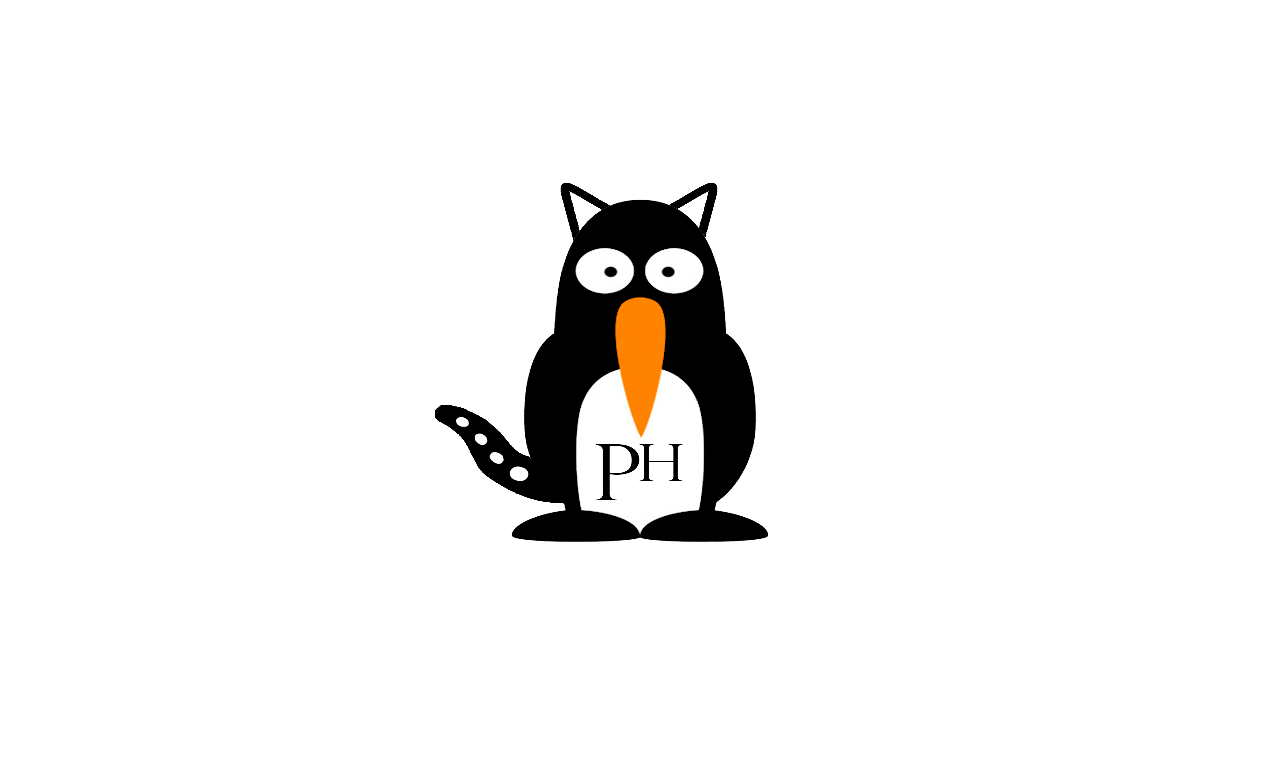
**PlaceHolder**Arquitectura e Integración de Sistemas Software

Grado de Ingeniería del Software

Curso 2º

Autores (por orden alfabético):

**José Gamaza Díaz** (joviproductions@gmail.com)

**Nicolás Pazos Sardella** (nicopolinis99@gmail.com)

**Francisco Javier Rosa Rodríguez** (pacorosa99@gmail.com)

**Iván Santos Domínguez** (ivansd.99@gmail.com)

Tutor: Antonio Gámez Díaz

Número de grupo: 4

Enlace de la aplicación: http://placeholder-aiss.appspot.com/

Enlace de proyecto en GitHub: https://github.com/ivasandom/AISS-Project-PlaceHolder

Historial de versiones

|  |  |  |  |
| --- | --- | --- | --- |
| Fecha | Versión | Detalles | Participantes |
| 17/03/2019 | 1.0 | - Incluye introducción, prototipos de las interfaces de usuario y diagramas UML de componentes y despliegue. | José  Nicolás  Francisco Javier  Iván |
| 15/04/2019 | 1.1 | - Incluye la mejora de la versión 1.0 y además los demás diagramas necesarios. | José  Nicolás  Francisco Javier  Iván |
| 17/04/2019 | 1.4 | - Se añade un editor para los repositorios de GitHub, GitLab y Bitbucket. | José |
| 23/04/2019 | 1.5 | - Se cambia Trello por Todoist a raíz del problema con la implementación de Oauth1 de Trello. | José  Nicolás  Francisco Javier  Iván |
| 24/04/2019 | 1.6 | - Incluye el prototipo funcional de la  aplicación utilizando las distintas APIs  expuestas en los diagramas. | José  Nicolás  Francisco Javier  Iván |
| 26/04/2019 | 1.8 | - Incluye documentación de API propia además de su implementación en Swagger. | Iván |
| 28/04/2019 | 2.0 | - Incluye la revisión del código y arreglados problemas con la implementación de Todoist. | José  Nicolás  Francisco Javier  Iván |
| 17/05/2019 | 2.5 | - Añadido Harvest y enlazado con Todoist para tener una mejor gestión de proyectos con más opciones. | José |
|  |  |  |  |

Índice

[1 Introducción 4](#_Toc7272950)

[1.1 Aplicaciones integradas 4](#_Toc7272951)

[1.2 Evolución del proyecto 4](#_Toc7272952)

[2 Prototipos de interfaz de usuario 5](#_Toc7272953)

[2.1 Vista Index 5](#_Toc7272954)

[2.2 Vista listaTareas.jsp (sin repositorios añadidos) 5](#_Toc7272955)

[2.3 Vista listaTareas.jsp (con repositorios añadidos) 6](#_Toc7272956)

[2.4 Vista error.jsp 7](#_Toc7272957)

[3 Arquitectura 8](#_Toc7272958)

[3.1 Diagrama de componentes 8](#_Toc7272959)

[3.2 Diagrama de despliegue 8](#_Toc7272960)

[3.3 Diagrama de secuencia de alto nivel 9](#_Toc7272961)

[3.4 Diagrama de clases 9](#_Toc7272962)

[3.5 Diagramas de secuencia 10](#_Toc7272963)

[4 Implementación 17](#_Toc7272964)

[5 Pruebas 18](#_Toc7272965)

[6 Manual de usuario 19](#_Toc7272966)

[6.1 Mashup 19](#_Toc7272967)

[6.2 API REST 19](#_Toc7272968)

# Introducción

//PONER LA INTRODUCCIÓN.

## Aplicaciones integradas

|  |  |
| --- | --- |
| Nombre aplicación | URL documentación API |
| GitHub | https://developer.github.com/v3/ |
| GitLab | https://docs.gitlab.com/ee/api/ |
| Bitbucket | https://developer.atlassian.com/bitbucket/api/2/reference/ |
| Todoist | https://developer.todoist.com/rest/v8/ |
| Harvest | https://api.harvestapp.com/v2/ |

Tabla 1. Aplicación integradas

## Evolución del proyecto

1ª Revisión: Hemos tenido que hacer un cambio radical en cuanto a la idea del proyecto ya que lo que teníamos pensado era demasiado tedioso y difícil de conseguir teniendo en cuenta el tiempo que disponemos. Las APIs que queríamos integrar desde un principio se mantienen.

2ª Revisión: Hemos decidido cambiar Trello por Todoist a raíz de problemas con la implementación de Oauth 1 propio de Trello, pensamos que nos iba a consumir más tiempo del que disponíamos. Hemos implementado un editor para poder editar los repositorios desde nuestra propia aplicación. De momento solo podemos navegar en ellos.

3ª Revisión (Última): Hemos añadido Harvest ya que ofrece más opciones a la hora de gestionar proyectos, y lo hemos enlacado con Todoist para asociar los proyectos de una aplicación y las tareas de la otra. El editor de repositorios implementado ya funciona correctamente.

# Prototipos de interfaz de usuario

## Vista Index

Esta es la página principal donde encontramos la descripción de la aplicación y el enlace que nos lleva a la documentación de la API en Swagger. Al hacer log in en Todoist nos lleva a la siguiente vista.

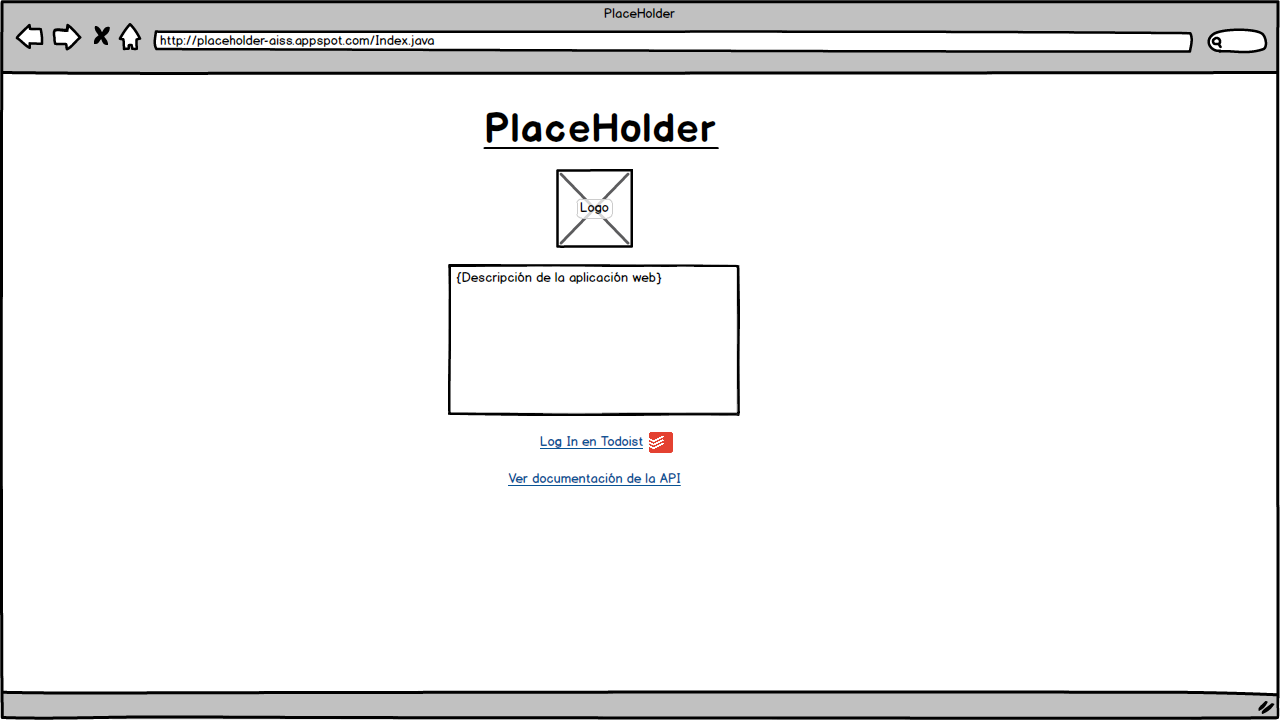


Figura 1. Prototipo de interfaz de usuario de la vista Index

## Vista listaTareas.jsp (sin repositorios añadidos)

En esta vista se muestran los proyectos de Todoist con sus respectivas tareas, sin embargo, si queremos añadir los repositorios de GitHub, GitLab y Bitbucket para una mejor organización sobre qué proyecto corresponde a qué repositorio y también para editarlos tenemos que pulsar en la parte superior derecha el icono de la aplicación donde queremos cargar los repositorios, nos llevará primero al login Oauth de la seleccionada.

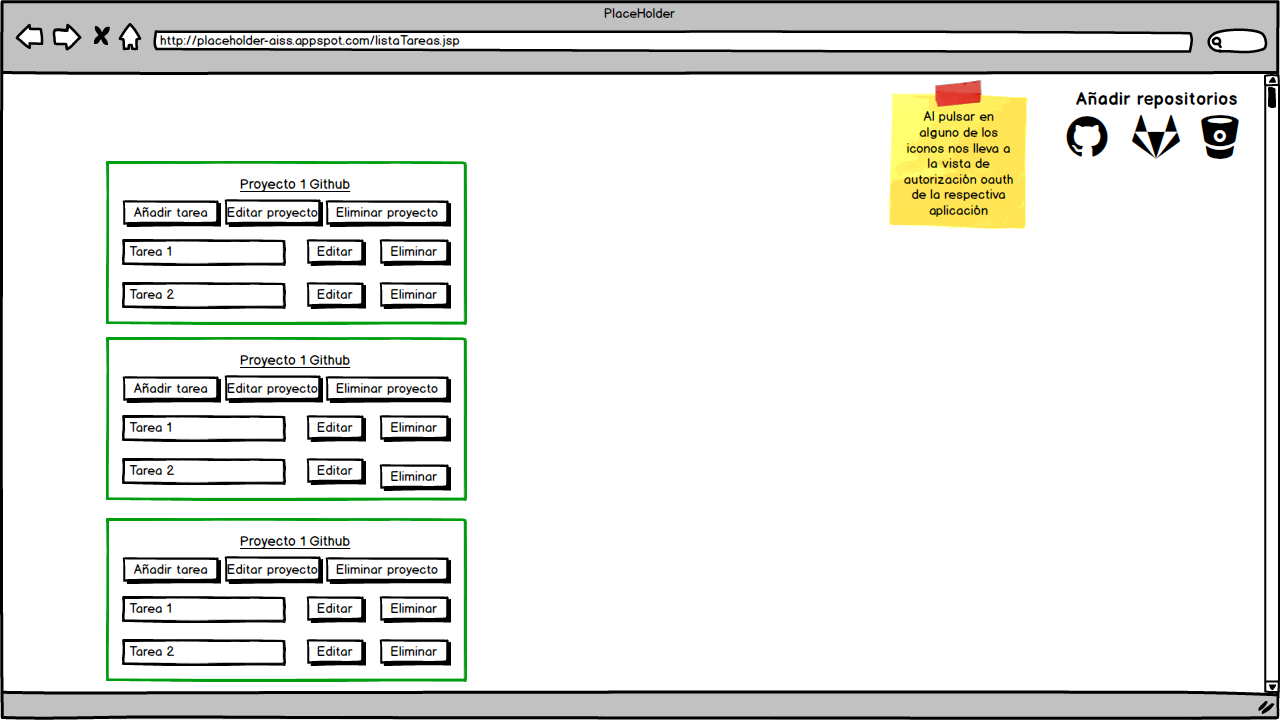


Figura 2. Prototipo de interfaz de usuario de la vista LISTATAREAS.JSP (SIN REPOSITORIOS AÑADIDOS)

## Vista listaTareas.jsp (con repositorios añadidos)

Cuando nos hayamos logueado en la aplicación seleccionada nos volverá a llevar automáticamente a la misma vista listaTareas pero saldrá una lista con los repositorios de esa aplicación y la opción de editarlos con el editor que hemos integrado.

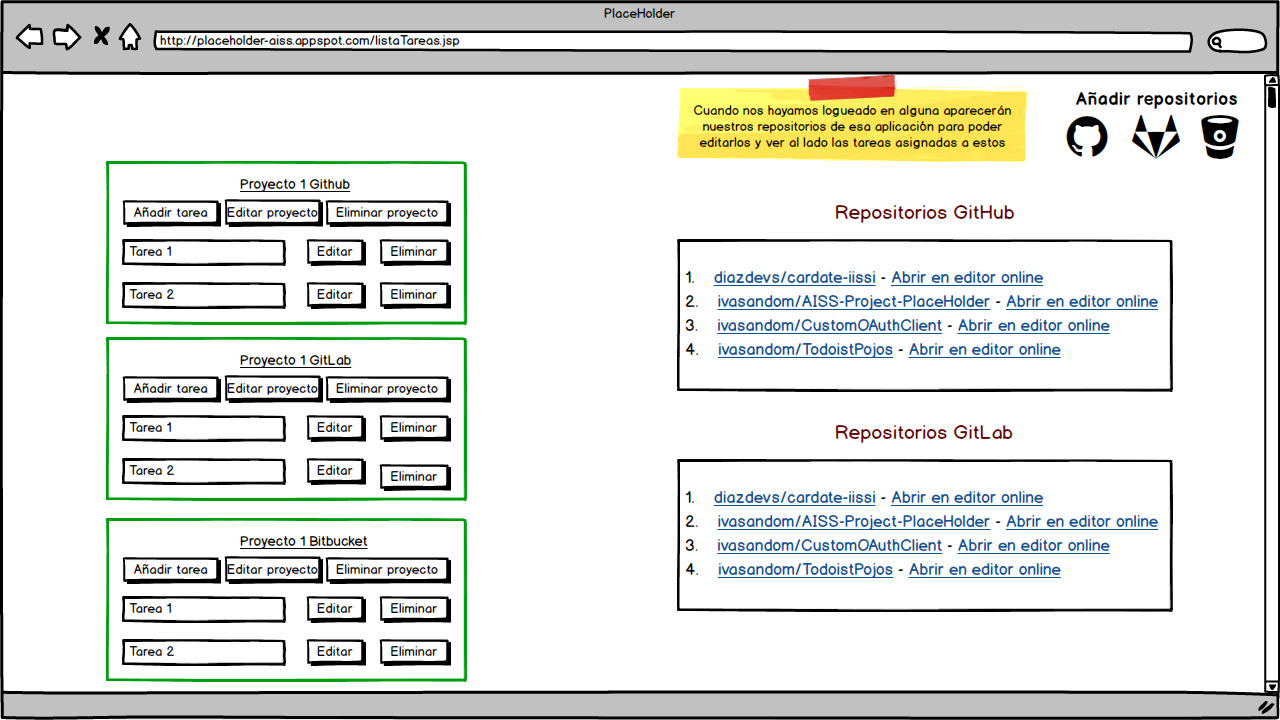


Figura 3. Prototipo de interfaz de usuario de la vista LISTATAREAS.JSP (CON REPOSITORIOS AÑADIDOS)

## Vista error.jsp

Página de error en el caso de que algo vaya mal en la web.

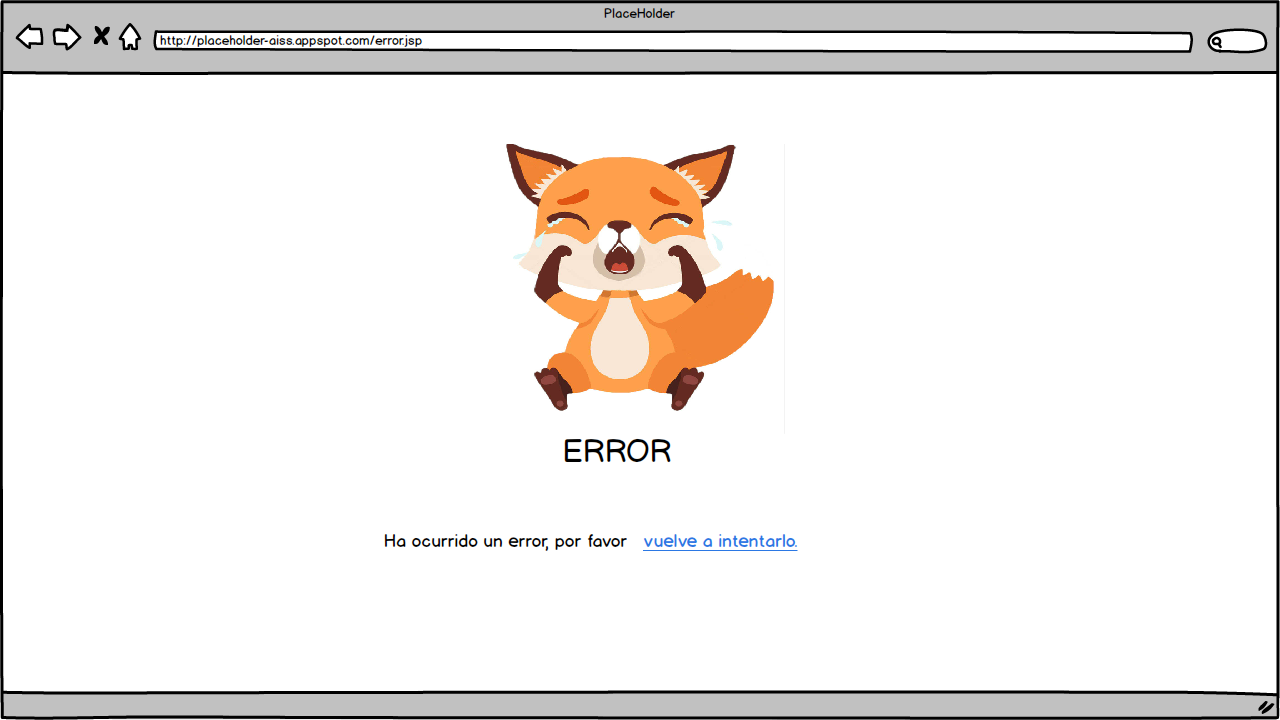
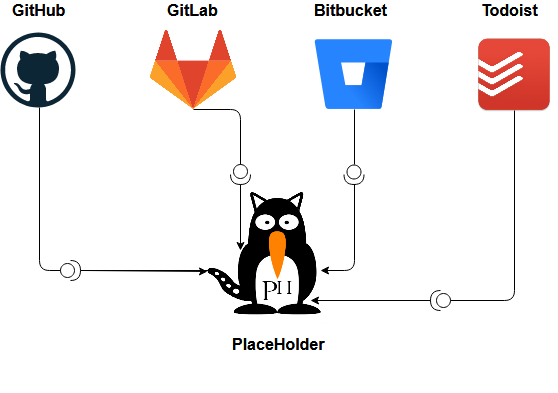


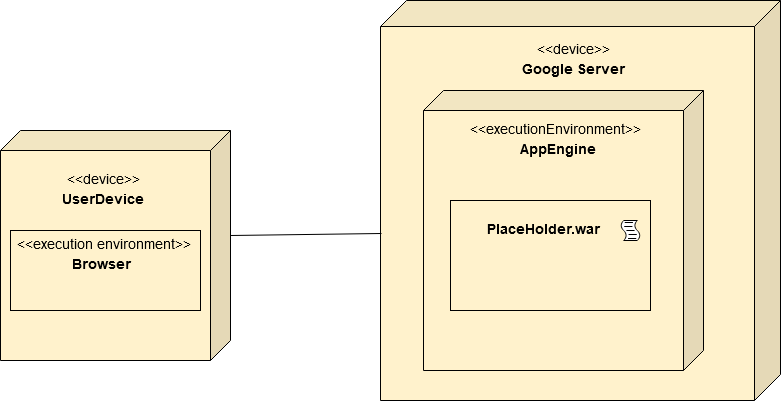
Figura 4. Prototipo de interfaz de usuario de la vista error.jsp

# Arquitectura

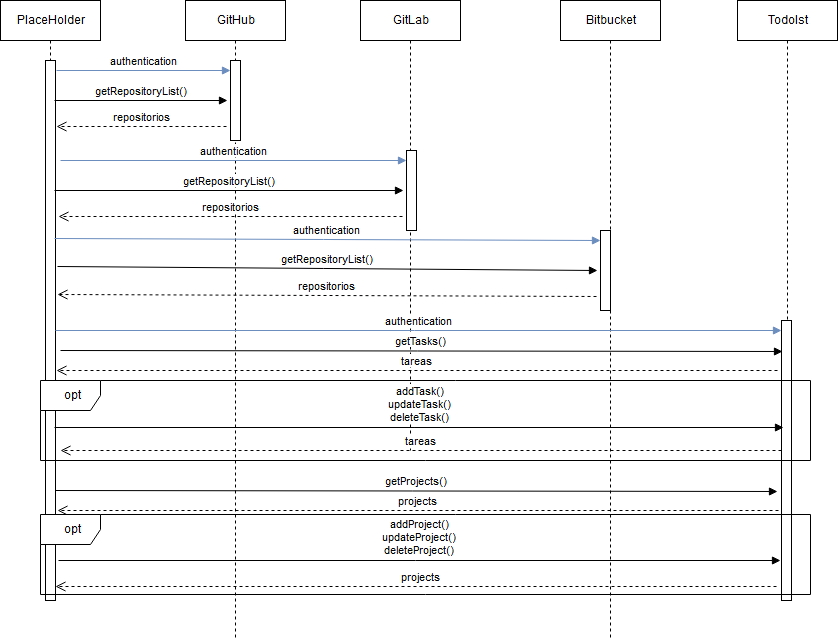
## Diagrama de componentes



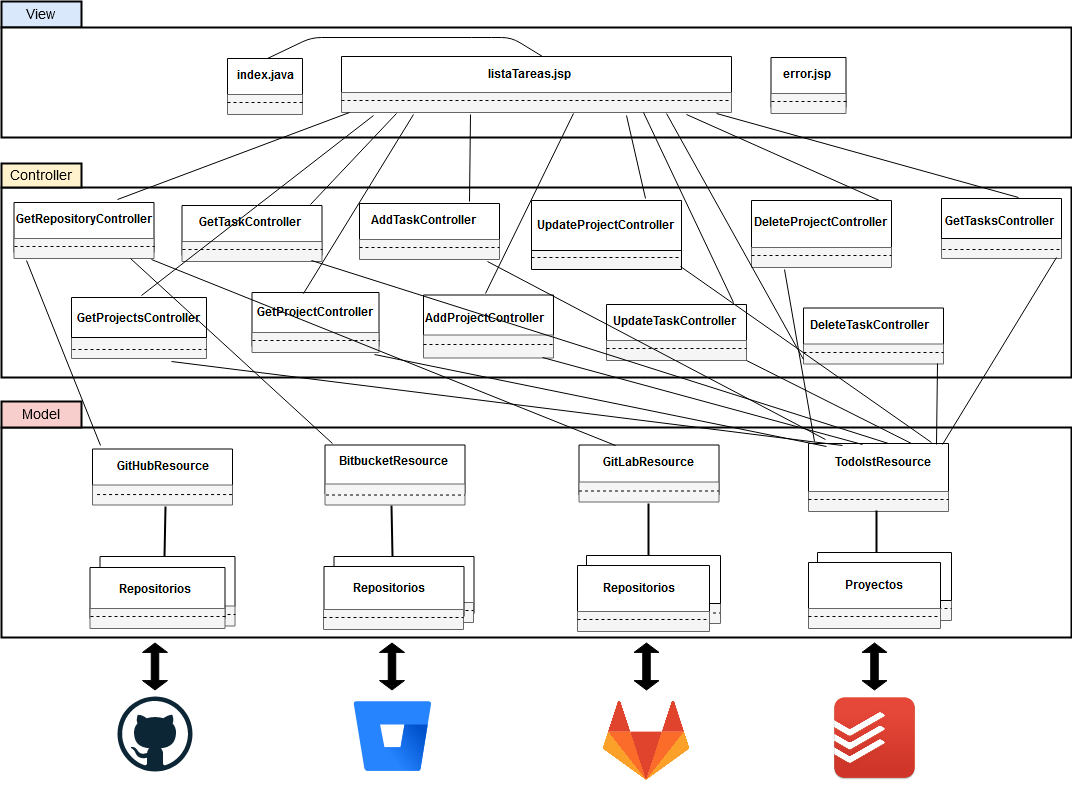
## Diagrama de despliegue



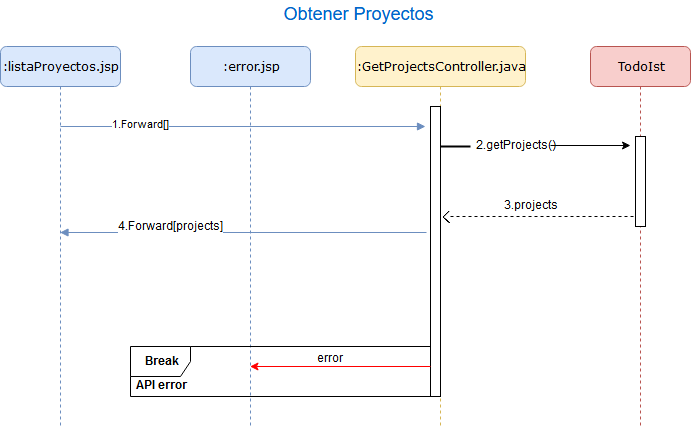
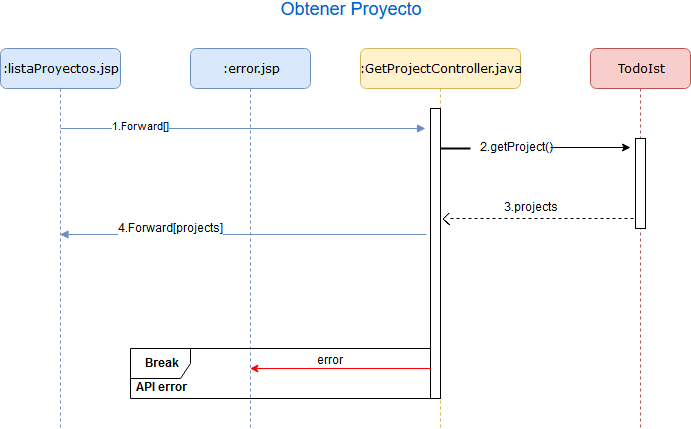
## Diagrama de secuencia de alto nivel

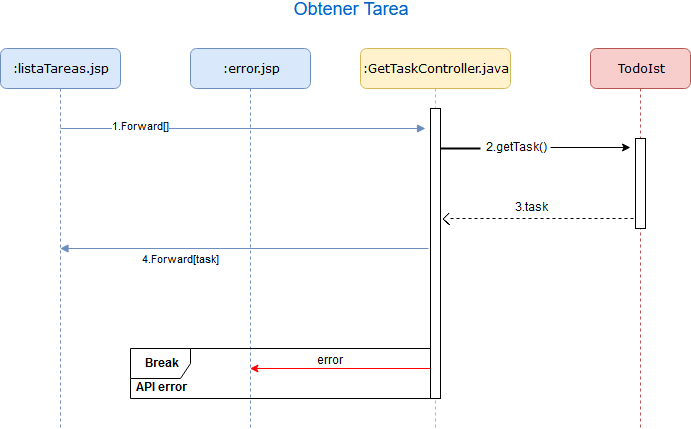
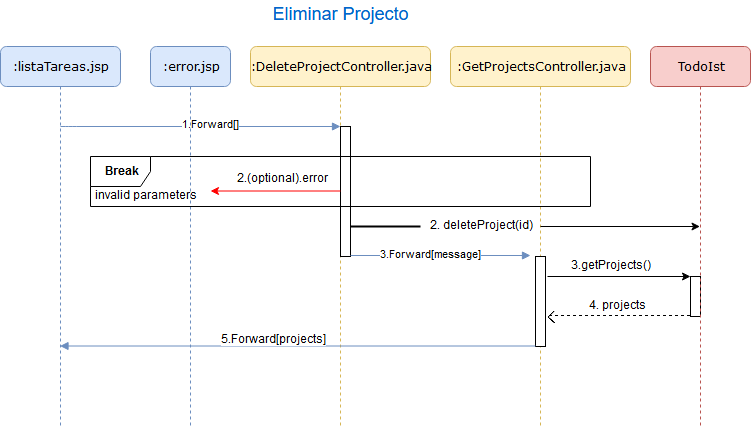
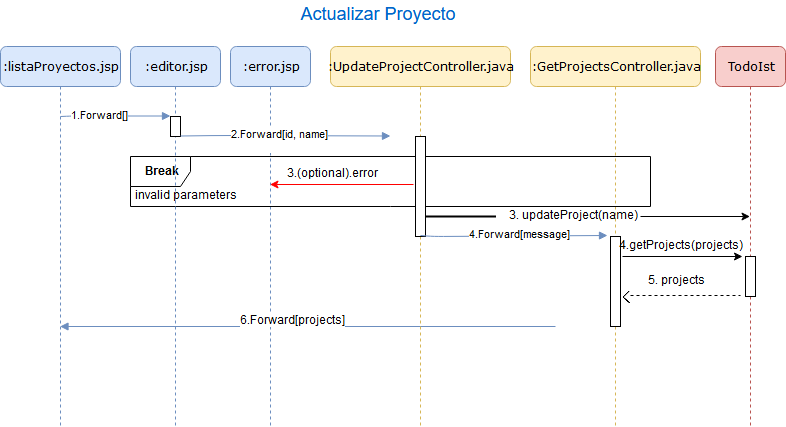
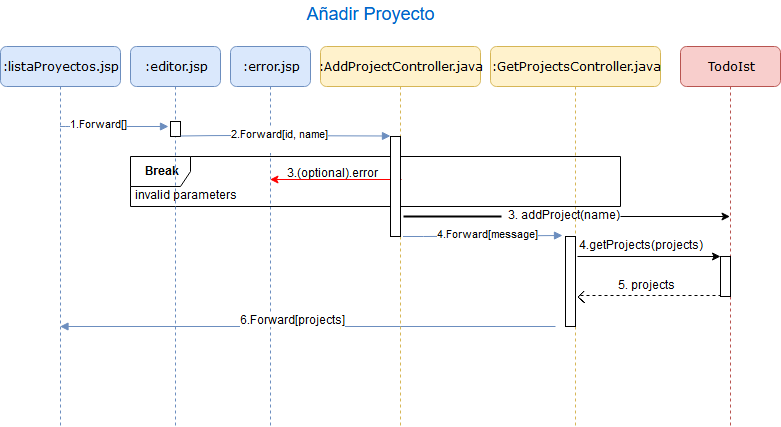


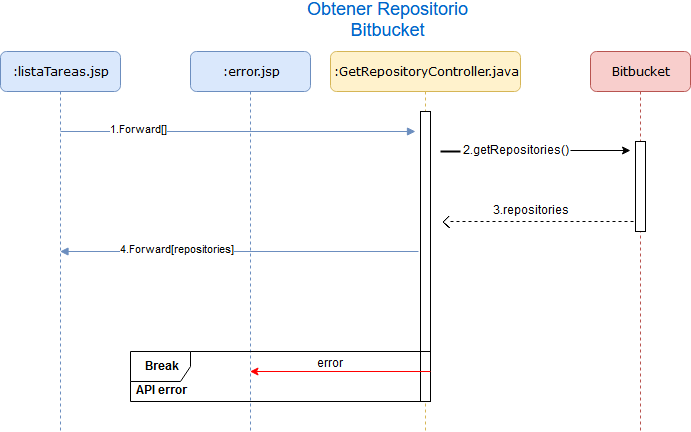
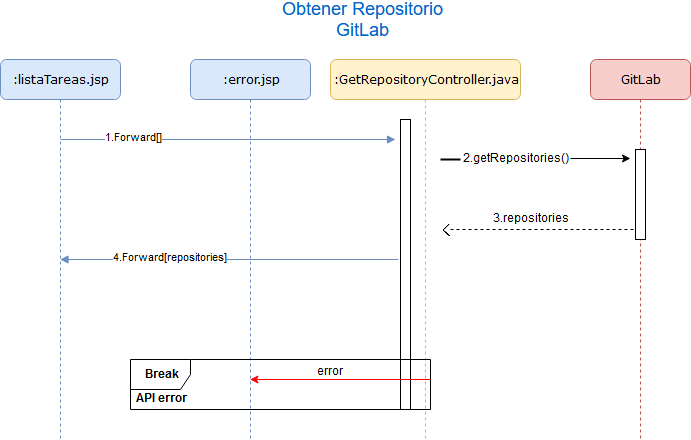
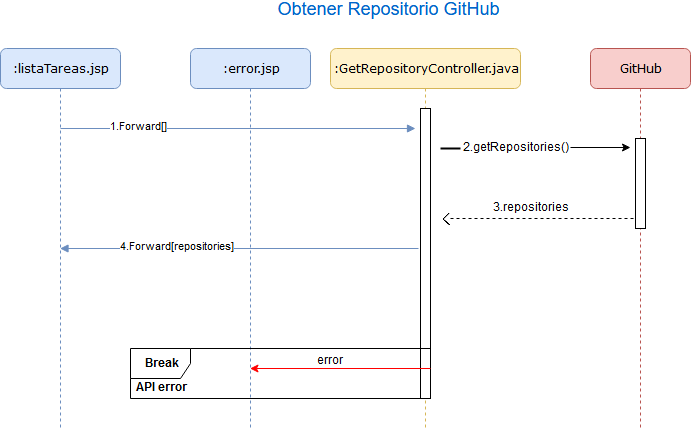
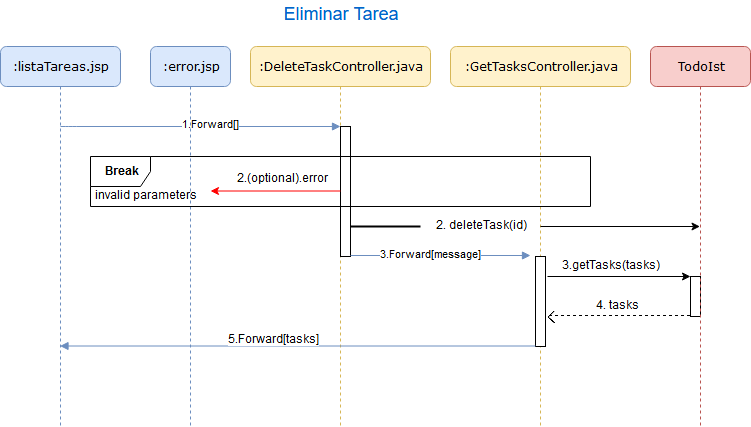
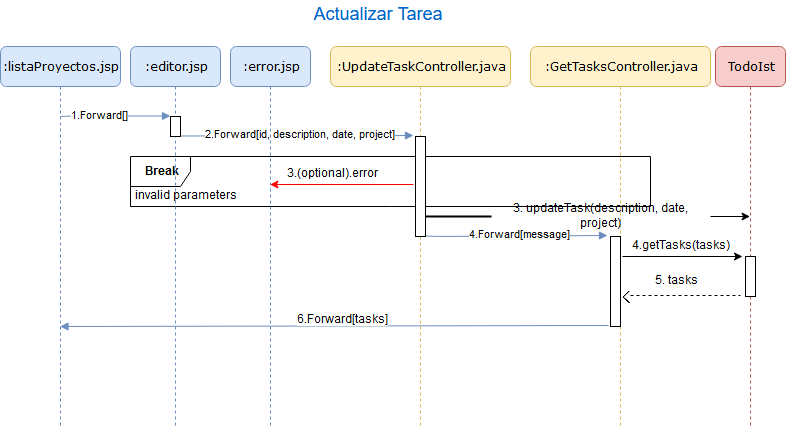
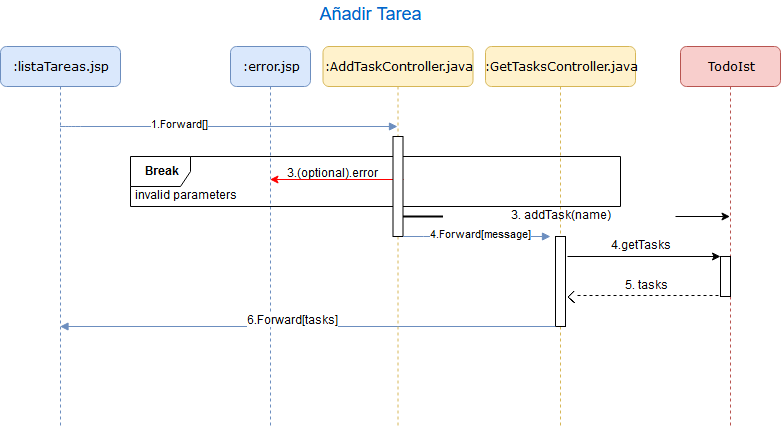
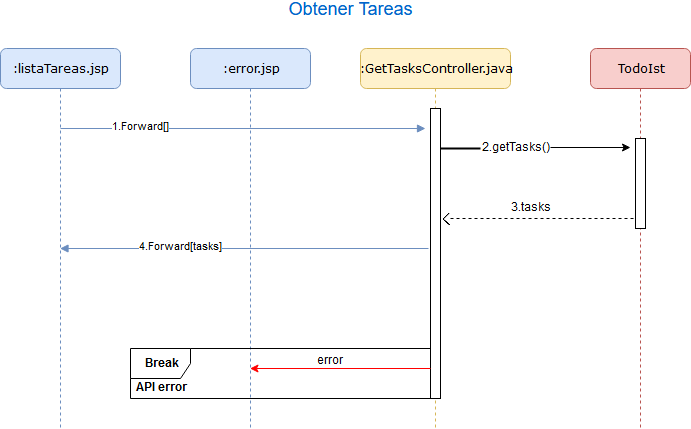
## Diagrama de clases



## Diagramas de secuencia







# Implementación

Describir brevemente los aspectos de la implementación que creen da más mérito al trabajo. Añadir algún fragmento de código si se considera oportuno.

//PONER LO DE GITHUB

Se ha implementado un editor de repositorios en la aplicación para que se pueda trabajar con ellos de una manera más cómoda, siendo más accesible para los usuarios poder ver los proyectos y tareas que tienen asociados a los repositorios y a la vez poder editarlos de una forma más sencilla y simple.

Se han añadido nuevas dependencias en el archivo pom.xml (línea 45) ya que daba un error en ClientResource.Path, en específico “Invalid HTTP method path”.

Se han añadido nuevos archivos en el paquete aiss.utility “Checkers.java” y “ProjectConfig.java” que sirven de ayuda en la gestión de código.

En cuanto a la interfaz, hemos utilizado el framework Bootstrap para que quede todo lo más intuitivo posible, y en específico hemos utilizado SweetAlert2 (<https://sweetalert2.github.io/>) para que salten pop-ups cuando se vayan a borrar o editar valores, ya que ofrece más dinamismo en la aplicación.

# Pruebas

Documentar las pruebas realizadas a la aplicación. Justificar textualmente la estrategia de pruebas seguida y por qué (ej. pruebas incrementales ascendentes).

Indicar el número total de pruebas realizadas y cuáles de ellas han sido automatizadas mediante JUnit.

|  |  |
| --- | --- |
| Resumen |  |
| Número total de pruebas realizadas | 25 |
| Número de pruebas automatizadas | 20 (80%) |

|  |  |
| --- | --- |
| ID | **Prueba 1** |
| Descripción | Prueba para la detección de errores al implementar búsquedas en Spotify usando servicios RESTful. |
| Entrada | Se hace uso de la librería XXX para invocar al servicio usando la URI YYY desde nuestra aplicación. |
| Salida esperada | Los datos devueltos en formato JSON son mapeados a una clase Java y a continuación se muestran por pantalla. |
| Resultado | **EXITO** |
| Automatizada | Sí |

# Manual de usuario

## Mashup

Indique textualmente e **incluyendo capturas de pantalla** el manual de uso del mashup.

## API REST

|  |  |  |
| --- | --- | --- |
| HTTP | URI | Descripción |
| GET | /projects | Devuelve todos los proyectos.  Si se reciben correctamente, devuelve un “200 sucessful operation”. Si no se reciben, devuelve “unexpected error”. |
| GET | /projects/{projectId} | Devuelve el proyecto con id={projectId}.  En caso de que no exista un proyecto guardado que coincida devuelve un “404 Not Found”.  Si el proyecto no tiene una id no válida se devuelve un error “400 Invalid ID supplied”.  Si se recibe correctamente, devuelve un “200 sucessful operation”. |
| PUT | /projects/{projectId} | Actualiza un proyecto con id={projectId}.  En caso de que no exista un proyecto guardado que coincida devuelve un “404 Not Found”.  Si el proyecto no tiene una id no válida se devuelve un error “400 Invalid ID supplied”.  Si se actualiza correctamente, devuelve un “204 No Content”. |
| POST | /projects | Se añade un proyecto.  Si el proyecto no es válido (null o vacío) se devuelve un error “400 Bad Request”.  Si se añade correctamente, se devuelve un “201 Created”. Si ya existe un proyecto con el mismo nombre devuelve un error “409 an existing project already exists”. |
| DELETE | /projects/{projectId} | Se elimina un proyecto con id={projectId}.  Si el id del proyecto no existe, devuelve un “404 Not Found’”.  Si el proyecto no tiene una id no válida se devuelve un error “400 Invalid ID supplied”.  Si se elimina correctamente, devuelve un “204 No Content”. |

Para toda la documentación de nuestra API, acceder aquí: <http://placeholder-aiss.appspot.com/docs/swagger.yaml>

{

"id": "p1",

"name": "Tareas de casa",

"tasks": [

{

"id": "t1",

"name": "Hacer la cena",

},

{

"id": "t2",

//...

}

]

}