que tiene almacenadas.6. El usuario termina de introducir la versión o introduce una de las desplegadas.7. El usuario selecciona la opción de comenzar a planificar. |
| Escenario Alternativo | <p>7.a. El proyecto introducido no existe. 7.a.1 El sistema devuelve un mensaje de error "El proyecto no existe".</p> <p>7.b. La versión introducida no existe. 7.b.1 El sistema devuelve un mensaje de error "La versión no existe".</p> |

CU 2 Visualizar Issues a Planificar

| | |
|-----------------------|---|
| Título | Visualizar Issues a Planificar |
| Descripción | El usuario debe poder visualizar los Issues que va a a planificar para saber si está de acuerdo con la posible planificación que se va a hacer. |
| Pre-Condición | El usuario se encuentra en la vista principal de Replan y ha seleccionado exitosamente proyecto y versión. |
| Post-Condición | El sistema muestra los issues a planificar. |
| Autor | Gestor de Proyecto |
| Escenario Principal | <ol style="list-style-type: none">1. El usuario selecciona la opción de comenzar a planificar.2. El sistema obtiene todos los issues que estén en estado TO DO en la versión y el proyecto previamente especificados.3. El sistema realiza la transformación de issue a feature por cada uno de estos obtenidos.4. El sistema muestra los diversos issues a planificar. |
| Escenario Alternativo | <p>3.a El sistema detecta algún error inesperado. 3.a.1 El sistema muestra el mensaje de error.</p> <p>4.a El usuario selecciona un issue a planificar. 4.a.1 El sistema muestra detalladamente este issue.</p> |

CU 3 Previsualizar Plan

| | |
|-----------------------|--|
| Título | Previsualizar Plan |
| Descripción | El usuario debe poder previsualizar el plan que va a persistir con el fin de saber exactamente qué está planificando. |
| Pre-Condición | El usuario se encuentra en la ventana de Issues a Planificar. |
| Post-Condición | El sistema muestra el plan a realizar. |
| Autor | Gestor de Proyecto |
| Escenario Principal | <ol style="list-style-type: none">1. El usuario selecciona la opción <i>Obtener Plan</i>.2. El sistema carga todos los parámetros requeridos para obtener un plan.3. El sistema realiza la petición para obtener un plan.4. El sistema obtiene el plan.5. El sistema obtiene los <i>issues</i> que no se han podido planificar.6. El sistema muestra el plan.7. El sistema muestra los issues que no se han podido planificar. |
| Escenario Alternativo | <p>3.a. La petición da error y no se puede realizar.</p> <p>3.a.1 El sistema muestra un mensaje de error "No se ha podido realizar el plan".</p> |

CU 4 Persistir Plan

| | |
|-----------------------|---|
| Título | Persistir Plan |
| Descripción | El usuario debe poder persistir el plan con el fin de aplicarlo a su proyecto. |
| Pre-Condición | El usuario se encuentra en la ventana de Previsualización de Plan. |
| Post-Condición | El sistema persiste el plan propuesto. |
| Autor | Gestor de Proyecto |
| Escenario Principal | <ol style="list-style-type: none">1. El usuario selecciona la opción <i>Persistir Plan</i>.2. El sistema obtiene el plan a persistir.3. El sistema realiza las transformaciones pertinentes para poder persistir el plan.4. El sistema persiste el plan.5. El sistema muestra el plan así como un mensaje de éxito: "El plan ha sido persistido correctamente". |
| Escenario Alternativo | <p>1.a. El usuario selecciona la opción "Obtener otro plan".</p> <p>1.a.1 El sistema obtiene un nuevo plan.</p> <p>4.a. El sistema detecta un error a la hora de persistir el plan.</p> <p>4.a.1 El sistema muestra un mensaje de error "El plan no se ha podido persistir".</p> |

12. Diseño del Sistema

En esta sección se procederá a explicar cómo se ha desarrollado el diseño de la solución del sistema ReactivePlan for JIRA tratando más a fondo los aspectos técnicos de la especificación propuesta anteriormente, así como la justificación del por qué se ha tomado cada decisión. También se tratarán aspectos técnicos del desarrollo de plugins para *JIRA Core* ya que no es una tarea sencilla y debe tenerse en cuenta de cara a desarrollar de la forma más completa el sistema en cuestión.

12.1 Arquitectura de un Plugin en JIRA

Afin de poder entender cómo se va a proceder a diseñar el sistema, lo primero que se debe tener en cuenta es cómo funciona la arquitectura de un plugin en esta plataforma.

Para ello, se debe aclarar una serie de conceptos técnicos de JIRA los cuáles serán necesarios definir para poder entender cómo se ha desarrollado la solución. Para ello nos ayudaremos del diagrama Arquitectónico de JIRA así como la definición de qué es un plugin para JIRA.

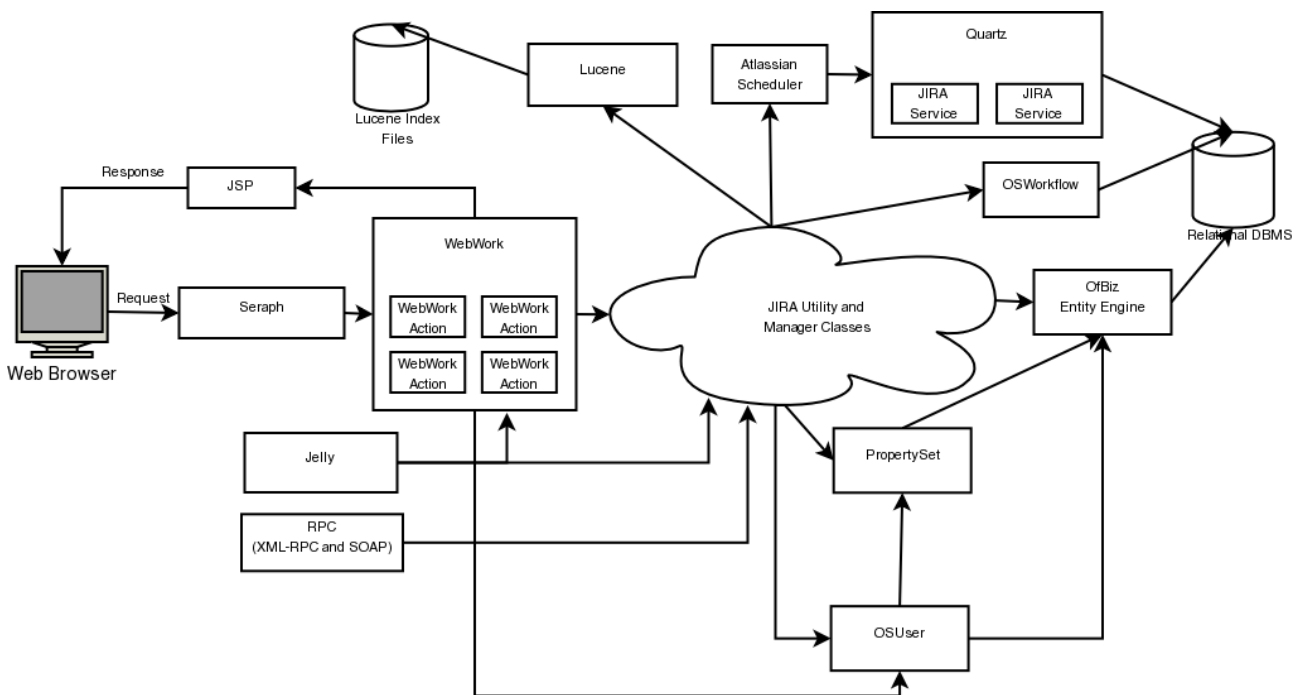


FIG 11. ARQUITECTURA DE JIRA [20]

De este diagrama, lo que debemos sacar en claro de cara al diseño de nuestra solución es la existencia del ***Jira Utility And Manager Classes***, los cuáles serán totalmente necesarios para poder obtener la diferente información de JIRA, así como editar la información existente. Ya que estas diferentes clases son las que se encargan de controlar todo el flujo de datos a lo largo de la aplicación.

Por siguiente, de cara a desarrollar un plugin, debemos saber diversas cosas sobre estos.

Un **plugin** en **JIRA** se adhiere al sistema como un **componente**, y cada plugin a su vez constará de diversos **módulos**.

A fin de aclarar exactamente qué entendemos por ***Jira Utility And Manager Classes***, así como **Módulos de un Plugin** se procederá a definir qué es cada uno de estos elementos.

Módulo de un Plugin:

Los módulos de un plugin son extensiones de la Interfaz de usuario que las aplicaciones pueden utilizar para insertar contenido en diversas zonas de la interfaz de la aplicación. Los módulos de un plugin son parte de la plataforma de Jira y del mismo modo, son comunes a todas las aplicaciones de Jira. Sin embargo, hay que tener en cuenta que también se pueden emplear estos módulos para acceder a partes específicas de la interfaz de usuario de la aplicación. De cara a desarrollar un plugin para JIRA habrá diversos módulos de entre los que se podrán seleccionar de cara a realizar un plugin.

Jira Utility And Manager Classes:

Implementan la lógica de negocio de JIRA, la cuál está repartida en cientos de clases en JAVA.

12.2 Módulos de ReactivePlan for JIRA

Afín de diseñar una correcta solución, primero se deben escoger qué módulos ha de implementar el *plugin* para que lo desarrollado esté correctamente integrado con JIRA.

Para ello, se han usado dos módulos diferentes (*definidos en el atlassian-plugin.xml*).

12.2.1 Web Item

De cara a poder acceder a las diferentes ventana de REPLAN, se ha creado un Web Item [21] llamado **Replan**. Este módulo es bastante simple puesto que lo que hace únicamente es añadir un enlace dentro del propio JIRA .

12.2.2 Servlet

Para poder desarrollar el plugin así como realizar el control y gestión de las llamadas web pertinentes, se ha utilizado el módulo de **Servlet**. Hay que decir también que este módulo es importante puesto que permite inyectar en él las diferentes **Utility and Manager Classes** mencionadas anteriormente y no existe otra forma de acceder a ellas.

Para desarrollar esta aplicación, se han creado dos módulos de Servlets:

- **ReplanServlet:** Es el encargado de todo el control de la aplicación. Tanto de realizar las llamadas a las diversas planificaciones y clases implementadas como de mostrar la información gráficamente.
- **TestServlet:** Es un servlet creado para realizar las diferentes pruebas de la aplicación que requieren el uso de un Servlet para obtener los diferentes **Utility and Manager Classes**. Pese a que este servlet no realiza aserciones, se ha empleado de cara a realizar las diferentes pruebas de funcionalidades en la aplicación.

12.3 Diagramas de Clases de la Solución

Para poder mostrar cómo va a ser la solución planteada, se va a proceder a desarrollar el diagrama de clases de la solución con todas las relaciones y funciones que ofrecerán las distintas clases desarrolladas. Debido a que el nombre de cada operación es autodescriptivo, no se procederá a describir ninguna de las funciones salvo que sea excesivamente compleja y pueda causar confusión.

Por otro lado también se omitirán los **getters y setters** de los diversos atributos pues se dará por hecho que ya están implementados. Los constructores tampoco se añadirán en el diagrama de clases de la solución.

También se tendrán en cuenta las entidades del modelo conceptual de JIRA que se mencionó en el apartado 10.3 para desarrollar esta solución.

Cabe decir que por cómo se ha realizado la solución, habrá diversas capas a la hora de hacer el diagrama de clases, las cuáles serán las siguientes:

- **Entidades**
- **Lógica**
- **Conversores de Jira a Replan y viceversa.**
- **Api Handler**
- **Servlet**

12.3.1 Entidades

La primera parte de la solución corresponde en implementar las diferentes entidades que utiliza *Replan Optimizer* de cara a la elaboración de un plan, de tal modo que se pueda trabajar con entidades en lugar de con valores abstractos para realizar las diferentes peticiones. Es por ello que se ha diseñado una solución de cara a importar la entidades que utiliza esta herramienta.

Como se puede apreciar en la imagen que muestra a continuación, se han diseñado los diferentes elementos que aparecían en los apartados anteriores a lo largo del modelo conceptual. También se han añadido diversas funciones para que el modelo encaje lo más cómodamente posible en la plataforma de JIRA.

El único hecho que hay que comentar concretamente en este modelo es sobre los **DaySlots**.

Esta clase, por como está diseñada en *Replan Optimizer*, requiere un número de día y de semana que dista de figurar en una fecha real, y empieza en el valor 0. Para poder obtener la fecha real de cada DaySlot basándose en una fecha base, se ha optado por añadir la función **getDate()**, la cual tiene como parámetro de entrada una fecha que es la que se utilizará como fecha base para añadir los diversos días y semanas que correspondan afín de poder calcular la fecha real para mostrar en JIRA la fecha en la que los diferentes Issues deben ser planificados.

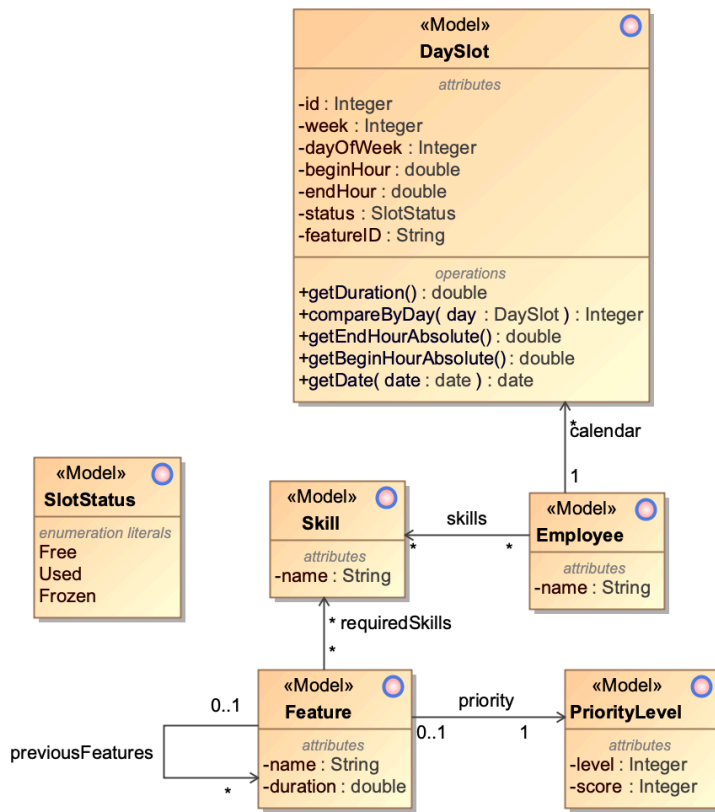


FIG 12. ENTIDADES REPLAN

12.3.2 Lógica

Con el fin de acceder a los diversos elementos que ofrece JIRA, se ha procedido a diseñar una lógica que se encargará de las funciones de obtener los diferentes elementos de la plataforma y enviárselo al resto de capas de ReactivePlan for JIRA para desacoplar la lógica de negocio con el resto de capas del sistema.

Concretamente, esta capa es la que accede a los **JIRA Utility and Manager Classes**, y realiza las peticiones necesarias para poder desarrollar correctamente el plugin.

Cabe también destacar que al ser simplemente una capa de lógica, se ha aplicado el patrón de diseño **Singleton** de tal forma que sólo pueda existir una única instancia de cada una de las clases que se van a mostrar en el diagrama de clases a continuación.

La función de esta lógica será la de permitir realizar la **creación, actualización, edición y eliminación** de los diferentes objetos que se van a tratar en JIRA, así como efectuar toda la persistencia necesaria para la aplicación.

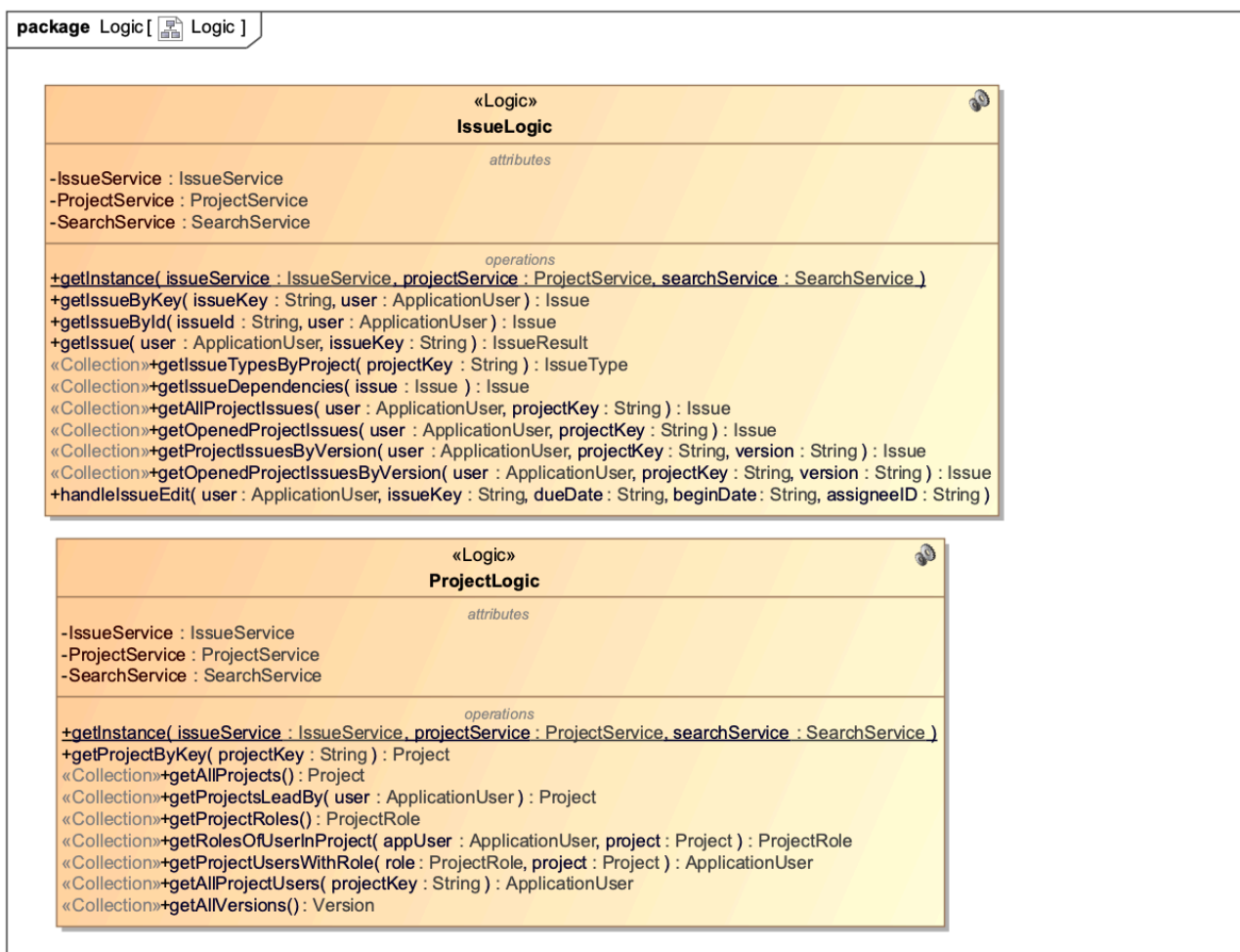


FIG 13. LÓGICA REPLAN

12.3.3 Conversores

Debido a que los modelos de JIRA y Replan son bastante diferentes, tal y como se mencionó en el apartado de **Comparación Conceptual**, es necesario definir una lógica de conversión eficiente con tal de que se trabaje con los datos en el formato correcto de tal modo que cada una de las interfaces pueda emplear estos datos así como, en el caso final, poder persistirlos en la aplicación y hacerlos visibles para el usuario.

Para ello, se han diseñado dos clases cuyos métodos serán **estáticos** y requerirán en ciertos casos la inyección de la lógica de negocio mencionada anteriormente para poder realizar diversas modificaciones en el sistema. El fin de estos conversores es, como ya se ha mencionado separar la lógica de conversión de un formato a otro de la del resto del sistema, produciendo el mayor desacople posible en este sentido.

Para ello, se han creado dos conversores diferentes.

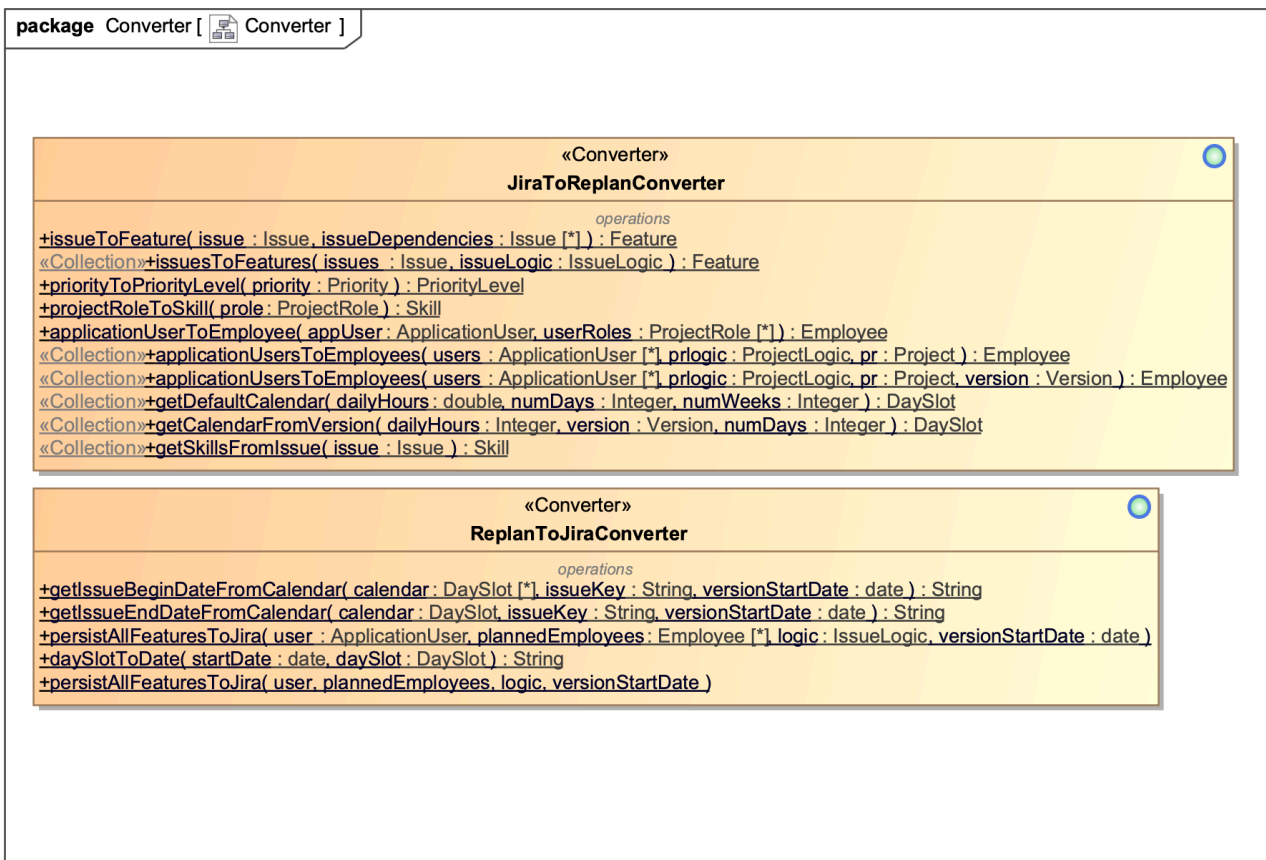


FIG 14. CONVERSORES REPLAN

Uno será el **JiraToReplanConverter** cuya funcionalidad será la de extraer los diferentes datos de JIRA y transformarlos al formato de Replan.

Entre estas tareas podemos destacar:

- Generación de calendarios **DaySlots** a partir de fechas de **Versiones**.
- Obtención de las diferentes **Skills** de un **Issue**.
- Transformación de los **Project Roles** de un proyecto en **Skills**.

Por consiguiente, una vez se ha obtenido la planificación se utilizará el **ReplanToJiraConverter** que realizará la función inversa. También se encargará de la lógica de persistir un conjunto de **Issues** ya planificados y demandar a la **lógica** que los persista en el formato correcto.

12.3.4 API Handlers

Con el fin de poder realizar correctamente la gestión de llamadas a la API del **Replan Optimizer** se ha propuesto el desarrollo de dos **API Handlers**. Estos objetos, además de tener como función encapsular todas las entidades que componen una llamada a ésta herramienta tanto en su **petición** como en su **respuesta**, se encargarán de toda la lógica de **serialización o deserialización** de los objetos que requiera el **replan optimizer** para desacoplar el manejo de la API con el del resto del sistema.

Para satisfacer esta necesidad, se han creado las clases que se muestran en el siguiente diagrama:

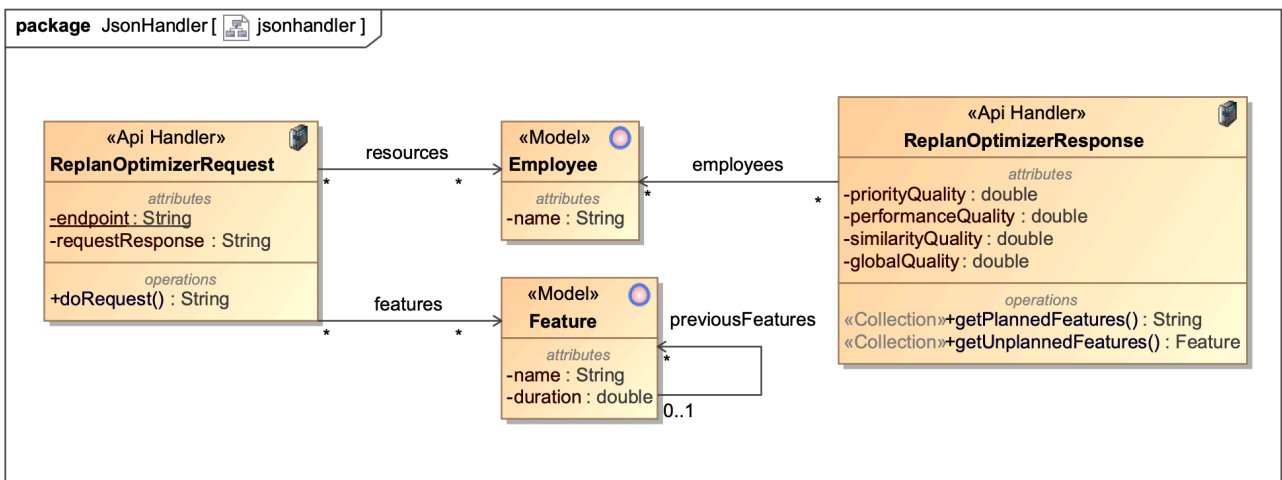


FIG 15. API HANDLERS REPLAN

12.3.5 Servlet

Con el fin de poder gestionar todas las peticiones HTTP que se hagan a lo largo del sistema así como controlar el flujo de la aplicación se ha procedido a diseñar un único **Servlet** que será el en cargado de gestionar todas las peticiones llamado ReplanServlet.



FIG 16. SERVLET REPLAN

12.3 Diagrama de Paquetes de la Solución

Con el fin de poder visualizar rápidamente cómo se van a organizar los distintos paquetes de la solución así como poder ver cuáles serán sus dependencias se ha optado por diseñar un diagrama de paquetes.

Para ver el paquete en el que se encuentra cada conjunto de elementos, sólo debemos mirar en la esquina superior de cada diagrama y así se puede visualizar a qué paquete pertenece cada una de las clases propuestas en los diagramas anteriores.

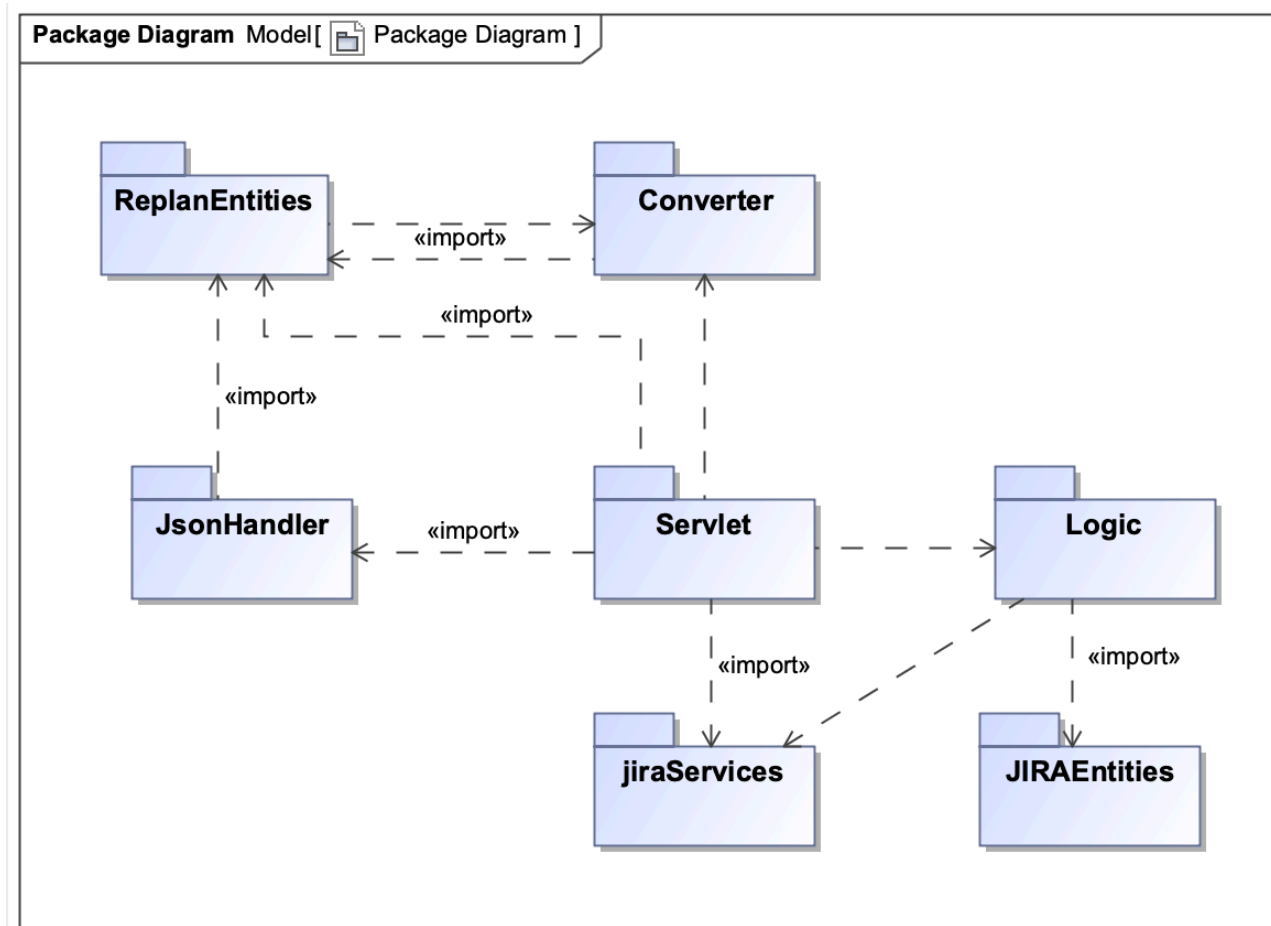


FIG 17. DIAGRAMA PAQUETES REPLAN

12.4 Diagrama de Navegación

Debido a que JIRA emplea diversas vistas y componentes *web* para poder realizar la visualización de los diferentes elementos, se ha procedido a realizar el diseño de la navegación del sistema **ReactivePlan for JIRA** con el fin de establecer una navegación sólida y sin errores así como realizar el diseño de la forma más precisa posible.

Con el fin de diseñar qué es lo que se va a representar en cada una de las vistas y cómo se transita entre ellas, se ha procedido a diseñar un diagrama de flujo de navegación. Como se puede observar en el diagrama, existirán tres vistas principales las cuáles contendrán cierta información relacionada con el estado de la planificación en el que se encuentre el sistema.

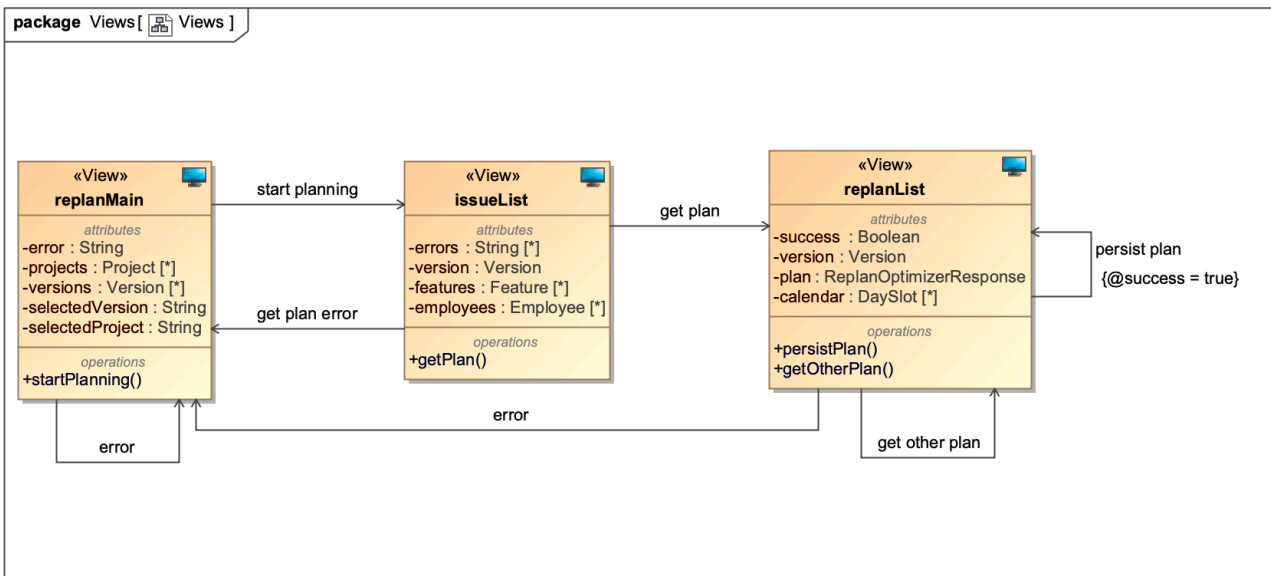


FIG 18. DIAGRAMA DE FLUJO DE NAVEGACIÓN

12.5 Diseño interno de la capa de Presentación

Afín de diseñar cómo se renderizarán las diversas vistas así como qué peticiones hay que realizar para dar paso a cada **Caso de Uso** de los definidos previamente y decidir qué clases interactuarán en cada petición, se ha procedido en realizar un diseño interno de la capa de presentación de la aplicación. Este diseño mostrará las peticiones que se realizan desde cada vista así como la gestión que realiza el controlador en cuestión.

12.5.1 Diseño interno del CU1 y CU2

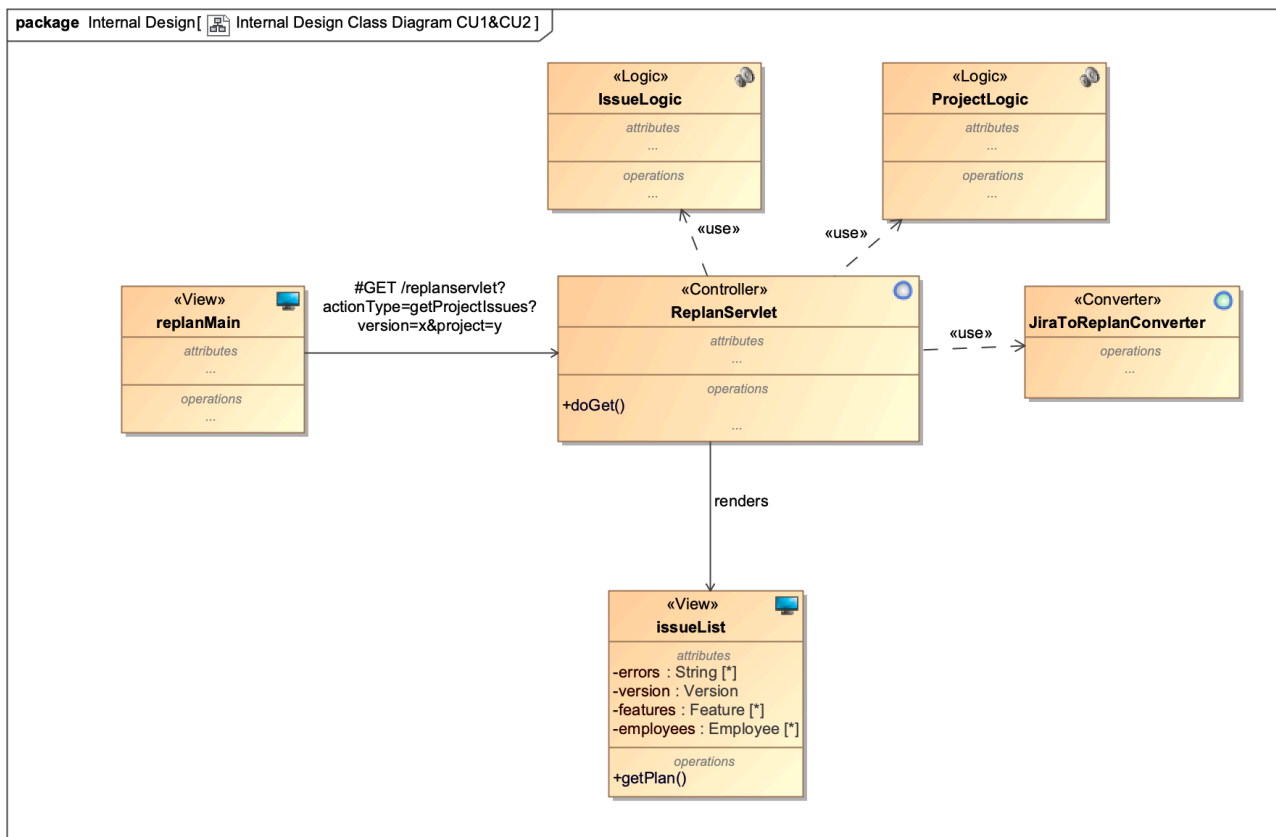


FIG 19. DISEÑO INTERNO CASOS DE USO 1 Y 2

Como podemos ver, el flujo de la aplicación comienza en la vista **replanMain** enviándole una petición **GET** al **Servlet** con los parámetros **project** y **versión** establecidos en el **Caso de Uso 1**. Por siguiente, el **Servlet** se encargará de gestionar esta petición y realizar las llamadas pertinentes a las diversas clases con las que se establece una relación *use* con el fin de poder pintar una lista de los diferentes **issues** que se van a proceder a planificar.

12.5.2 Diseño interno del CU3

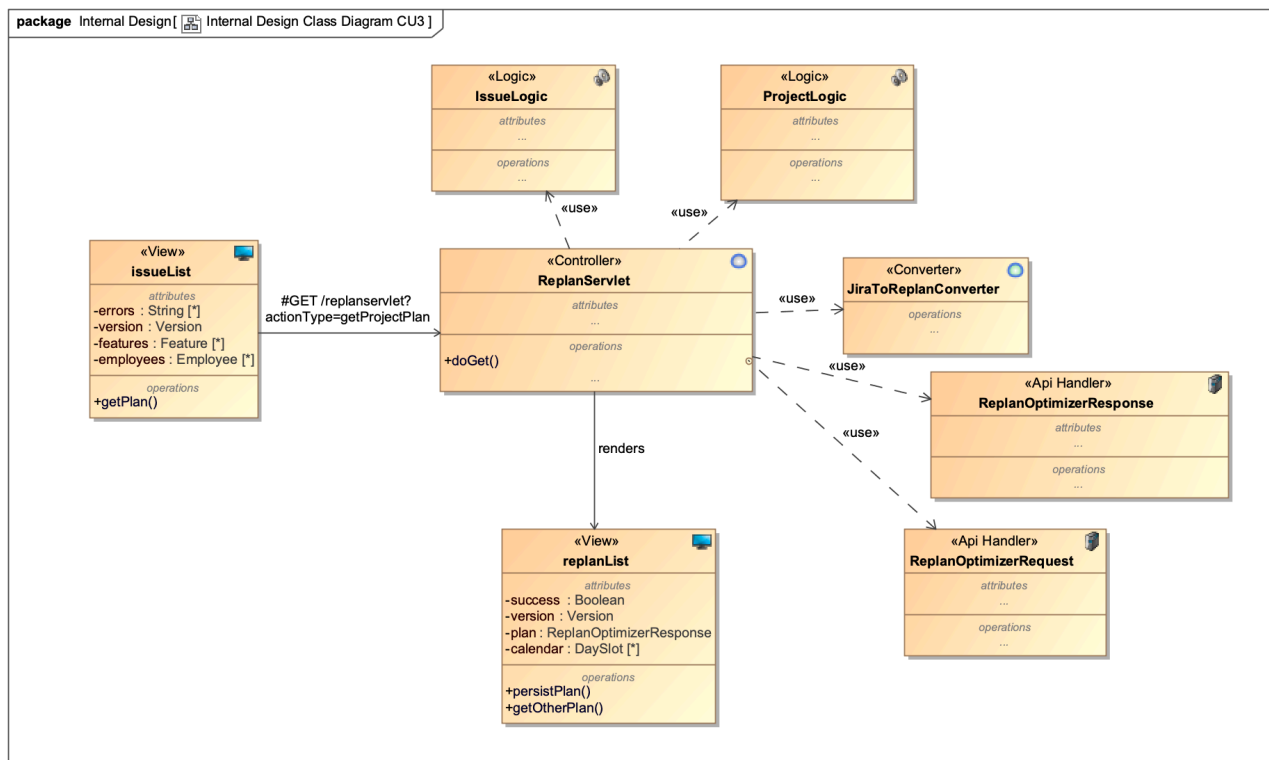


FIG 20. DISEÑO INTERNO CASOS DE USO 3

Como se puede apreciar en la figura, el procedimiento es análogo al mostrado en los casos de uso anteriores sólo que añadiendo nuevos usos. En este Caso de Uso ya se comienzan a emplear los **API Handlers** debido a que el **Servlet** ya gestiona la llamada con el **Replan Optimizer**.

A la hora de realizar la petición **GET**, no se han añadido parámetros ya que los parámetros de **proyecto** y **versión** se encargará el servlet de almacenarlos previamente con el fin de poder reutilizarlos en las siguientes peticiones.

12.5.3 Diseño interno del CU4

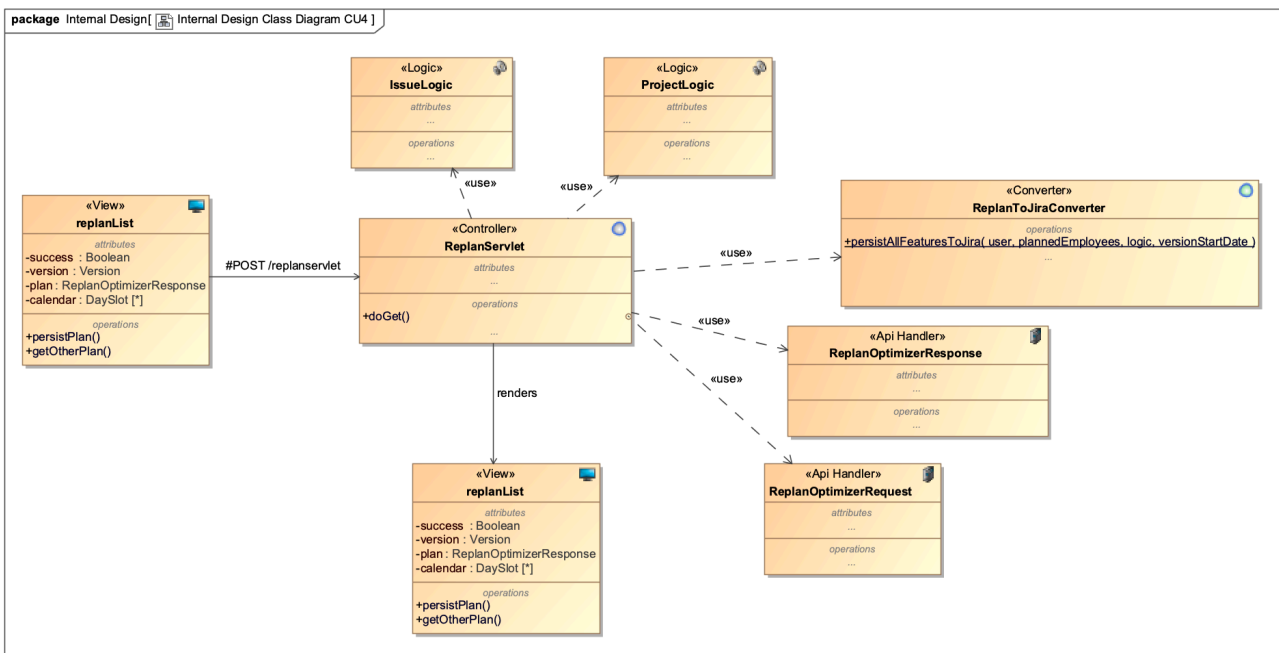


FIG 21. DISEÑO INTERNO CASOS DE USO 4

De cara a este diseño, debemos observar dos cosas que se han decidido.

La primera es que el **Servlet**, en lugar de redirigir la petición **POST** hacia otra vista tras persistir el plan, redirige a la misma estableciendo un mensaje de éxito si se ha podido persistir el plan o de error si ha habido cualquier problema.

La segunda sería el uso de la clase **ReplanToJiraConverter** para que se encargue de toda la lógica de esta persistencia.

12.6 Diseño de llamadas: Diagramas de Secuencia

Con el fin de especificar aún más cómo interactúan en cada **Caso de Uso**, las diferentes clases entre así, así con cómo se dispara cada uno de los eventos, se ha procedido a diseñar diversos **diagramas de secuencia** con las llamadas reales que efectuará el sistema.

Para el desarrollo de estos diagramas sólo se han tenido en cuenta las llamadas dentro del propio sistema **ReactivePlan for JIRA** y no las conexiones directas con **JIRA** o **Replan** ya que aumentaría la dificultad de comprensión de estos diagramas.

Para la gestión de las llamadas externas se ha decidido encapsularlas en las operaciones que realmente se encargarán de realizar estas llamadas.

Por otro lado, hay que aclarar que únicamente se ha tenido en cuenta el escenario de éxito de cara a diseñar estos diagramas de secuencia con lo que la gestión de errores o escenarios de alternativos no se ha diseñado.

12.6.1 CU1 - Diagrama de Secuencia

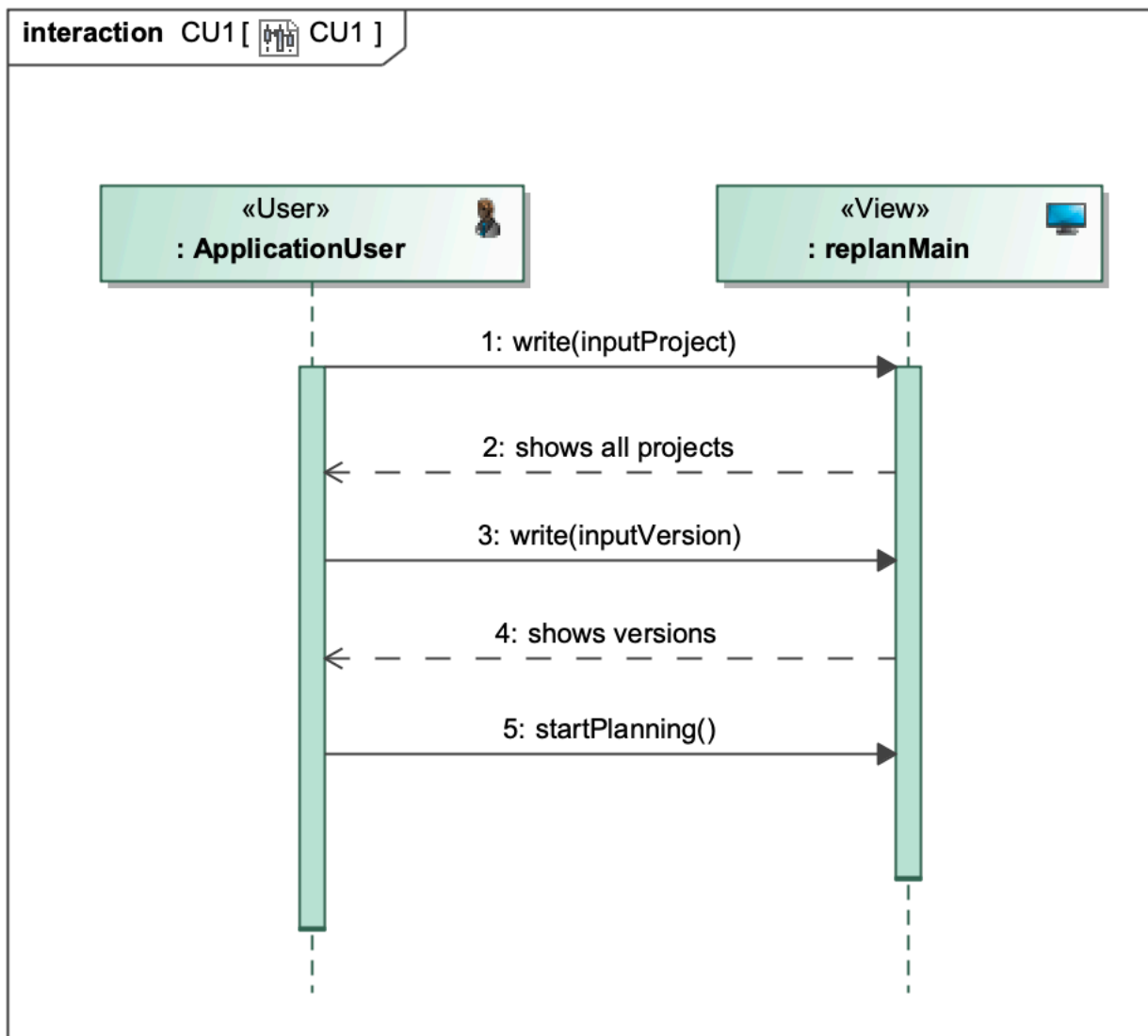


FIG 22. DIAGRAMA DE SECUENCIA CU1

Este diagrama de secuencia muestra la interacción entre el usuario y la vista principal de **ReactivePlan for JIRA**.

12.6.2 CU2 - Diagrama de Secuencia

En este diagrama de secuencia se muestra todo el proceso de obtener los **issues** que se van a planear posteriormente hasta la creación de la vista que los muestra.

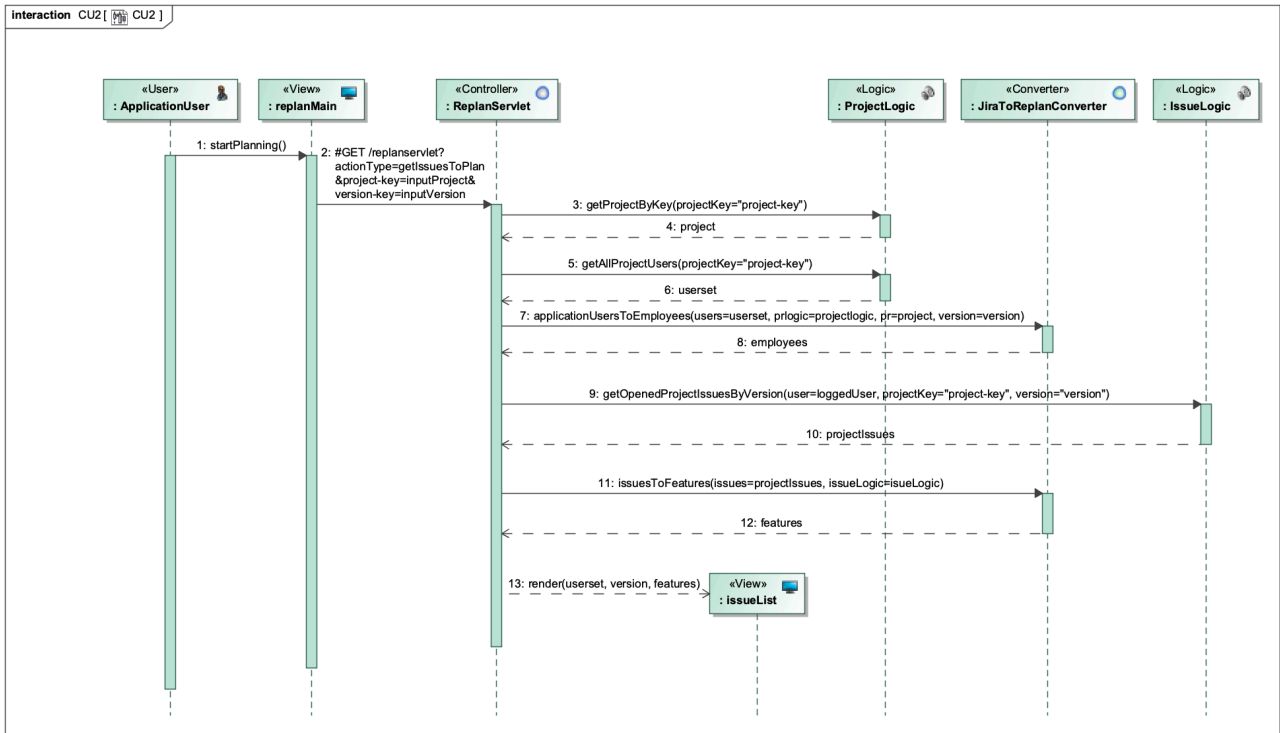


FIG 23. DIAGRAMA DE SECUENCIA CU2

12.6.3 CU3 - Diagrama de Secuencia

Este diagrama de secuencia muestra las diversas peticiones que se realizan de cara a la obtención de un plan así como crear la vista que los muestra.

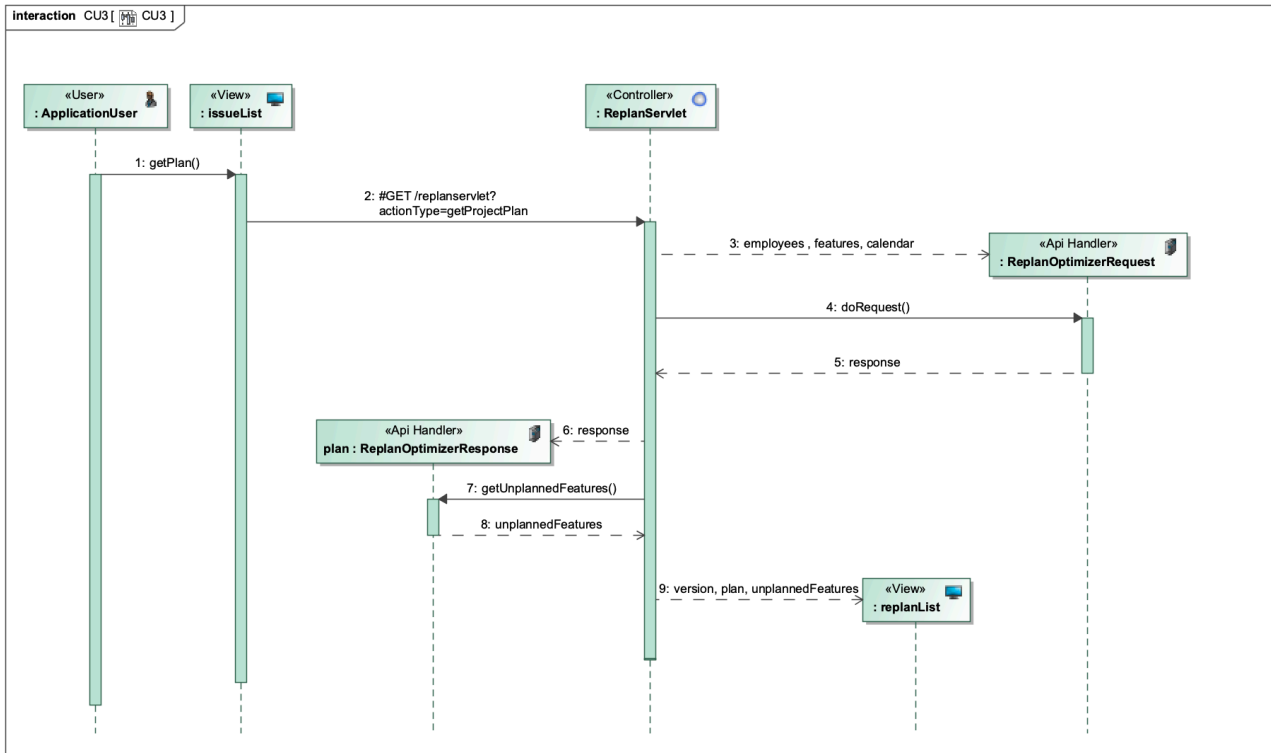


FIG 24. DIAGRAMA DE SECUENCIA CU3

12.6.4 CU4 - Diagrama de Secuencia

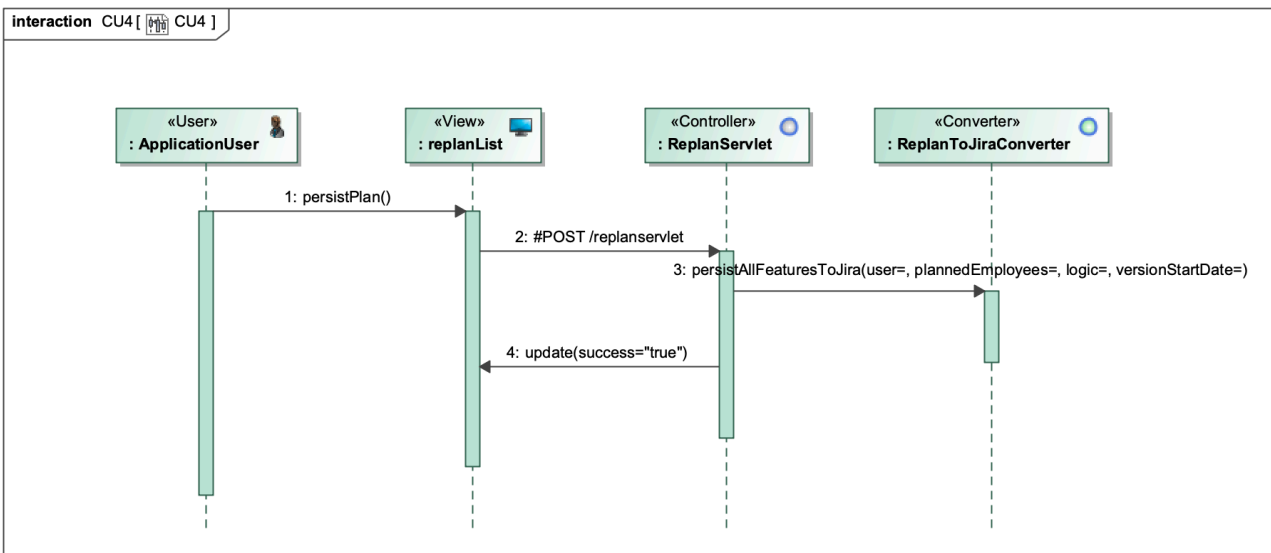


FIG 25. DIAGRAMA DE SECUENCIA CU4

Este diagrama de secuencia muestra la persistencia del plan en JIRA.

12.7 Tecnologías a Emplear

Por último, una vez se ha diseñado todo aquello que se va a implementar, se ve a proceder a describir las tecnologías.

12.7.1 Back-End

Por cómo se ha escogido el desarrollo de este plugin y la integración con JIRA, así como el funcionamiento de toda la lógica, tiene que ser desarrollada en **JAVA** ya que **JIRA** está desarrollado en este lenguaje.

De este modo, de cara a la implementación todas las clases mencionadas que no pertenezcan implícitamente a ninguna vista serán desarrolladas en este lenguaje. Cabe también decir que una de las ventajas de emplear **Java** es que a la hora de implementarlo basado en un diagrama de clases **UML** como los mostrados en los apartados anteriores, resulta ser directa.

12.7.2 Front-End

Aún existiendo una gran libertad de cara el desarrollo del Front-End, pudiendo emplear **JSP** para desarrollar estas vistas al tener **JIRA** implementado el **Spring Framework** se ha decidido desarrollar todo el Front-End en **Velocity Templates** [21] debido a la facilidad que existe para gestionar los datos en estas plantillas así como gestionar los diferentes atributos.

12.8 Patrones de Diseño Empleados

En este apartado se va a proceder a definir de forma breve qué patrones de diseño se han aplicado a lo largo de diseño así como su justificación.

12.8.1 Singleton

Se ha aplicado este patrón en las diferentes clases de **lógica** ya que, al encargarse estas clases únicamente de acceder a **JIRA** el sistema sólo requerirá de una **instancia** de cada una de las clases.

12.8.2 Modelo Vista Controlador

Para evitar cargar de lógica excesiva a los diferentes componentes que engloban la aplicación, se ha optado por separar **modelo** (*entities*), **vista** (views) y **controlador** (servlet) de tal modo que esta división en tres capas favorezca a la sencillez del desarrollo así como en un futuro la mantenibilidad del código.

13. Implementación del Sistema

En este apartado se va a proceder a mencionar todas las herramientas lenguajes, librerías y otros factores que se han tenido en cuenta de cara a poder implementar **ReactivePlan For Jira**, así como la justificación de por qué se ha tomado todas y cada una de las decisiones.

13.1 Atlassian SDK

Atlassian SDK es un framework [22] desarrollado por **Atlassian** para facilitar el desarrollo de **add-ons** para su plataforma.

Esta herramienta ha sido la empleada para correr el servidor de **JIRA** así como crear el esqueleto de un plugin.

En el caso de este proyecto, se ha utilizado este framework para realizar las siguientes tareas:

- Creación de un Plugin
- Gestión de Dependencias (Maven)
- Añadir módulos al plugin en cuestión
- Servidor de JIRA

De cara al desarrollo de este proyecto, toda las herramientas de Atlassian han sido instaladas y desplegadas en **local**, en lugar de alguna plataforma externa.

Por último también cabe decir que, de cara a poder desarrollar correctamente en **JIRA**, ha sido necesario solicitar a **Atlassian** una licencia de **JIRA Core** que ha sido necesaria aplicar a través de este framework para poder desarrollar pasado el **tiempo de prueba** de la herramienta.

13.2 Replan Optimizer

De cara a implementar este plugin, se ha utilizado un **end-point** de **Replan Optimizer** ya desplegado por sus desarrolladores de tal modo que no ha sido necesario configurar nada respecto a esta herramienta puesto que lo único que se ha hecho ha sido utilizarla en remoto.

También hay que decir que por cómo está desarrollada esta herramienta sólo acepta archivos **JSON** como entrada y es por ello que, en etapas anteriores de este documento se especificaron objetos para realizar las conversiones de estos archivos.

El **end-point** utilizado en la versión final de este proyecto es el siguiente:

http://gessi-sw.essi.upc.edu:8080/replan_optimizer-0.0.1/replan

13.3 IntelliJ IDEA 2019.1

De cara a realizar toda la programación de este proyecto se ha empleado ese **IDE** el cuál ofrece soporte para la **programación** en **JAVA** así como diversos **plugins** para facilitar la programación. En nuestro caso, se ha instalado también un plugin que ofrece soporte para las **velocity templates**.

También ofrece una gestión de dependencias de **MAVEN** y permite escribir correctamente las dependencias en el **pom.xml** para que no exista ningún error en este archivo.

Alguna de las características que ofrece esta herramienta es la de corregir código así como la refactorización y la autocompleción. También facilita el control de versiones con **Git** mostrando de diferentes colores qué archivos han sido modificados y cuáles no.

13.4 POSTMAN

Postman [23] es una herramienta de **desarrollo de apis** que permite configurar diferentes **end-points** así como realizar todo tipo de **peticiones HTTP** a éstas, guardarlas y gestionarlas.

Esta herramienta ha sido utilizada principalmente en las fases iniciales del desarrollo para realizar las primeras pruebas a la **herramienta Replan Optimizer** así como verificar que los diferentes archivos **JSON** que devolvía el sistema, estaban bien formados antes de realizar cualquier tipo de petición de forma automática.

13.5 Github y Github Desktop

Con el fin de realizar un correcto control de versiones así como la posibilidad de tener el código en una plataforma externa al ordenador donde se ha desarrollado este proyecto, se ha utilizado la plataforma de **Github**, siendo una de las plataformas de desarrollo de software más empleadas en la actualidad.

Por otro lado, de cara a hacer las diferentes operaciones que se pueden realizar en **GitHub** para dar lugar al control de versiones, se ha empleado **GitHub Desktop** para que sea más sencillo y gráfico realizar este control.

De cara a la implementación del código de este proyecto se ha creado un único **repositorio** el cual se puede encontrar en el siguiente enlace:

<https://github.com/Zaexv/ReactivePlanPlugin>

13.6 Andamiaje del Proyecto

Como bien se ha mencionado en el apartado 13.1, es el **Atlassian SDK el que ofrece el esqueleto del proyecto**, de tal forma que si queremos ver el esqueleto de plugins de JIRA sólo debemos consultar la documentación de estos [3]. Por consiguiente, en este apartado se procederá a mostrar y describir brevemente cómo se ha realizado el andamiaje de este proyecto a nivel de **carpetas y clases**. Sólo se tendrán en cuenta las carpetas y clases que se encuentren dentro de **/src/main**. Por otro lado, también se procederá a explicar brevemente algunos archivos clave de cara al desarrollo de plugins en **JIRA** que no se hayan explicado previamente en otros apartados.

Con tal de mostrar cómo ha sido el andamiaje del proyecto con la mayor claridad posible, se hará uso de un **Diagrama de Paquetes**, donde los paquetes empleados en **Java** tendrán el nombre real. De cara a representar los archivos usados se usarán los **artefectos**, especificando también el formato en cuestión en el que han sido creados.

13.6.1 Andamiaje JIRA

De cara a representar el andamiaje de las clases que se han utilizado para implementar la lógica de JIRA se ha creado el siguiente diagrama:

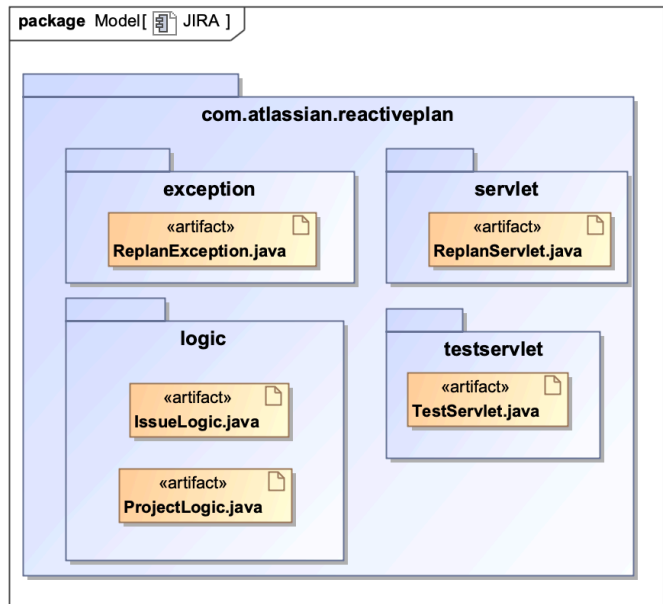


FIG 26. ANDAMIAJE IMPLEMENTACIÓN JIRA

13.6.2 Andamiaje REPLAN

De cara a representar cómo se ha desarrollado el andamiaje de las clases que se han utilizado para implementar las entidades, conversores y *api handlers* de Replan, se ha creado el siguiente diagrama:

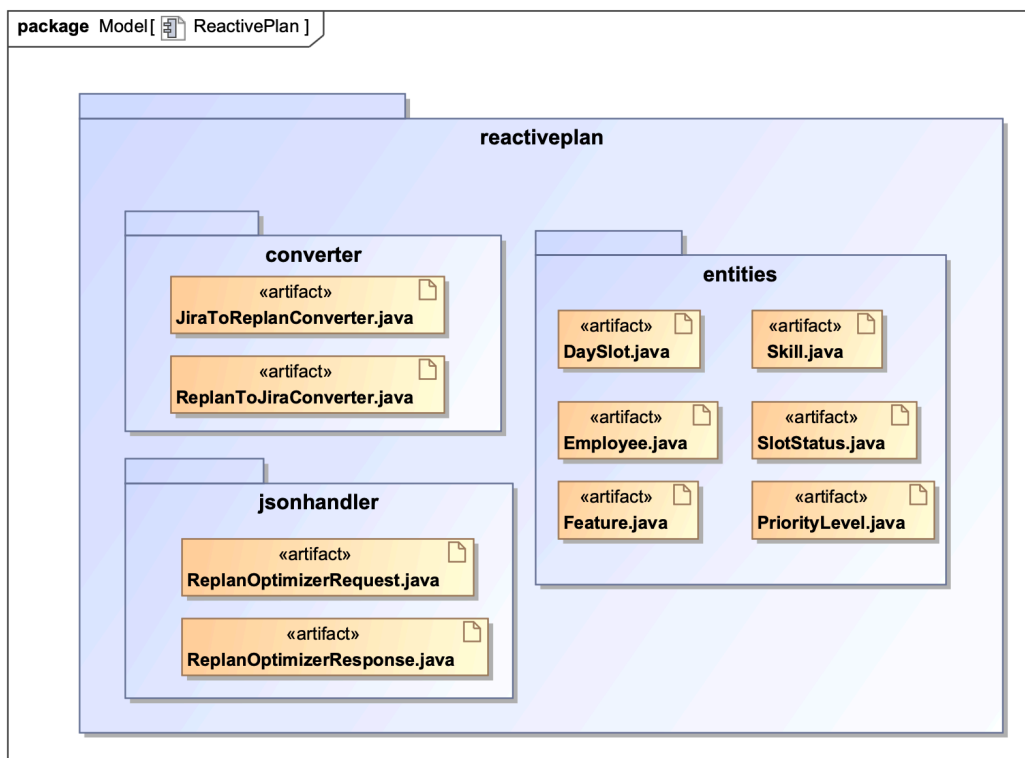


FIG 27. ANDAMIAJE IMPLEMENTACIÓN REPLAN

13.6.3 Andamiaje del Front-End

De cara a representar cómo se ha desarrollado el andamiaje de las diferentes vista y recursos empleados se ha procedido a desarrollar el siguiente diagrama de paquetes.

Cabe destacar que como se ha mencionado las diversas plantillas se encuentran en formato **velocity**.

Quizá lo más importante de este archivo es la existencia del *artefacto* **atlassian-plugin.xml**, este archivo es el encargado de definir todos los componentes que tendrá el plugin en cuestión así como cierta información básica sobre el propio plugin.

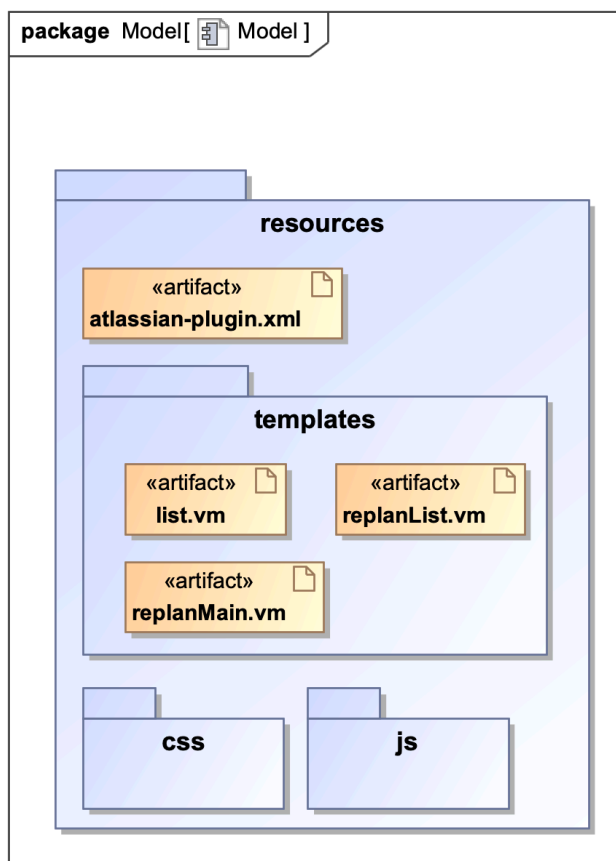


FIG 28. ANDAMIAJE IMPLEMENTACIÓN DE RECURSOS

13.7 Librerías Externas Empleadas

De cara a la realización de este proyecto se han empleado varias librerías externas para evitar duplicar código o tener que desarrollar más código del necesario.

Estas librerías se han empleado de cara a la conexión con la **API del Replan Optimizer** así como a la **Serialización / Deserialización** de las peticiones y respuestas a ésta.

Las librerías empleadas han sido las siguientes:

com.google.gson.Gson: Es una librería creada por Google cuya finalidad es serializar / deserializar los objetos JAVA en JSON. Para ello, utiliza los nombres de los diferentes atributos de cada uno de estos objetos, siendo la transformación 1:1. Se ha utilizado esta librería debido a su sencillez de uso y fácil aprendizaje.

org.apache.http.client: Es una librería creada por Apache cuya finalidad es ofrecer un cliente **HTTP** para realizar diferentes peticiones a cualquier API o servicio web. Esta librería ha sido empleada para realizar las peticiones al **replan optimizer**, así como gestionar su respuesta y los diferentes errores **HTTP** que podría dar. Se ha escogido a este librería debido a que el **Atlassian SDK** cuando crea un plugin, genera la dependencia por defecto ya que es la que usa **JIRA** para sus peticiones de modo que se ha escogido con el fin de tener el menor número de dependencias posibles a lo largo del proyecto.

14. Pruebas

Con el fin de asegurar la calidad del sistema, se han diseñado pruebas tanto para realizar a lo largo del proyecto como para probar la versión final.

Para ello, se han agrupado las pruebas principalmente en dos situaciones. Aquellas desarrolladas durante el desarrollo así como las pruebas **de Sistema** una vez éste ha estado desarrollado totalmente.

14.1 Pruebas Funcionales

De cara a probar las diversas funcionalidades del sistema durante el desarrollo, principalmente se ha optado por crear un **Servlet** para realizar estas pruebas llamado **TestServlet**. Para verificar el correcto funcionamiento de las diferentes operaciones, se ha creado un proyecto llamado **Proyecto de Prueba** y utilizado sus datos para las diferentes operaciones que este **Servlet** ofrece.

Por siguiente, lo se muestra por pantalla los resultados de las diferentes operaciones y a partir de esos resultados, el desarrollador decide si la función está implementada correctamente o no.

Las pruebas tienen un nombre autodescriptivo. La siguiente tabla muestra el nombre de cada prueba, una pequeña descripción de en qué consiste y qué fallos se han detectado en cada una de ellas.

| Prueba | Descripción | Errores Detectados |
|--|--|--|
| testIssueLogic_getIssueByKey | Se obtiene un issue cualquier a a través de su key y se muestra por pantalla. | Ninguno |
| testIssueLogic_getProjectIssues | Se obtienen todos los issues de un proyecto. | Ninguno |
| testProjectLogic_getProjectRoles | Se obtienen todos los roles de un proyecto. | -A la hora de obtener los diferentes roles de proyecto algunos aparecían repetidos. |
| testProjectLogic_getProjectUsersWithRole | Se obtienen todos los usuarios trabajando en un proyecto con un rol en concreto. | Ninguno |
| testJiraToReplanConverter_issueToFeature | Se transforma un issue en un feature . | -La duración de un Issue es diferente a la de un Feature. Hubo que realizar transformaciones. -Los features podían depender de sí mismos. |
| testJiraToReplanConverter_getDefaultCalendar | Se obtiene un calendario a partir de un número de días y de semanas. | Ninguno |

| Prueba | Descripción | Errores Detectados |
|------------------------------|---|---|
| testReplanToJira_generalTest | Se ejecuta un caso de uso completo del sistema, desde la obtención de issues hasta la persistencia. | -JSON mal formados. -Errores en Replan Optimizer (si la duración de un issue es 0.0, devuelve planificaciones erróneas). |
| testAllIssueGetters | Se prueban todos los getters de issues de la lógica. | -Devolvía issues ya iniciados. |

14.2 Pruebas de Sistema

En este apartado se mostrarán las diversas pruebas realizadas al sistema en conjunto. Para ello, se ha hecho uso de la interfaz gráfica de **JIRA** tanto para configurar los proyectos como los diferentes casos de prueba.

Cada caso de prueba probará una ejecución completa desde que se selecciona **versión** y **proyecto** hasta que éstos se **persisten** en la base de datos de **JIRA** mostrándolos en la pestaña del proyecto en cuestión.

Por comodidad, se han establecido dos casos de prueba de sistema que engloban las siguientes posibilidades:

- Issues con Dependencias.
- Issues con Skills que existen en el proyecto.
- Issues con Skills que no existen en el proyecto.
- Versión Corta (1 semana).
- Versión Larga (1 mes).
- Gran Cantidad de Issues (mayor que 5).
- Gran cantidad de Empleados (mayor que 5).

14.2.1 Definición del conjunto de pruebas.

En la siguiente tabla se mostrarán los diversos Issues y Empleados a probar, así como sus dependencias, skills requeridos y duración.

Para ello, se han definido dos versiones:

- **1.0** Empieza: 2/Jul/2019 Termina: 05/Jul/2019
- **2.0** Empieza: 02/Jul/2019 Termina: 02/Ago/2019

En la siguiente tabla se muestran los Issues con su clave así como los atributos relevantes de cara a **Replan**.

| Issue Key | Duration (hours) | Priority | Required Skills | Depends On | Fix Version(s) |
|-----------|------------------|----------|------------------------|---------------|----------------|
| RPFJ-1 | 8.0 | 5 | Developers FrontEnd | | 1.0 |
| RPFJ-3 | 72.0 | 1 | QA | | 1.0 |
| RPFJ-5 | 1.0 | 2 | Administrators | | 1.0 |
| RPFJ-8 | 8.0 | 3 | QA | | 1.0 |
| RPFJ-10 | 32.0 | 3 | JAVA | | 1.0 |
| RPFJ-11 | 24.0 | 3 | Developers | | 1.0 |
| RPFJ-12 | 1.0 | 4 | | | 1.0 |
| RPFJ-13 | 1.0 | 4 | | | 1.0 |
| RPFJ-14 | 1.0 | 4 | Chef | | 1.0 |
| RPFJ-2 | 80.0 | 2 | Developers UX | | 2.0 |
| RPFJ-4 | 8.0 | 3 | Developers | | 2.0 |
| RPFJ-6 | 4.0 | 3 | | | 2.0 |
| RPFJ-7 | 16.0 | 4 | FrontEnd | | 2.0 |
| RPFJ-9 | 0.5 | 3 | Administrators | | 2.0 |
| RPFJ-15 | 0.5 | 2 | FrontEnd | | 2.0 |
| RPFJ-16 | 16.0 | 4 | UX | RPFJ-7 RPFJ-2 | 2.0 |
| RPFJ-17 | 1.0 | 4 | | | 2.0 |
| RPFJ-18 | 120.0 | 1 | | | 2.0 |
| RPFJ-19 | 200.0 | 1 | | | 2.0 |

Como podemos ver, se han tratado de englobar todos los casos mencionados anteriormente en este apartado.

Ahora se procederá a definir los **Empleados** con los que se ejecutarán las diversas pruebas.

| Worker | Skills |
|--------|------------------------|
| Kael | FrontEnd |
| admin | Administrators |
| Brann | BackEnd |
| Arthas | JAVA QA |
| Ilidan | Developers FrontEnd |

| Worker | Skills |
|----------|------------------|
| Ragnaros | QA |
| Sylvanas | Developers UX |

Se han añadido una serie de empleados. Por definición de **Replan for JIRA**, para que un empleado pertenezca a un proyecto, deberá tener asignado un rol en éste.

14.2.1.1 Caso de Prueba 1. Planificación de Todo el Proyecto

Ya que en **Replan for JIRA**, se ha implementado la opción de planificar todo un proyecto a un plazo de cuatro semanas desde la fecha actual, se va a proceder a planificar todo el proyecto mencionado anteriormente (sin tener en cuenta versiones) con esta restricción.

1) Para ello, los inputs de **Project y Versión** serán: “RPFJ” y “Total Project”, respectivamente.

2) Acto seguido, se comienza la planificación de tal forma que en la siguiente pantalla, se deberían mostrar todos los **Issues y Usuarios** del proyecto.

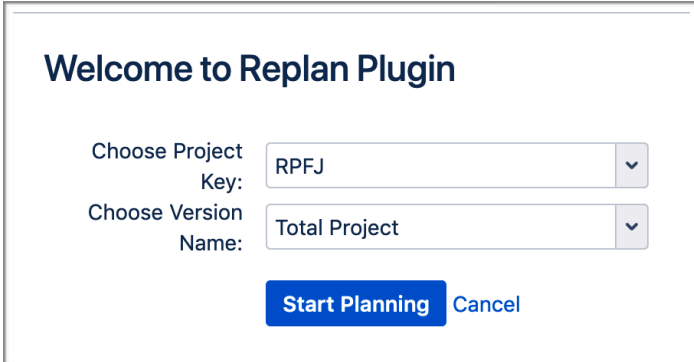


FIG 29. INPUTS CASO DE PRUEBA 1

3) El sistema muestra por pantalla todos los usuarios e **Issues del Proyecto** tal y como se esperaba.

Issues to Plan: 19

| Issue-Key | Duration (hours) | Priority Level | Required Skills | Depends On |
|-----------|------------------|----------------|--------------------|-----------------|
| RPFJ-19 | 200.0 | 1 | No skills required | No dependencies |
| RPFJ-18 | 120.0 | 1 | No skills required | No dependencies |
| RPFJ-17 | 1.0 | 4 | No skills required | No dependencies |
| RPFJ-16 | 16.0 | 4 | UX | RPFJ-7 RPFJ-2 |
| RPFJ-15 | 0.5 | 2 | FrontEnd | No dependencies |
| RPFJ-14 | 1.0 | 4 | Chef | No dependencies |
| RPFJ-13 | 1.0 | 4 | No skills required | No dependencies |
| RPFJ-12 | 1.0 | 4 | No skills required | No dependencies |
| RPFJ-11 | 24.0 | 3 | Developers | No dependencies |
| RPFJ-10 | 32.0 | 3 | JAVA | No dependencies |
| RPFJ-9 | 0.5 | 3 | Administrators | No dependencies |

FIG 30. FRAGMENTO DE LA VISTA DE LISTADO DE ISSUES A PLANIFICAR

4) Se selecciona la opción **Get Plan**.

5) El sistema debería mostrar una planificación válida así como los **Issues** que no se hayan podido planificar.

Plan for the whole project

| All | Kael | admin | Brann | Arthas | Ilidan | Ragnaros | Sylvanas | | | | | | | | | | | |
|-----------------|------|------------------------------|-------|-----------|--------|-----------|----------|-----------|--|-----------|--|----------|--|----------|--|----------|--|----------|
| Employee | | 25/jun/19 | | 26/jun/19 | | 27/jun/19 | | 28/jun/19 | | 29/jun/19 | | 2/jul/19 | | 3/jul/19 | | 4/jul/19 | | 5/jul/19 |
| Kael | | RPFJ-18 | | RPFJ-18 | | RPFJ-18 | | RPFJ-18 | | RPFJ-18 | | RPFJ-18 | | RPFJ-18 | | RPFJ-18 | | RPFJ-18 |
| admin | | RPFJ-6 RPFJ-5 RPFJ-9 | | | | | | | | | | | | | | | | |
| Brann | | RPFJ-12 | | | | | | | | | | | | | | | | |
| Arthas | | RPFJ-10 | | RPFJ-10 | | RPFJ-10 | | RPFJ-10 | | | | | | | | | | |
| Ilidan | | RPFJ-11 | | RPFJ-11 | | RPFJ-11 | | RPFJ-4 | | RPFJ-1 | | | | | | | | |
| Ragnaros | | RPFJ-3 | | RPFJ-3 | | RPFJ-3 | | RPFJ-3 | | RPFJ-3 | | RPFJ-3 | | RPFJ-3 | | RPFJ-3 | | RPFJ-3 |
| Sylvanas | | RPFJ-13 RPFJ-17 RPFJ-2 | | RPFJ-2 | | RPFJ-2 | | RPFJ-2 | | RPFJ-2 | | RPFJ-2 | | RPFJ-2 | | RPFJ-2 | | RPFJ-2 |

⚠️ Couldn't plan 2 features

| Issue-Key | Duration (hours) | Priority Level | Required Skills | Depends On |
|-----------|------------------|----------------|--------------------|-----------------|
| RPFJ-19 | 200.0 | 1 | No skills required | No dependencies |
| RPFJ-14 | 1.0 | 4 | Chef | No dependencies |

Persist this plan
Cancel

FIG 30. FRAGMENTO DE LA VISTA DE PLANIFICACIÓN

6) Se selecciona la opción de persist this plan.

7) El sistema persiste toda la planificación y muestra un mensaje de éxito.

8) Se comprueba en, al menos un Issue que está planificación es correcta. En este caso se comprueba que el RPFJ-18 esté asignado a **Kael** y su **fecha de entrega** sea el **13 de Julio**.



Longest Task Ever!

[Editar](#)
[Comentar](#)
[Asignar](#)
[Más](#)
[Iniciar Progreso](#)
[Listo](#)
[Administración](#)
[Exportar](#)

Detalles

Tipo: Tarea Estado: **POR HACER**

Prioridad: ↑ Highest (Ver Flujo de Trabajo)

Versión(es): Ninguno Resolución: Sin resolver

Afectada(s): Versión(es): 2.0

Etiquetas: Ninguno Correctora(s):

Personas

Responsable: Kael Thas Sunstrider
[Asignarme a mí](#)

Informador: admin

Votos: 0

Interesados: 1 [Dejar de observar esta incidencia](#)

Descripción
Haga clic para añadir una descripción

Adjuntos ...

Suelte los archivos para adjuntarlos o [explorar](#).

Fechas

A entregar: 13/jul/19

Creada: Hace 2 horas

Actualizada: Hace 1 minuto

FIG 31. PERSISTENCIA EN EL ISSUE RPFJ-18

Como se puede observar en la imagen, el responsable ahora es Kael y su fecha a *entregar* es el día correcto.

14.2.1.2 Caso de Prueba 2. Planificación de la versión 1.0

Version: 1.0

| Start Date | End Date |
|-----------------------|-----------------------|
| 2019-07-01 00:00:00.0 | 2019-07-05 00:00:00.0 |

Issues to Plan: **9**

| Issue-Key | Duration (hours) | Priority Level | Required Skills | Depends On |
|-------------------------|------------------|----------------|------------------------|-----------------|
| RPFJ-14 | 1.0 | 4 | Chef | No dependencies |
| RPFJ-13 | 1.0 | 4 | No skills required | No dependencies |
| RPFJ-12 | 1.0 | 4 | No skills required | No dependencies |
| RPFJ-11 | 24.0 | 3 | Developers | No dependencies |
| RPFJ-10 | 32.0 | 3 | JAVA | No dependencies |
| RPFJ-8 | 8.0 | 3 | QA | No dependencies |
| RPFJ-5 | 1.0 | 2 | Administrators | No dependencies |
| RPFJ-3 | 72.0 | 1 | QA | No dependencies |
| RPFJ-1 | 8.0 | 5 | Developers FrontEnd | No dependencies |

FIG 32. LISTADO DE ISSUES A PLANIFICAR EN VERSIÓN 1.0

Este caso de prueba es similar al anterior con la variación de que en lugar de planificarse todo el proyecto se procederá a planificar solamente la versión **1.0**.

Para ver que este caso se ha ejecutado correctamente, se mostrarán las imágenes de las salidas a los pasos 3) y 5) del caso anterior.

Plan for Version: 1.0 2019-07-01 00:00:00.0 2019-07-05 00:00:00.0

All [Kael](#) [admin](#) [Brann](#) [Arthas](#) [Ilidan](#) [Ragnaros](#) [Sylvanas](#)

| Employee | 1/jul/19 | 2/jul/19 | 3/jul/19 | 4/jul/19 | 5/jul/19 |
|-----------------|----------|----------|----------|----------|----------|
| Kael | RPFJ-13 | | | | |
| admin | RPFJ-5 | | | | |
| Brann | RPFJ-12 | | | | |
| Arthas | RPFJ-10 | RPFJ-10 | RPFJ-10 | RPFJ-10 | |
| Ilidan | RPFJ-1 | | | | |
| Ragnaros | RPFJ-8 | | | | |
| Sylvanas | RPFJ-11 | RPFJ-11 | RPFJ-11 | | |

⚠ Couldn't plan 2 features

| Issue-Key | Duration (hours) | Priority Level | Required Skills | Depends On |
|-------------------------|------------------|----------------|-----------------|-----------------|
| RPFJ-14 | 1.0 | 4 | Chef | No dependencies |
| RPFJ-3 | 72.0 | 1 | QA | No dependencies |

FIG 33. PLANIFICACIÓN DE LA VERSIÓN 1.0

Como podemos observar más detalladamente, el **RPFJ-3** no se planificará debido a su larga duración. El **RPFJ-14** tampoco, al no existir ningún **employee** que sea **Chef**.

14.2.1.3 Caso de Prueba 3. Planificación de la versión 2.0

Este caso de prueba es similar a los anteriores pero esta vez se realizará con la versión **2.0**.

Version: 2.0

| Start Date | End Date |
|-----------------------|-----------------------|
| 2019-07-02 00:00:00.0 | 2019-08-02 00:00:00.0 |

Issues to Plan: **10**

| Issue-Key | Duration (hours) | Priority Level | Required Skills | Depends On |
|-----------|------------------|----------------|--------------------|-----------------|
| RPFJ-19 | 200.0 | 1 | No skills required | No dependencies |
| RPFJ-18 | 120.0 | 1 | No skills required | No dependencies |
| RPFJ-17 | 1.0 | 4 | No skills required | No dependencies |
| RPFJ-16 | 16.0 | 4 | UX | RPFJ-7 RPFJ-2 |
| RPFJ-15 | 0.5 | 2 | FrontEnd | No dependencies |
| RPFJ-9 | 0.5 | 3 | Administrators | No dependencies |
| RPFJ-7 | 16.0 | 4 | FrontEnd | No dependencies |
| RPFJ-6 | 4.0 | 3 | No skills required | No dependencies |
| RPFJ-4 | 8.0 | 3 | Developers | No dependencies |
| RPFJ-2 | 80.0 | 2 | Developers UX | No dependencies |

FIG 34. LISTADO DE ISSUES A PLANIFICAR EN VERSIÓN 2.0

Plan for Version: 2.0 2019-07-02 00:00:00.0 2019-08-02 00:00:00.0

All [Kael](#) [admin](#) [Brann](#) [Arthas](#) [Ilidan](#) [Ragnaros](#) [Sylvanas](#)

| Employee | 2/jul/19 | 3/jul/19 | 4/jul/19 | 5/jul/19 | 6/jul/19 | 9/jul/19 | 10/jul/19 | 11/jul/19 | 12/jul/19 |
|-----------------|-------------------------|----------|----------|----------|----------|----------|-----------|-----------|-----------|
| Kael | RPFJ-15 | | | | | | | | |
| admin | RPFJ-9 | | | | | | | | |
| Brann | RPFJ-17 | | | | | | | | |
| Arthas | | | | | | | | | |
| Ilidan | RPFJ-18 | RPFJ-18 | RPFJ-18 | RPFJ-18 | RPFJ-18 | RPFJ-18 | RPFJ-18 | RPFJ-18 | RPFJ-18 |
| Ragnaros | RPFJ-19 | RPFJ-19 | RPFJ-19 | RPFJ-19 | RPFJ-19 | RPFJ-19 | RPFJ-19 | RPFJ-19 | RPFJ-19 |
| Sylvanas | RPFJ-4 | RPFJ-2 | RPFJ-2 | RPFJ-2 | RPFJ-2 | RPFJ-2 | RPFJ-2 | RPFJ-2 | RPFJ-2 |

[Persist this plan](#) [Cancel](#)

FIG 35. FRAGMENTO DE LA PLANIFICACIÓN EN LA VERSIÓN 2.0

14.3 Errores Detectados a través de las diferentes pruebas.

En este apartado se va a dar lugar a una tabla que permitirá visualizar rápidamente qué errores se han detectado y corregido.

| Fuente | Error | Corregido |
|------------------|--|--|
| Replan Optimizer | Los features con poca duración o 0.0, se planifican erróneamente. | Se ha aplicado un tiempo mínimo de 0.5 horas en caso de que no esté definido. |
| Replan Optimizer | Si existe una dependencia con un feature que no está en el proyecto, no lo planifica. | No |
| JIRA Core | Permite crear versiones cuya fecha de inicio es mayor que la fecha de publicación. | En este caso, se establecerá la fecha de inicio a la fecha actual para que Replan pueda ejecutarse correctamente. |
| Replan for JIRA | Los calendarios creados a partir de una versión a medias son erróneos. | En el caso de que una versión esté a medias, se establecerá la fecha de inicio como la fecha actual para generar el calendario. |
| Replan for JIRA | Permite planificar versiones obsoletas. | En el caso de que una versión esté obsoleta, devolverá un mensaje de error. |
| Replan for JIRA | No muestra la planificación de una versión. | Sí |
| Replan for JIRA | Si existe el proyecto pero no la versión introducida, da un error no gestionado. | Se ha procedido a gestionar el error. Ahora el sistema solicita que introduzcas una versión correcta. |
| Replan for JIRA | Si una versión está ya publicada la planifica. | En el caso de que una versión esté publicada, devuelve un mensaje de error indicándolo. |
| Replan for JIRA | La vista de planificación devolvía las fechas erróneamente. | Se procedió a separar la lógica de establecimiento de fechas a las clases DaySlot y ReplanToJiraConverter para desacoplarla y que éste funcione correctamente. |
| Replan for JIRA | El formato de Replan Optimizer y JIRA de fechas no es el mismo y devolvía error a la hora de trabajar con éstas. | Sí. |
| Replan for JIRA | La duración estaba establecida en formatos diferentes y devolvía planificaciones incoherentes. | Sí. ReplanToJiraConverter realiza correctamente esta transformación. |
| Replan for JIRA | Diversas excepciones no estaban controladas. | Sí. |

15. Despliegue

A lo largo del desarrollo del proyecto la herramienta ha sido desplegada en *la máquina local*.

Por otro lado, como se ha mencionado en otros apartados se ha empleado el **end-point** donde está desplegado el **Replan Optimizer** para poder trabajar con él.

El despliegue se ha realizado de esta forma ya que desplegar **JIRA** en un servidor externo resulta especialmente complicado y no se ha dispuesto de un servidor con la suficiente potencia y/o funcionalidades para poder desplegar todo el **Software Requerido**.

En el caso de que se quiera desplegar esta herramienta en alguna de las máquinas junto con **JIRA**, sólo hay que seguir la guía de **Atlassian** para ello [3].

16. Legislación Sobre el proyecto

En este apartado se procederán a tratar todos aquellos aspectos que pudiesen afectar al proyecto a nivel legislativo, así como las diversas leyes de cara a la *Protección de Datos* así como otros aspectos legales.

16.1 Aspectos Relevantes del Proyecto

Este proyecto, procederá a utilizar información de los diversos proyectos que haya establecidos en la plataforma de JIRA en la que se instale el plugin.

Cabe destacar que no trabajará ni persistirá información de carácter personal o que pudiese identificar a algún usuario con su identidad real.

Se trabajará exclusivamente con los datos que ofrece la plataforma de JIRA, asumiendo que serán los dueños de la plataforma en la que se instale los encargados de proteger esta información puesto que desde ReactivePlan for JIRA no se utilizarán estos datos para absolutamente nada.

16.2 Aplicación de la GDPR por parte de JIRA Atlassian.

De cara a desarrollar este proyecto, se tendrá en cuenta la ley GDPR [16] que es la actual ley **Europea** de cara a la protección de datos.

Para ello, los desarrolladores de Atlassian han establecido un acuerdo [17] que se cumplirá de cara a la realización de este proyecto.

Debido a que, en este proyecto no se tratan de forma directa datos de carácter personal, tal y como se ha mencionado en el apartado anterior se delega toda la responsabilidad sobre la aplicación de esta ley a **Atlassian** así como sus algoritmos de cifrado y políticas de protección de datos.

Por último hay que aclarar que el **Replan Optimizer** no persiste datos en ninguna de sus configuraciones, y además tampoco trabajará con datos de carácter personal

Por definición propia del proyecto y al no tramitarse datos de carácter personal, esta ley no afectará de forma directa al proyecto, pero sí de forma indirecta al ser un **plugin** para ésta plataforma de Atlassian.

17. Conclusión

En esta última sección se va a proceder a relatar la conclusión de todo el proceso de la elaboración de un trabajo de final de grado *ReactivePlan for JIRA*.

17.1 Cumplimiento de Objetivos

A la hora de elaborar un trabajo de fin de grado, cada estudiante debe cumplir una serie de objetivos propuestos en función del tipo de trabajo que se vaya a elaborar. En el caso de este proyecto, el cumplir los objetivos ha resultado una tarea sencilla e indirecta puesto que para dar lugar a un *buen desarrollo* ha sido necesario tener unos amplios conocimientos en las materias que se proponen así como aplicarlas de la forma más racional y eficiente, con lo cual cada uno de los objetivos se ha cumplido exitosamente.

CES1.1: Desarrollar, mantener y evaluar sistemas y servicios software complejos y/o críticos.

Este objetivo se cumplió en el momento en el que toda la herramienta fue diseñada desde un principio para que fuese totalmente escalable y mantenible tal y como se propone en el apartado de diseño. Por otro lado, se ha aprendido muchísimo sobre el desarrollo de software en sistemas complejos ya que JIRA no es una plataforma especialmente sencilla, y, lidiar con todos los inconvenientes que ha producido la implementación de un plugin en una herramienta que inicialmente era desconocida no ha sido tarea fácil sin embargo, a través de leer diferentes proyectos relacionados y consultar las dudas más técnicas con la comunidad de Atlassian se ha podido desarrollar un sistema que funciona correctamente.

CES1.2: Dar solución a problemas de integración en función de las estrategias, de los estándares y de las tecnologías disponibles.

La integración de sistemas ha resultado una tarea totalmente compleja ya que, la documentación tanto de la plataforma JIRA como la de Replan era más bien escasa. Es por ello que se ha tenido que realizar *un poco* de ingeniería inversa a estas dos herramientas para poder averiguar cuál era realmente su funcionamiento y complementar la falta de documentación existente.

Por otro lado, la utilización de un *SDK* y la verdadera dificultad para inyectar componentes de JIRA ha dado paso a un aprendizaje extenso sobre cómo desarrollar una herramienta teniendo un conocimiento parcial sobre el funcionamiento de estos componentes ha servido para ser capaz de realizar correctamente una gestión de interfaces y ser capaz de desacoplar el funcionamiento de estos otros componentes con el de la herramienta a desarrollar.

Por último, la utilización constante de una API Rest como es la de Replan Optimizer y tener que realizar diversas pruebas de integración para ver que aquello que se había estado desarrollando funcionaba correctamente y no daba ningún error debido a una mala integración o mala conversión de datos ha sido una tarea compleja que se ha resuelto a través de cientos de pruebas y llamadas a esta API así como reuniones con los desarrolladores. Este objetivo se ha cumplido correctamente.

CES1.3: Identificar, evaluar y gestionar los potenciales riesgos asociados a la construcción de software que se pudiesen presentar.

Afortunadamente, los principales riesgos de este proyecto se definieron al principio del desarrollo con lo que no fue difícil tenerlos en cuenta. La compleja documentación de JIRA o la escasa del Replan Optimizer (así como los fallos que este pudiera tener) se han tenido en cuenta desde el principio con lo cuál no han causado grandes desviaciones a lo largo del proyecto. Claramente este objetivo se ha cumplido con rotundo éxito.

CES1.4: Desarrollar, mantener y evaluar servicios y aplicaciones distribuidas con soporte de la red.

De cara a satisfacer este objetivo lo que hay que tener en cuenta ha sido el hecho de que sólo se ha conectado al Replan Optimizer. Para que esta herramienta sea desarrollarle y mantenible se ha separado totalmente la lógica de la conexión al optimizer con la del resto del sistema. Por otro lado es importante tener en cuenta que de todas las funcionalidades respecto a la red se han encargado principalmente JIRA y las diferentes librerías utilizadas, de modo que este objetivo también se ha cumplido correctamente.

CES1.7: Controlar la calidad y diseñar pruebas en la producción de software.

Como se estableció en la planificación inicial, la mayor parte de las pruebas se destinaron al final del desarrollo puesto que era lo que tenía más sentido probar en esta herramienta al no tener funcionalidades (a nivel individual) de forma compleja. Sin embargo, tal y como se menciona en el apartado de documentación se han ido desarrollando pruebas a lo largo de todo el proyecto para probar cada una de las funcionalidades y asegurar que funciona proyectadamente. Al no existir errores en el sistema se puede afirmar que este objetivo se ha cumplido con éxito.

CES2.1: Definir y gestionar los requisitos de un sistema de software.

Definir requisitos a lo largo del desarrollo del sistema ha sido especialmente complejo debido a que según se iba adquiriendo conocimiento sobre la herramienta, empezaban a mostrarse las diferentes posibilidades sobre las que se podía actuar en JIRA. Estos requisitos, tal y como se definieron en la metodología de trabajo se han obtenido y verificado a través de las diferentes reuniones con el tutor, y lo que se puede ver en este documento es la versión final de los requisitos del sistema. Se puede decir que este objetivo se ha cumplido satisfactoriamente.

17.2 Trabajo Futuro

Debido a que el tiempo disponible para realizar un proyecto de fin de grado, hay diversas funcionalidades que no han podido ser implementadas con lo cual se eliminaron de los requisitos del proyecto, pero sería ideal aclararlas por si se continuase el desarrollo de esta herramienta.

Entre ellas, está la capacidad de replanificar, es decir, obtener un plan ya hecho a través de una versión y replanificarlo para que se ajusta a la situación actual.

Otra del trabajo que queda por hacer, además de mejorar el diseño de la interfaz, sería el de mostrar las *Skills* en casillas personalizadas de JIRA así como permitir ver la planificación actual.

18. Conocimientos Empleados

En este apartado se procederá a exponer todos los conocimientos empleados durante la realización de este proyecto así como la asignatura del grado en las que éstos conocimientos han sido adquiridos.

18.1 Programación

De cara a elaborar lo requerido durante el desarrollo del plugin de JIRA, se han empleado diversos conocimientos de programación afín de poder desarrollar el código de la forma más eficiente, mantenible, reescalable y elegante posible.

18.1.1 Programación Orientada a Objetos

De cara a poder programar los diferentes *objetos* que aparecen a lo largo del código, así como el desarrollo en JAVA se han empleado los conocimientos adquiridos de cara a la Programación Orientada a Objetos.

Entre los conocimientos empleados de esta disciplina, podemos destacar:

- Creación de un objeto, así como sus constructores.
- Instanciación de un objeto.
- Comparación entre objetos.
- Getters y setters.
- Uso de mapas y estructuras de datos básicas.

18.1.2 Estructura de Datos

Con el fin de emplear las estructuras de datos correctas según se necesite, así como tratar estructuras de datos más complejas se han empleado los conocimientos adquiridos de cara a las Estructuras de Datos.

Entre los conocimientos empleados de esta disciplina, podemos destacar:

- Búsqueda en cualquier estructura.
- Funcionamiento de las diversas estructuras de datos.
- Optimización de las estructuras.
- Lógica adquirida de cara a las estructuras de datos.

18.2 Tecnologías

De cara al desarrollo del proyecto se han empleado múltiples tecnologías

18.2.1 Tecnologías Web

Con el fin de conocer cómo funcionan las tecnologías web y el intercambio básico de datos se han empleado conocimientos adquiridos de cara a las Tecnologías Web.

Entre los conocimientos empleados de esta disciplina, podemos destacar:

- Uso de la tecnología de JAVA Servlets.
 - Métodos de HTTP Requests.
 - Gestión de parámetros y atributos.
-

18.2.2 Aplicaciones y Servicios Web

Con el fin de poder desarrollar correctamente el proyecto se han empleado conocimientos de Servicios Web, especialmente, aunque al ser JIRA una aplicación web, también se ha tenido en cuenta la parte de aplicaciones.

Entre los conocimientos empleados de esta disciplina, podemos destacar:

- Conocimiento de las estructuras de mensajes para intercambiar datos (XML, JSON, SOAP).
- Qué es un servicio web.
- Cómo funciona una API REST.
- Cómo enviar o recibir parámetros a un servicio web.

18.3 Análisis Funcional y Diseño

De cara a realizar el análisis funcional de lo que se pretende realizar en este proyecto, así como el diseño, se han empleado conocimientos de modelado, análisis, y requisitos.

18.3.1 Modelado y Diseño de Software

Afin de realizar un correcto diseño de las diferentes clases que interactúan en la aplicación resultando en que estas sigan los patrones de diseño correcto así como asegurar la mantenibilidad y reescalabilidad del código, se han empleado diversos conocimientos de modelado y diseño de software.

Entre los conocimientos empleados de esta disciplina, podemos destacar:

- Entendimiento y creación de Diagramas UML.
- Empleo de patrones de diseño.
- Diseño funcional de cara a desarrollar código mantenible.
- Diseño arquitectónico de cara a la implementación del código.

18.3.2 Ingeniería de Requisitos

De cara a realizar tareas como el análisis del contexto así como diseñar las diferentes funcionalidades de la plataforma, se han empleado diversos conocimientos de la disciplina de Ingeniería de Requisitos.

Entre los conocimientos empleados de esta disciplina, podemos destacar:

- Obtención de Requisitos.
- Definición de Casos de Uso.
- Análisis del Contexto.

18.3.3 Interfaces de Usuario

De cara a poder crear las diversas interfaces de usuario de tal forma que tengan una calidad mínima así como poder asegurar su calidad y su usabilidad.

18.4 Gestión Proyectos

De cara a desarrollar satisfactoriamente este proyecto, se han empleado diversos conocimientos sobre proyectos de software adquiridos.

18.4.1 Gestión de Proyectos

Los conocimientos de gestión de proyectos se han puesto en práctica a la hora de definir tareas así como realizar una correcta gestión del tiempo.

Entre los conocimientos empleados de esta disciplina, podemos destacar:

- Gestión de proyectos ágiles.
- Definición de épicas e historias de usuario.
- Definición de tareas.
- Planificación temporal.

18.5 Pruebas de Software

18.5.1 Mantenimiento y Pruebas de Software

Con el fin de asegurar el correcto funcionamiento del software desarrollado se han puesto en práctica diversos conocimientos sobre pruebas, así como la capacidad de asegurar el mantenimiento de este software a través de las pruebas.

Entre los conocimientos empleados de esta disciplina, podemos destacar:

- Pruebas unitarias.
- Pruebas de sistema.
- Pruebas de integración.

Glosario

Add-On: Añadido a una herramienta o plataforma.

Andamiaje: Estructura del código, carpetas o cualquier otro elemento relacionado con la informática.

CSRP: Abreviación de *Continuous Software Release Planning*.

Dependencia: Cuando una parte del código depende de otra.

Desarrollo: Todo el proceso de creación de software.

GEP: Asignatura de gestión de proyectos.

Implementación: Proceso en el que se escribe el código para desarrollar el software en cuestión.

Plugin: Herramienta externa anexionada a un sistema principal.

Replan: Abreviación de ReactivePlan
software

REST: Abreviación de *Representational State Transfer*.

SDK: Abreviación de Software Development Kit.

De/Serialización: Transformación de objetos de Java a JSON.

Sistema: Definimos por sistema al proyecto que se ha desarrollado.

Stakeholder: Hace referencia a cualquier interesado en un proyecto.

Third-Party Extension: Es aquella que no ha desarrollado la propia compañía para su producto sino que la realiza un agente externo.

TIC: Cualquier cosa relacionada con las tecnologías de la información.

Bibliografía

- [1] SUPERSEDE. *Supersede* [en línea]. [Consultada: 26 de Febrero de 2019]. Disponible en: < <https://www.supersede.eu/> >
- [2] D. Ameller, C. Farré, X. Franch, A. Cassarino, D. Valerio y V. Elvassore “Replan: A release planning tool”, Universitat Politècnica de Catalunya, Barcelona y Siemens AG Österreich, Austria.
- [3] ATlassian. Atlassian [en línea]. [Consultada: 26 de Febrero de 2019]. Disponible en: < <https://www.atlassian.com> >
- [4] Trello. Trello [en línea]. [Consultada: 26 de Febrero de 2019]. Disponible en: < <https://trello.com> >
- [5] Anàlisi de l'activitat dels desenvolupadors en plataformes de suport al desenvolupament col·laboratiu. M. Xamaní Moreno [en línea]. [Consultada: 26 de Febero de 2019]. Disponible en: < <https://upcommons.upc.edu/handle/2117/117991> >
- [6] JIRA. Wikipedia [en línea]. [Consultada: 26 de Febrero de 2019]. Disponible en: < <https://es.wikipedia.org/wiki/JIRA> >
- [7] Ameller, D., Cassarino, A., Elvassore, V., Farré, C., Franch, X., Valerio, D. *Replan architecture* [Figura]. [Consultado el 26 de Febrero de 2019]. Disponible en: < <http://www.essi.upc.edu/dameller/publicacions/ameller2017-saner-demo.pdf> >
- [8] Trabajo de Fin de Grado. Facultad de Informática de Barcelona [en línea]. [Consultada: 3 de Marzo de 2019]. Disponible en: < <https://www.fib.upc.edu/es/estudios/grados/grado-en-ingenieria-informatica/trabajo-de-fin-de-grado> >
- [8] Crédito Académico. Wikipedia [en línea]. [Consultada: 3 de Marzo de 2019]. Disponible en: < https://es.wikipedia.org/wiki/Crédito_académico >
- [9] Departament d'Organization de Empreses, “Modul 2.6 el Informe de Sostenibilidad 2018”, Universitat Politècnica de Catalunya, Barcelona.
- [10] Salario Promedio Barcelona. Payscale [en línea]. [Consultada: 5 de Marzo de 2019]. Disponible en: < <https://www.payscale.com> >
- [11] Documentación Swagger de Replan Controller. Github [en línea]. [Consultada: 16 de Mayo de 2019]. Disponible en: < https://github.com/2016LLAV00009/replan_controller/blob/master/local_swagger.yaml >

- [12] Documentación Swagger de Replan Optimizer. Github [en línea]. [Consultada: 16 de Mayo de 2019]. Disponible en: < https://github.com/2016LLAV00009/replan_optimizer_v2/blob/master/swagger.yaml >
- [13] Jira API REST . Atlassian [en línea]. [Consultada: 16 de Mayo de 2019]. Disponible en: < <https://developer.atlassian.com/server/jira/platform/rest-apis/> >
- [14] REST Java Client for Jira. Atlassian Marketplace [en línea]. [Consultada: 16 de Mayo de 2019]. Disponible en: < <https://marketplace.atlassian.com/apps/39474/rest-java-client-for-jira> >
- [15] REST Jira Server API [en línea]. [Consultada: 16 de Mayo de 2019]. Disponible en: < <https://docs.atlassian.com/software/jira/docs/api/8.1.0/> >
- [15] REST Jira Developers Forum [en línea]. [Consultada: 16 de Mayo de 2019]. Disponible en: < <https://community.developer.atlassian.com/c/Jira> >
- [16] General Data Protection Regulation [en línea]. [Consultada: 23 de Mayo de 2019]. Disponible en: < <https://gdpr-info.eu> >
- [17] Atlassian's GDPR Commitment [en línea]. [Consultada. 23 de Mayo de 2019]. Disponible en: < <https://www.atlassian.com/trust/privacy/gdpr#faq-bfe28c97-463a-46e9-acad-eca1f476ec89> >
- [18] Jira User's Guide [en línea]. [Consultada 18 de Junio de 2019]. Disponible en: < <https://confluence.atlassian.com/jira064/jira-user-s-guide> >
- [19] JIRA Agile tools for software teams[en línea]. [Consultada 18 de Junio de 2019]. Disponible en: < <https://www.atlassian.com/software/jira/agile> >
- [20] JIRA Architecture [en línea]. [Consultada 21 de Junio de 2019]. Disponible en: < <https://developer.atlassian.com/server/jira/platform/architecture-overview/> >
- [21] Web Item JIRA Module [en línea]. [Consultada 21 de Junio de 2019]. Disponible en: < <https://developer.atlassian.com/server/jira/platform/web-item/> >
- [21] Velocity Apache [en línea]. [Consultada 23 de Junio de 2019]. Disponible en: < <https://velocity.apache.org> >
- [22] Atlassian SDK [en línea]. [Consultada 23 de Junio de 2019]. Disponible en: < <https://developer.atlassian.com/server/framework/atlassian-sdk/install-the-atlassian-sdk-on-a-linux-or-mac-system/> >
- [23] Postman [en línea]. [Consultada 23 de Junio de 2019]. Disponible en: < <https://www.getpostman.com> >