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 (nicpazsar@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
24/05/2019	2.7	- Corregidos los diagramas y actualizados los mockups.	Nicolás Francisco Javier
25/05/2019	2.8	- Mejorada la API propia añadiendo filtrado y paginación y hechas y documentadas las pruebas del mashup.	Iván
26/05/2019	3.0	- Se elimina Bitbucket. Optimización final del editor. Actualizados diagramas. Correcciones de errores y revisión final de la aplicación.	José Francisco Javier Iván

Índice

1 Introducción

Placeholder es una plataforma de gestión de proyectos y tareas en la cual podrás enlazar tus repositorios y llevar un control exhaustivo del tiempo.

Podrás editar tus repositorios online sin necesidad de instalar ningún IDE ni extensión gracias al editor de Placeholder.

1.1 Aplicaciones integradas

Nombre aplicación	URL documentación API
GitHub	https://developer.github.com/v3/
GitLab	https://docs.gitlab.com/ee/api/
Todoist	https://developer.todoist.com/rest/v8/
Harvest	https://api.harvestapp.com/v2/

TABLA 1. APLICACIONES INTEGRADAS

1.2 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 enlazado con Todoist para asociar los proyectos de una aplicación y las tareas de la otra. También se ha añadido enlazar los repositorios cuando se está creando el proyecto para poder acceder al editor dentro de la vista del resumen de cada proyecto. El editor de repositorios implementado ya funciona correctamente.

En el último día hemos quitado Bitbucket ya que lo habíamos dejado de lado porque nos centramos en cosas más importantes ya que tenemos integradas dos aplicaciones de repositorios.

2 Prototipos de interfaz de usuario

2.1 Vista Index tras cargar repositorios

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. Como podemos ver hemos hecho login en una de las aplicaciones de repositorios, mostrándonos el token.

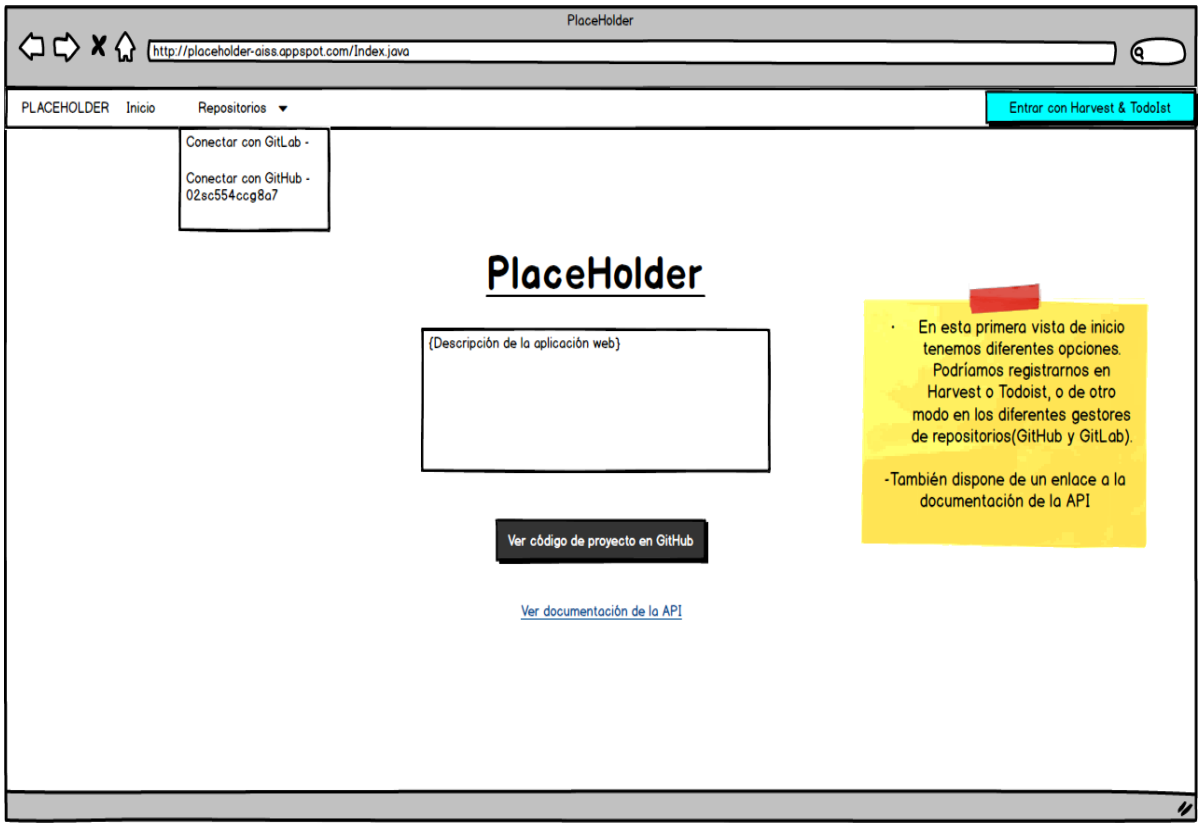


FIGURA 1. PROTOTIPO DE INTERFAZ DE USUARIO DE LA VISTA INDEX

2.2 Vista login

Tras haber hecho login en Harvest y Todoist nos redirigirá a la vista Index.

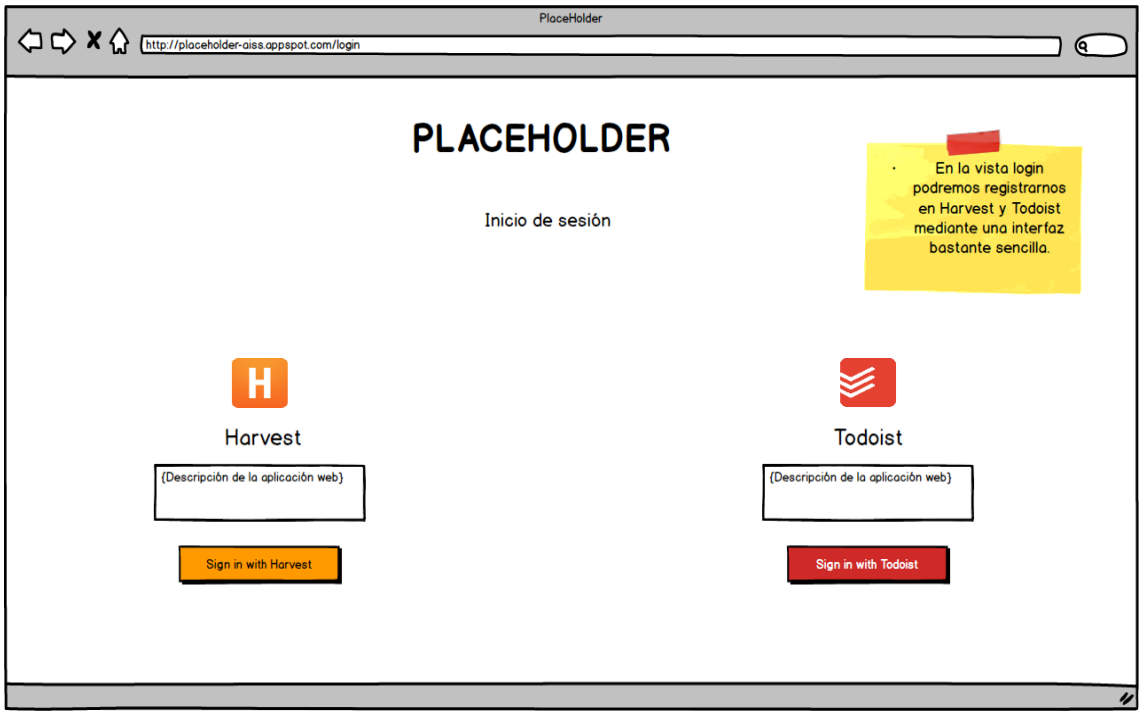


FIGURA 2. PROTOTIPO DE INTERFAZ DE USUARIO DE LA VISTA LOGIN

2.3 Vista Index tras cargar proyectos

En esta vista podemos acceder a los proyectos que tenemos o proceder a crear uno.

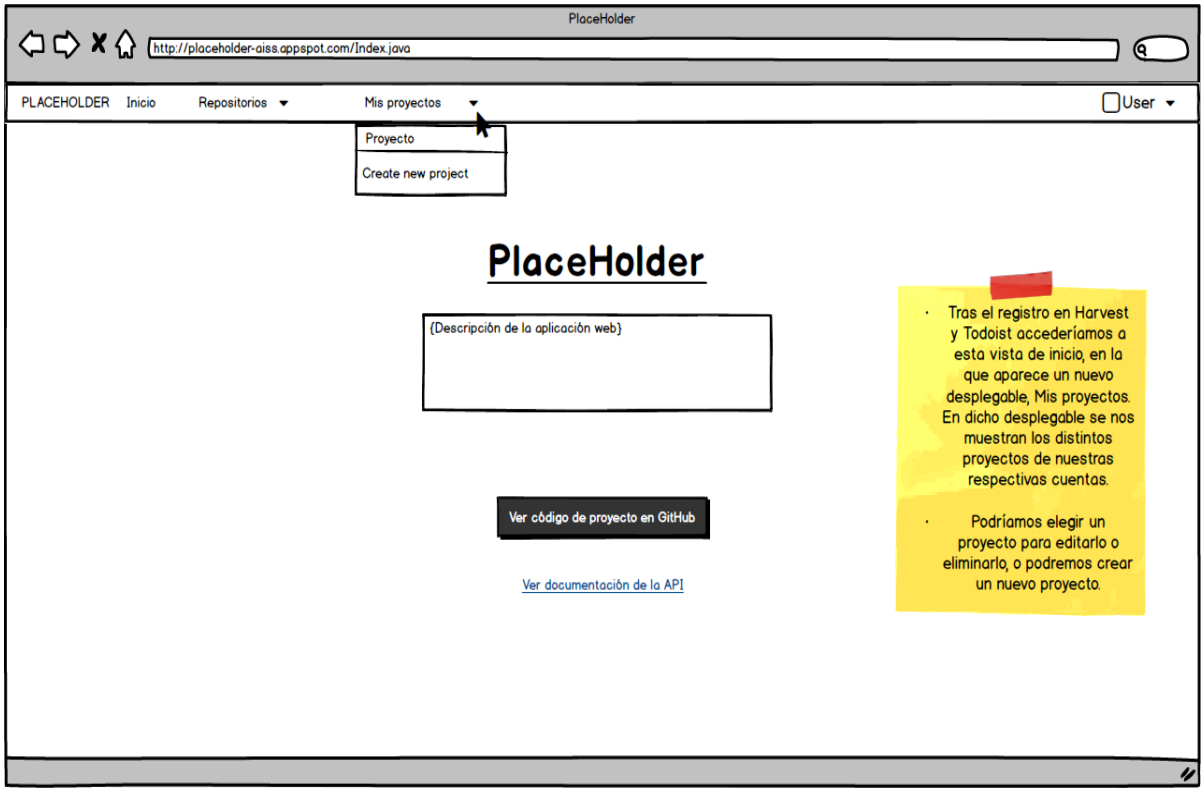


FIGURA 3. PROTOTIPO DE INTERFAZ DE USUARIO DE LA VISTA INDEX

2.4 Vista crear proyecto

Vista para crear el proyecto que luego aparecerá en la pestaña de “Mis proyectos”.

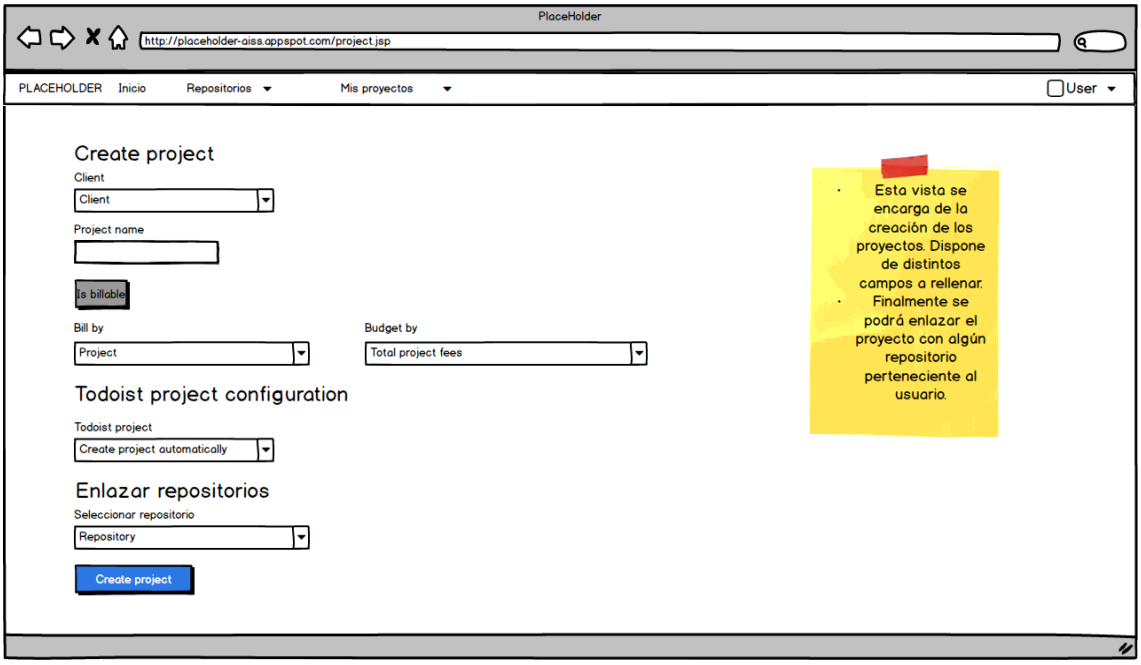


FIGURA 4. PROTOTIPO DE INTERFAZ DE USUARIO DE LA VISTA CREAR PROYECTO

2.5 Vista resumen proyecto

En esta vista podemos ver las distintas tareas de las que se compone el proyecto, también podemos añadir, eliminar y actualizar tareas.

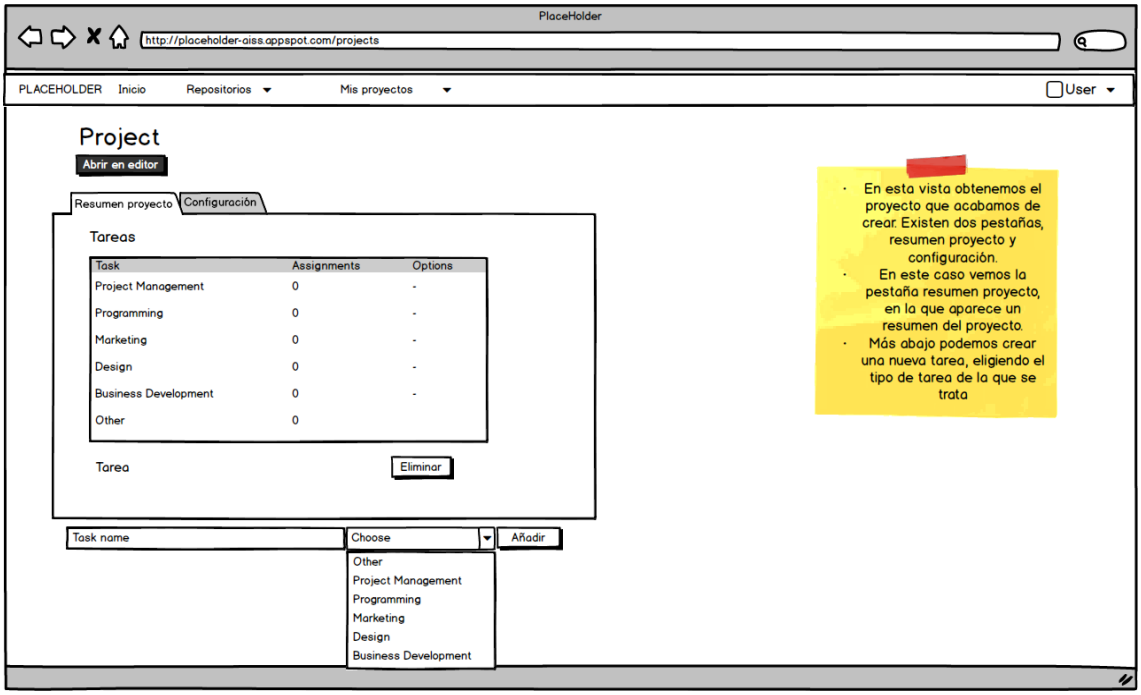


FIGURA 5. PROTOTIPO DE INTERFAZ DE USUARIO DE LA VISTA RESUMEN PROYECTO

2.6 Vista configuración proyecto

Aquí podemos ver los distintos repositorios que tenemos asociados al proyecto y también podemos eliminarlo y editarlo.

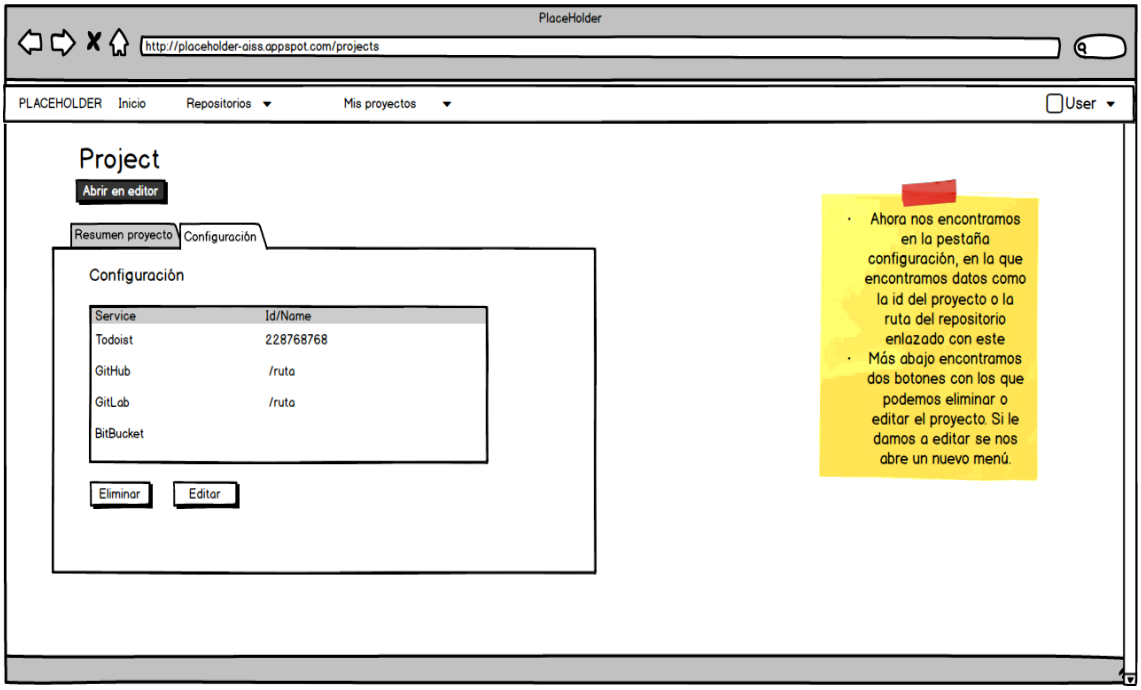


FIGURA 6. PROTOTIPO DE INTERFAZ DE USUARIO DE LA VISTA CONFIGURACIÓN PROYECTO

2.7 Vista editar proyecto

Aquí podemos editar algunas propiedades del proyecto.

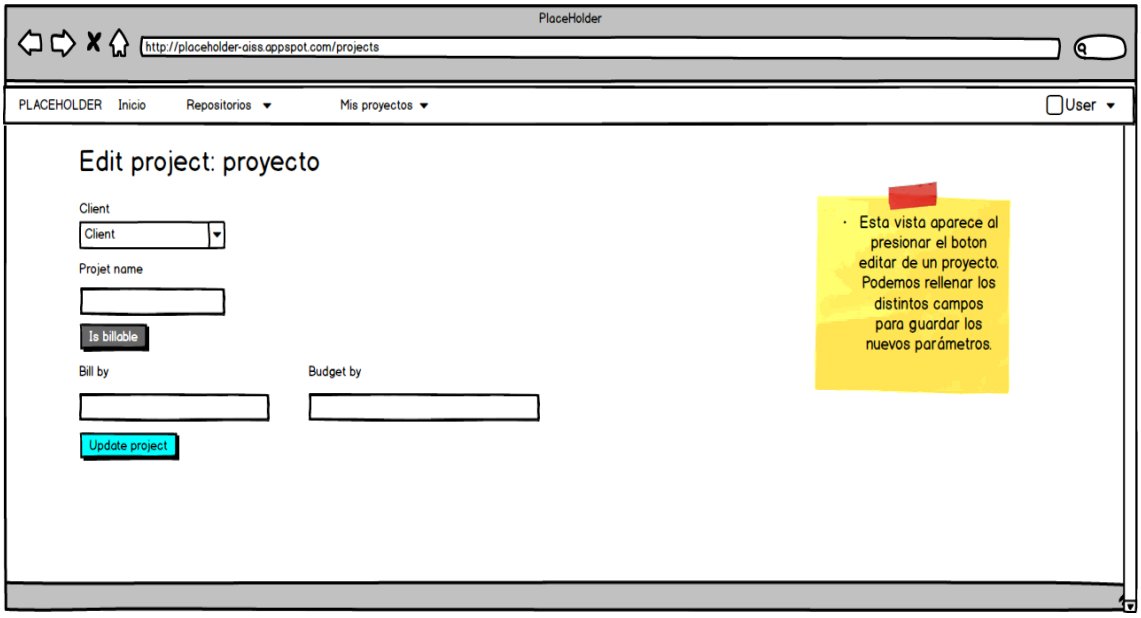


FIGURA 7. PROTOTIPO DE INTERFAZ DE USUARIO DE LA VISTA EDITAR PROYECTO

2.8 Vista editor repositorios

En esta vista podemos ver e incluso editar los repositorios que tenemos enlazados al proyecto.

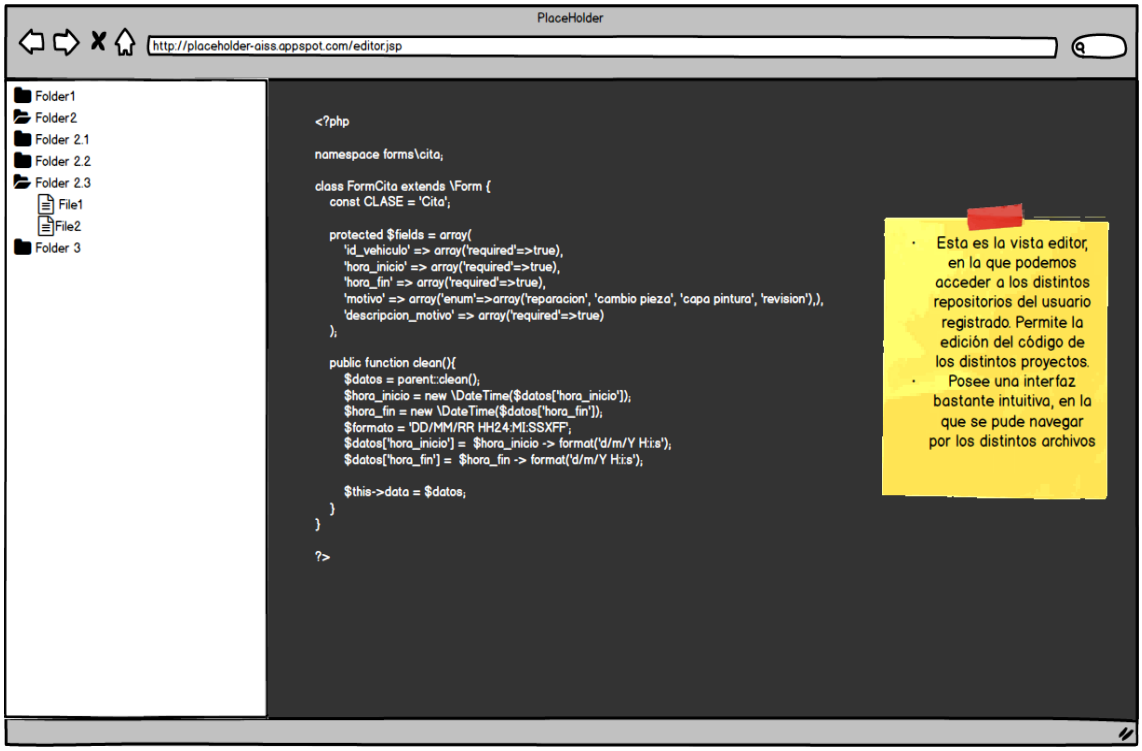


FIGURA 8. PROTOTIPO DE INTERFAZ DE USUARIO DE LA VISTA EDITOR REPOSITORIOS

2.9 Vista error

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

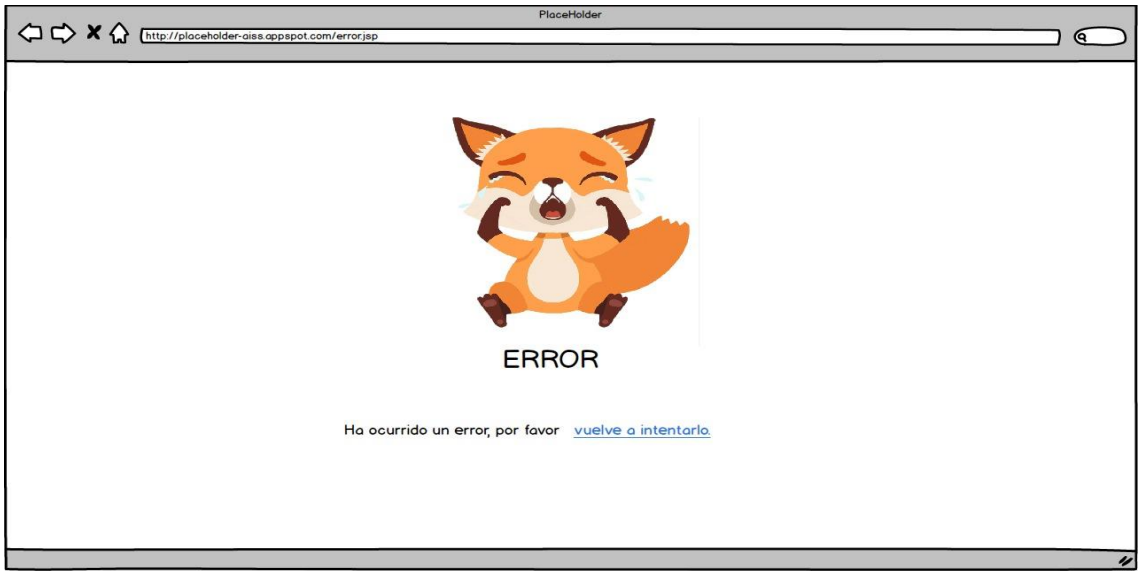
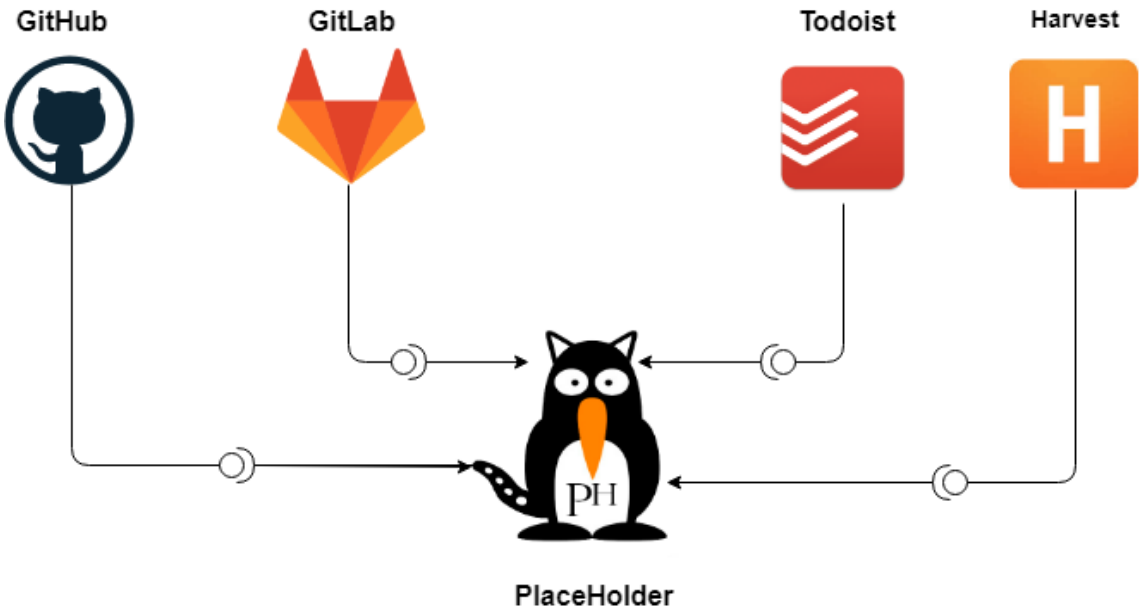


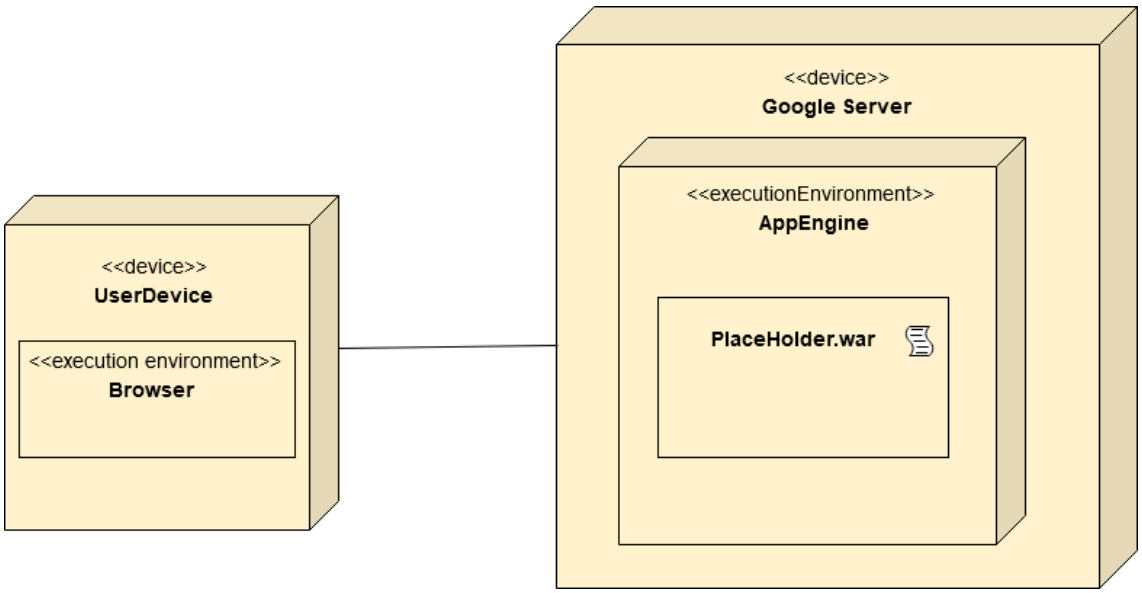
FIGURA 9. PROTOTIPO DE INTERFAZ DE USUARIO DE LA VISTA ERROR

3 Arquitectura

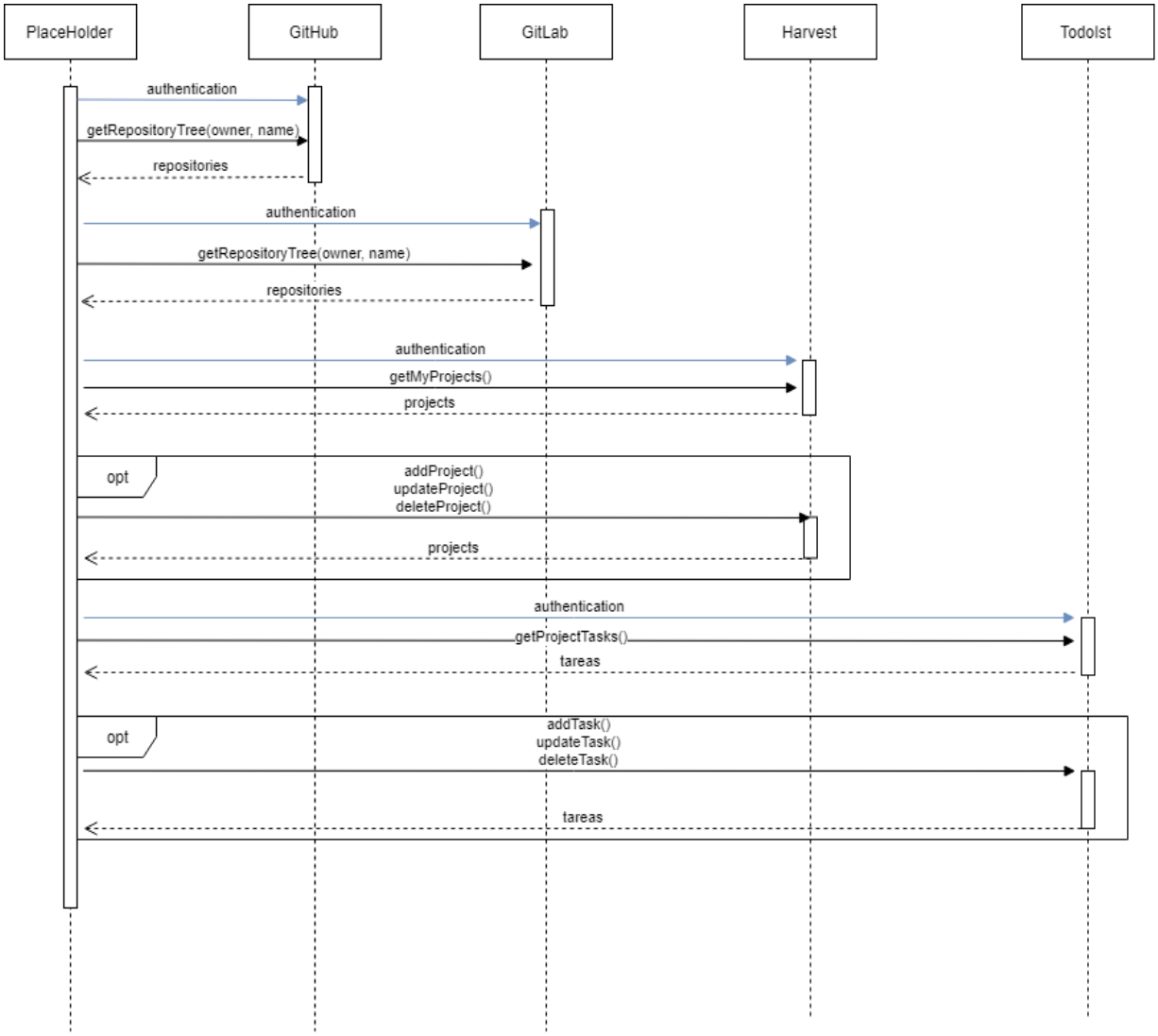
3.1 Diagrama de componentes



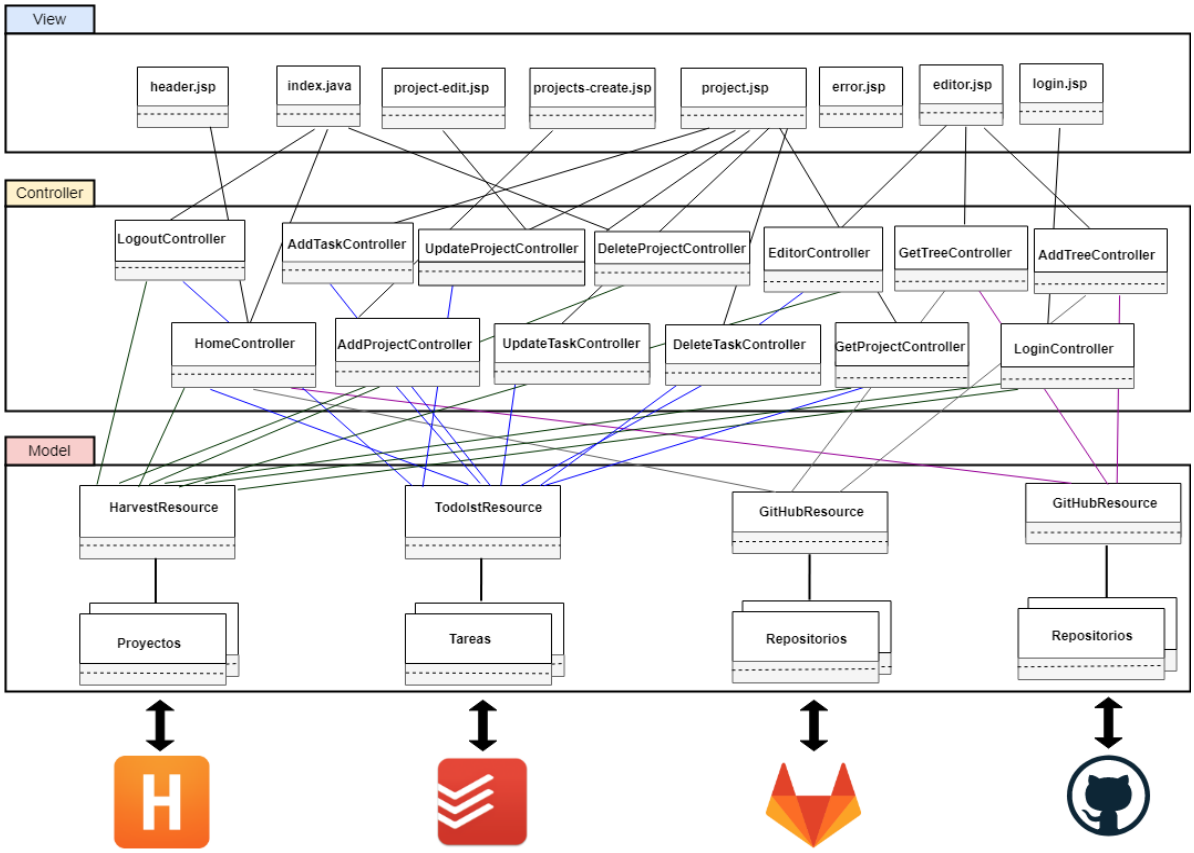
3.2 Diagrama de despliegue



3.3 Diagrama de secuencia de alto nivel

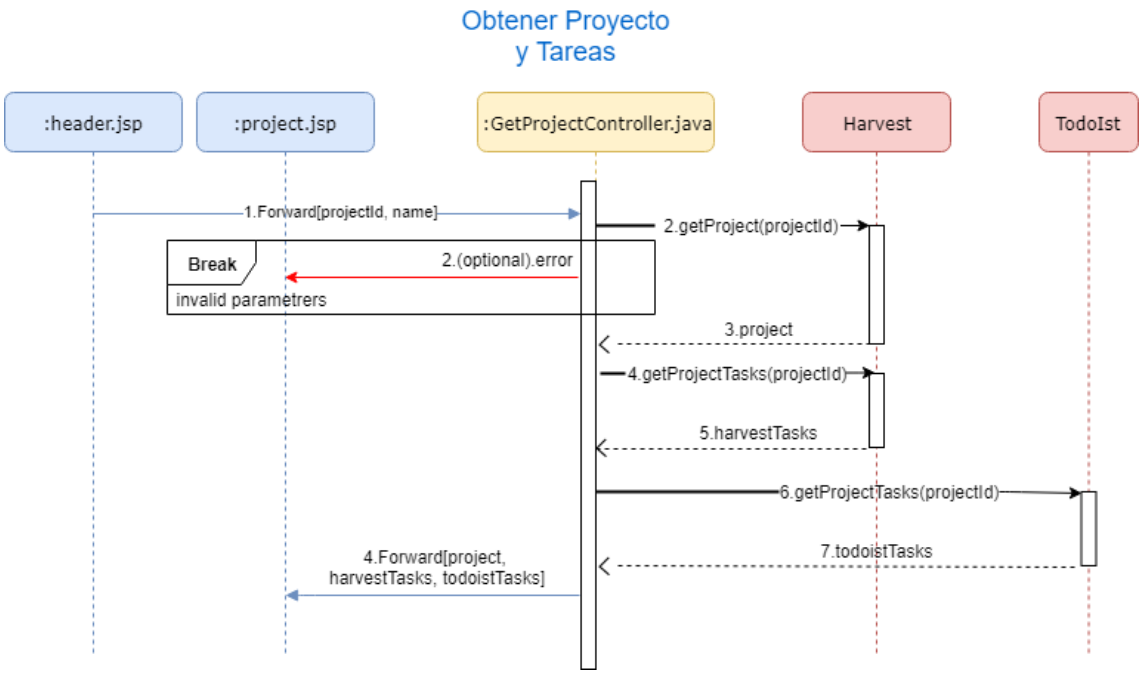


3.4 Diagrama de clases

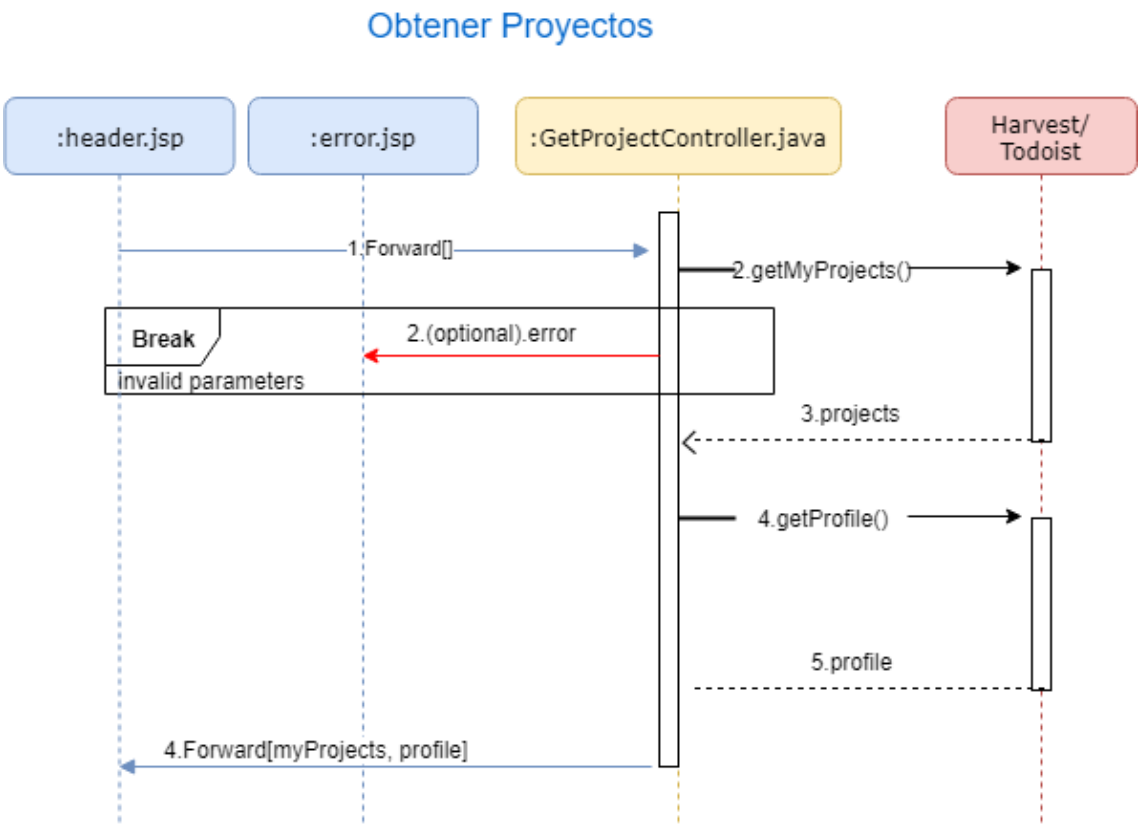


3.5 Diagramas de secuencia

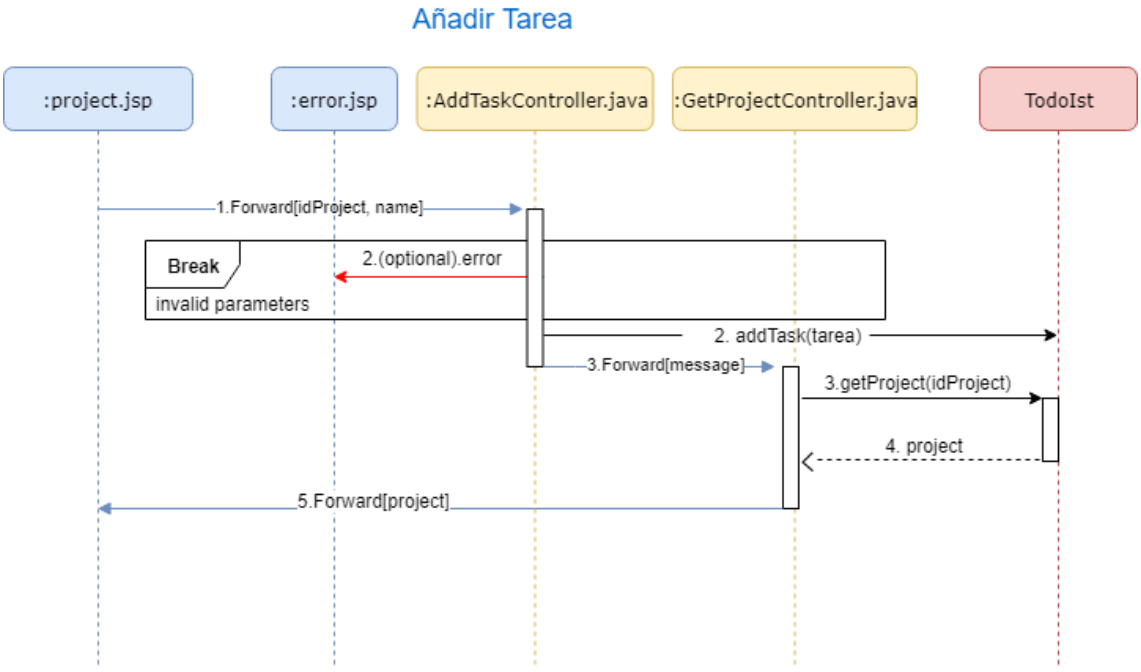
3.5.1 Obtener proyecto



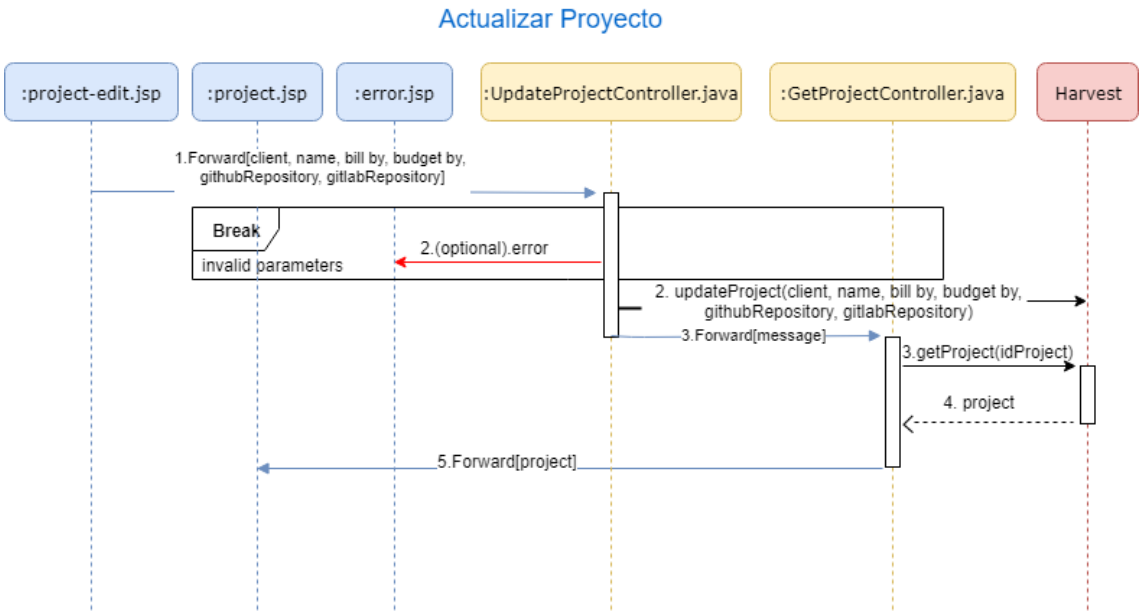
3.5.2 Obtener proyectos



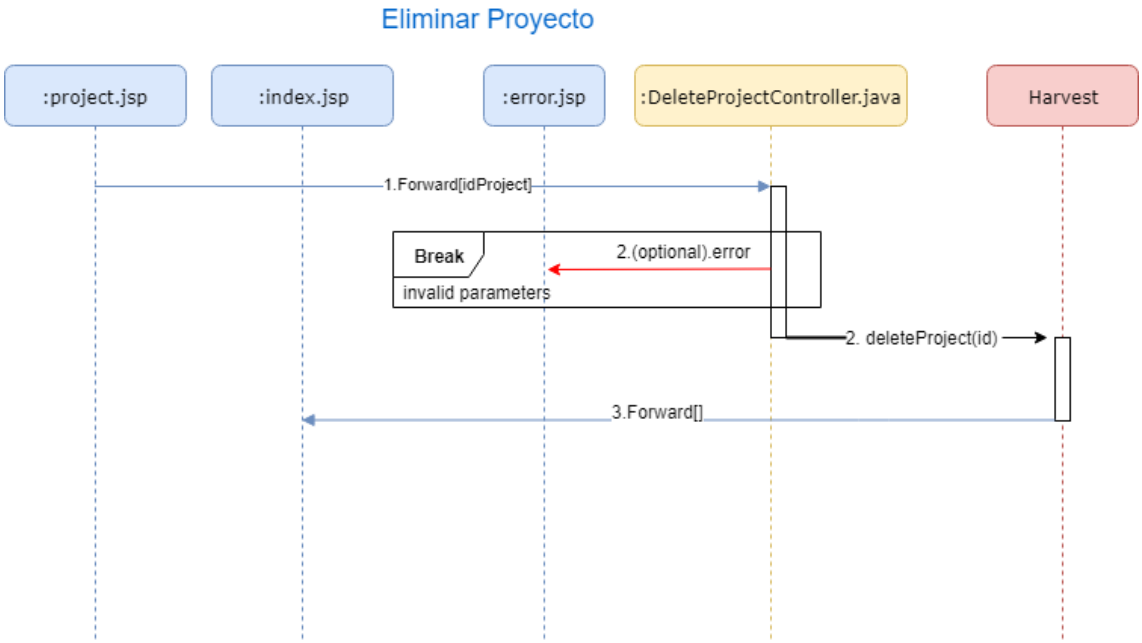
3.5.3 Añadir tarea



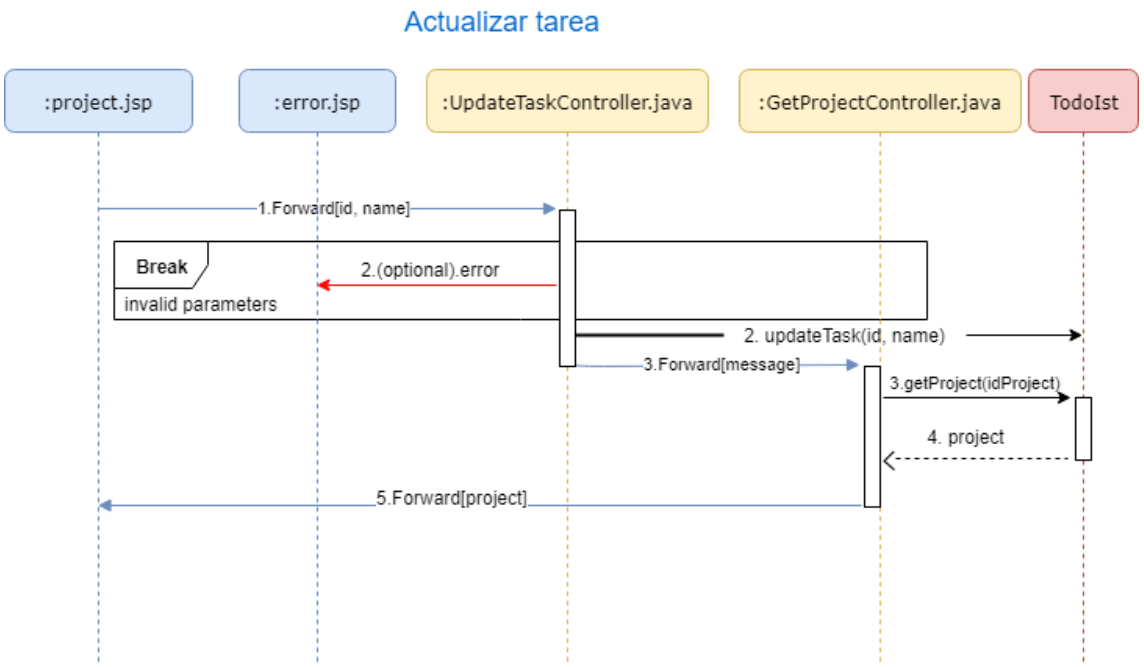
3.5.4 Actualizar proyecto



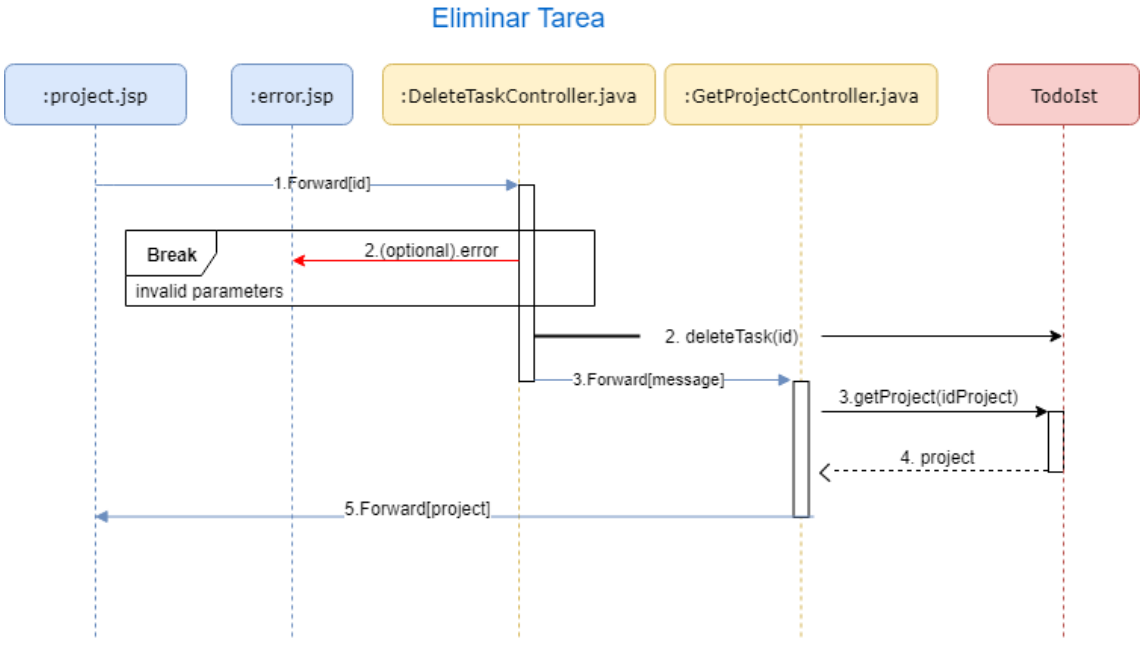
3.5.5 Eliminar proyecto



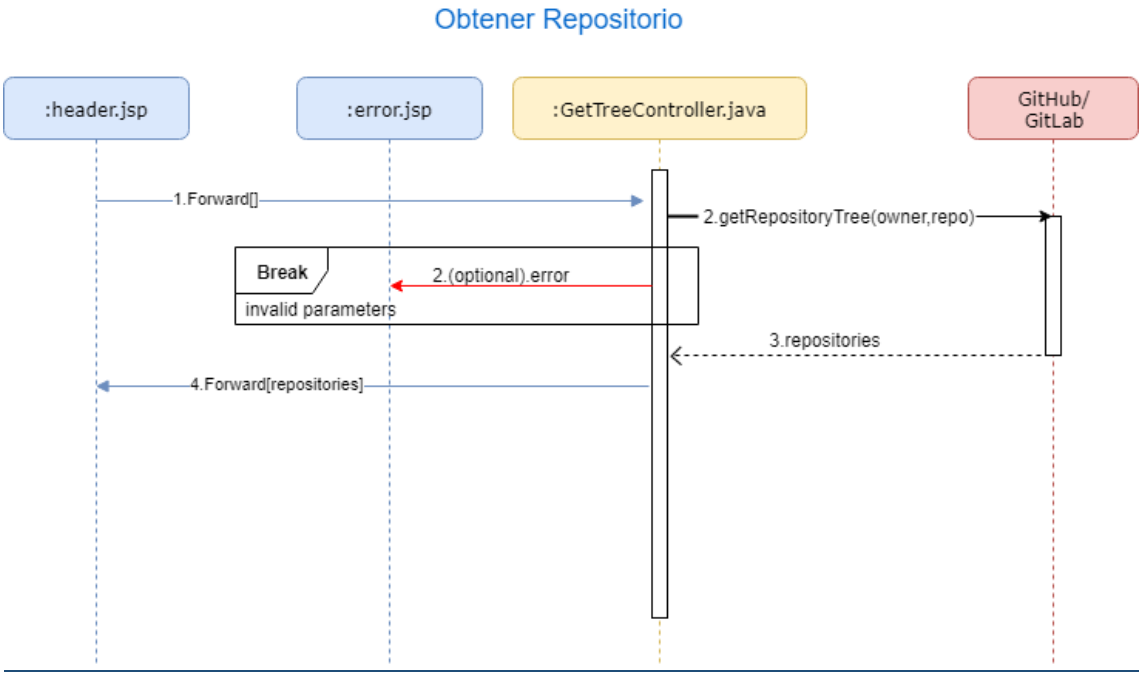
3.5.6 Actualizar tarea



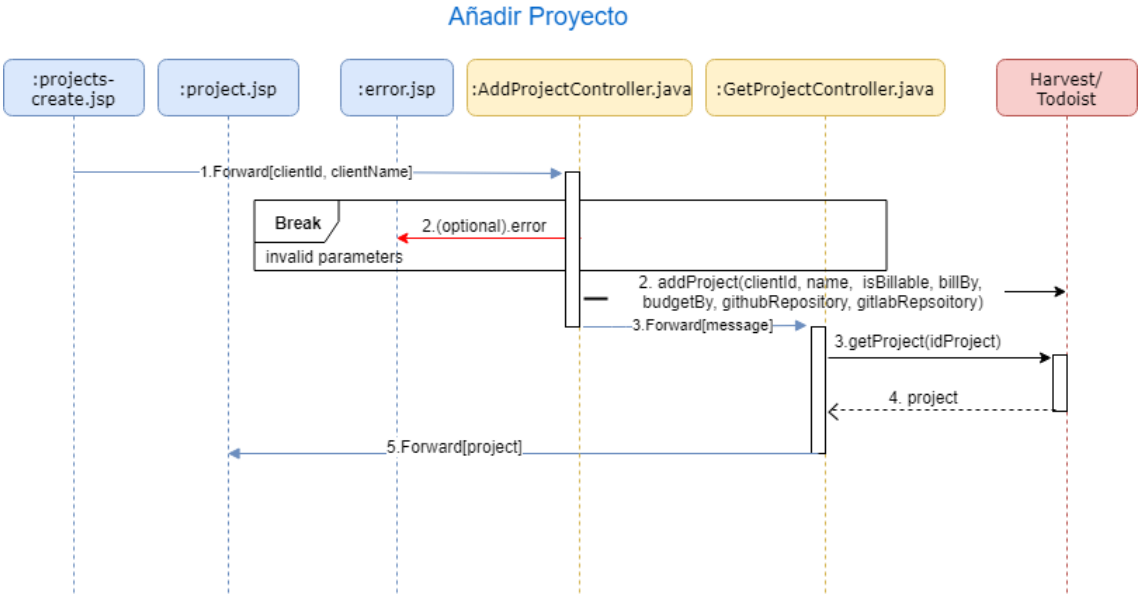
3.5.7 Eliminar tarea



3.5.8 Obtener repositorio



3.5.9 Añadir proyecto



4 Implementación

4.1.1 Editor de código

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.

Para ello se ha usado la librería de Microsoft https://microsoft.github.io/monaco-editor/ que usa Visual Studio Code.

4.1.2 Dependencias

Se han añadido nuevas dependencias en el archivo pom.xml (restlet.ext.httpclient línea 45) para solucionar el error al usar el método PATCH, en específico "Invalid HTTP method PATCH".

4.1.3 Clases de utilidad

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

Las clases ProjectConfig.java y TaskConfig.java guardan información extra como, por ejemplo, los repositorios relacionados. Estas clases se guardan en formato JSON en la propiedad notes y content respectivamente gracias a la clase JSONObject.

4.1.4 Interfaz

En cuanto a la interfaz, hemos utilizado el Framework Bootstrap4 (https://getbootstrap.com/) y la librería de javascriptSweetAlert2 (https://sweetalert2.github.io/) para que salten pop-ups cuando se vayan a borrar o editar valores, y así ofrecer más dinamismo a la aplicación.

También hemos utilizado jQuery (https://jquery.com/) para realizar peticiones AJAX y añadir elementos dinámicos de una forma más sencilla

4.1.5 OAuth2.0 GitHub

Al implementar la autenticación con GitHub, nos encontramos con que la plataforma devolvía el token con un Content-Type de texto plano (Content-Type: text/plain) causando el error en la autenticación.

Para solventar este problema, hemos añadido la cabecera Accept: application/json para que nos devuelva el token en formato json.

Aiss.controller.oauth.TokenRequestModificado.java (Línea 270

5 Pruebas

Hemos utilizado la estrategia de pruebas de integración de sándwich, combinando la integración ascendente y descendente por ramas. Hemos decidido probar primero los módulos atómicos, que en este caso son los resources, para detectar de primeras si existen fallos cuando se llama a las apis integradas ya que, de ser así, se pueden localizar fácilmente.

Seguidamente, hemos probado desde la vista index (entry point) hacia abajo por ramas, dependiendo de la vista a la que queremos acceder, para comprobar que se llama correctamente a los controladores y estos a los resources.

Resumen	
Número total de pruebas realizadas	10
Número de pruebas automatizadas	10 (100%)

ID	**Prueba 1**
Descripción	Prueba para la detección de errores al implementar búsqueda de repositorios en GitHub usando servicios RESTful.
Entrada	Se hace uso de la librería JUnit para invocar al servicio usando la URI "https://api.github.com/user/repos?access_token= 59b3940eef1d08a3916d11afbb48c8a4f48781b1" 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

ID	**Prueba 2**
Descripción	Prueba para la detección de errores al implementar búsqueda de repositorios en GitLab usando servicios RESTful.
Entrada	Se hace uso de la librería JUnit para invocar al servicio usando la URI "https://gitlab.com/api/v4/projects?access_token=98b9bde779f1a64 4341c612bcd8415605899d5faf7797f0753d1b1bb6119ae66" 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

ID	**Prueba 3**
Descripción	Prueba para la detección de errores al implementar búsqueda de proyectos en Harvest usando servicios RESTful.
Entrada	Se hace uso de la librería JUnit para invocar al servicio usando la URI "https://api.harvestapp.com/v2/projects?account_id=2796934" 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

ID	**Prueba 4**
Descripción	Prueba para la detección de errores al implementar eliminación en Harvest usando servicios RESTful.
Entrada	Se hace uso de la librería JUnit para invocar al servicio usando la URI "https://api.harvestapp.com/v2/projects/21216199 ?account_id==2796934" 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

ID	**Prueba 5**
Descripción	Prueba para la detección de errores al implementar actualización en Harvest usando servicios RESTful.
Entrada	Se hace uso de la librería JUnit para invocar al servicio usando la URI "https://api.harvestapp.com/v2/projects/21216212 ?account_id==2796934" 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

ID	**Prueba 6**
Descripción	Prueba para la detección de errores al implementar búsqueda de tareas en Harvest usando servicios RESTful.
Entrada	Se hace uso de la librería JUnit para invocar al servicio usando la URI "https://api.harvestapp.com/v2/projects/21216212/task_assignments ?account_id==2796934" 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

ID	**Prueba 7**
Descripción	Prueba para la detección de errores al implementar búsqueda de tareas en Todoist usando servicios RESTful.
Entrada	Se hace uso de la librería JUnit para invocar al servicio usando la URI "https://beta.todoist.com/API/v8/tasks?project_id=2210842978&token= 636a7278f087c14f6643a61517b38616cedba8f8" 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

ID	**Prueba 8**
Descripción	Prueba para la detección de errores al implementar eliminación de tareas en Todoist usando servicios RESTful.
Entrada	Se hace uso de la librería JUnit para invocar al servicio usando la URI "https://beta.todoist.com/API/v8/tasks/32145244218&token= 636a7278f087c14f6643a61517b38616cedba8f8" 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

ID	**Prueba 9**
Descripción	Prueba para la detección de errores al implementar actualización en Harvest usando interfaz de nuestra aplicación.
Entrada	Se hace uso de la extensión de mozilla Selenium IDE para grabar las acciones que hacemos manualmente para crear el proyecto y seguidamente reproducirlas paso a paso para comprobar que todo ha funcionado correctamente.
Salida esperada	Los datos devueltos en formato Selenium IDE's HTML se abren en una nueva ventana del plugin y se muestran los resultados en esa ventana. El test se ha subido a Google Drive para que se pueda ver y ejecutar: https://drive.google.com/file/d/1xx6sptt U9e9YrVmzh-cfC1tnovWdu8dO/view?usp=sharing
Resultado	**Éxito parcial.** Cuando se estaba ejecutando el test, al llegar a la parte en la que se le asigna un repositorio al nuevo proyecto, saltó un error, pero es debido a que la carga de archivos solo está soportada en la extensión de Google Chrome, no la de mozilla.
Automatizada	Sí

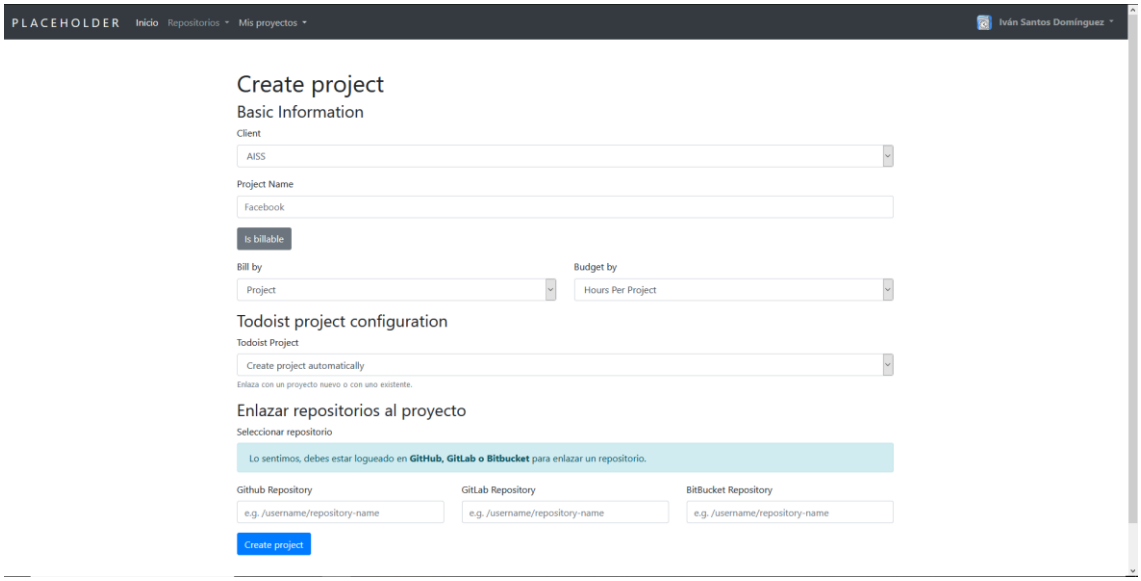
ID	**Prueba 10**
Descripción	Prueba para la detección de errores al implementar visualización de un repositorio de GitHub en el editor usando interfaz de nuestra aplicación.
Entrada	Se hace uso de la extensión de mozilla Selenium IDE para grabar las acciones que hacemos manualmente para abrir el repositorio asociado a un proyecto mediante el editor y visualizar su contenido y seguidamente reproducirlas paso a paso para comprobar que todo ha funcionado correctamente.
Salida esperada	Los datos devueltos en formato Selenium IDE's HTML se abren en una nueva ventana del plugin y se muestran los resultados en esa ventana. El test se ha subido a Google Drive para que se pueda ver y ejecutar: https://drive.google.com/file/d/1BGecjL_e2FdcBdqbTP8hym3eN7Kep-ec/view?usp=sharing
Resultado	**EXITO**
Automatizada	Sí

6 Manual de usuario

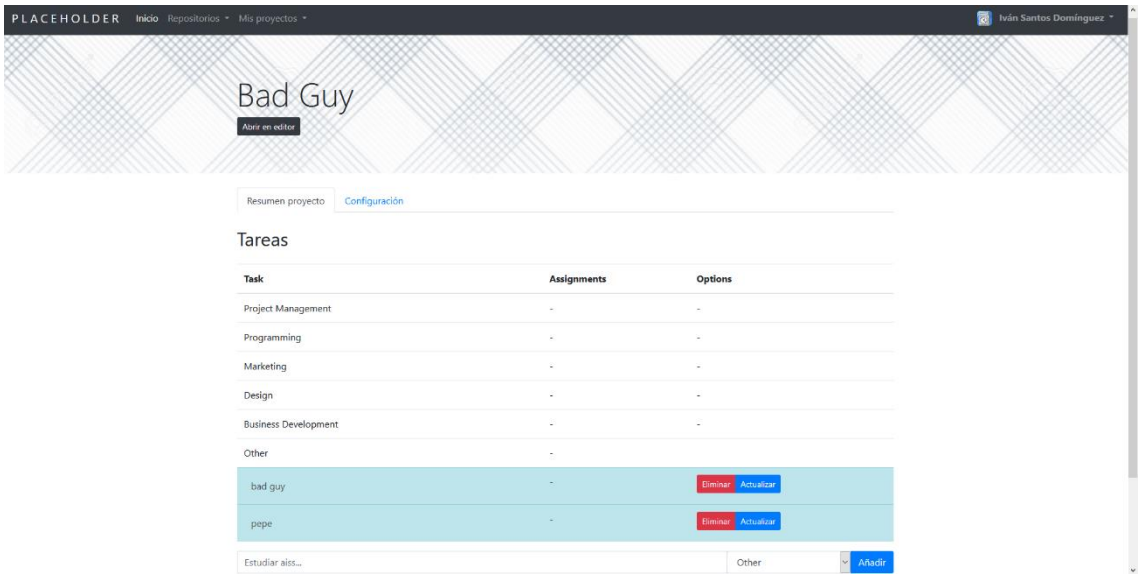
6.1 Mashup

Al entrar por primera vez en la página el usuario tendrá dos opciones para hacer primero; o hacer click en “Entrar con Harvest & Todoist” para cargar los proyectos y tareas y que salgan en una nueva pestaña del menú, o hacer click en la pestaña “Repositorios” para cargar los repositorios de las distintas aplicaciones.

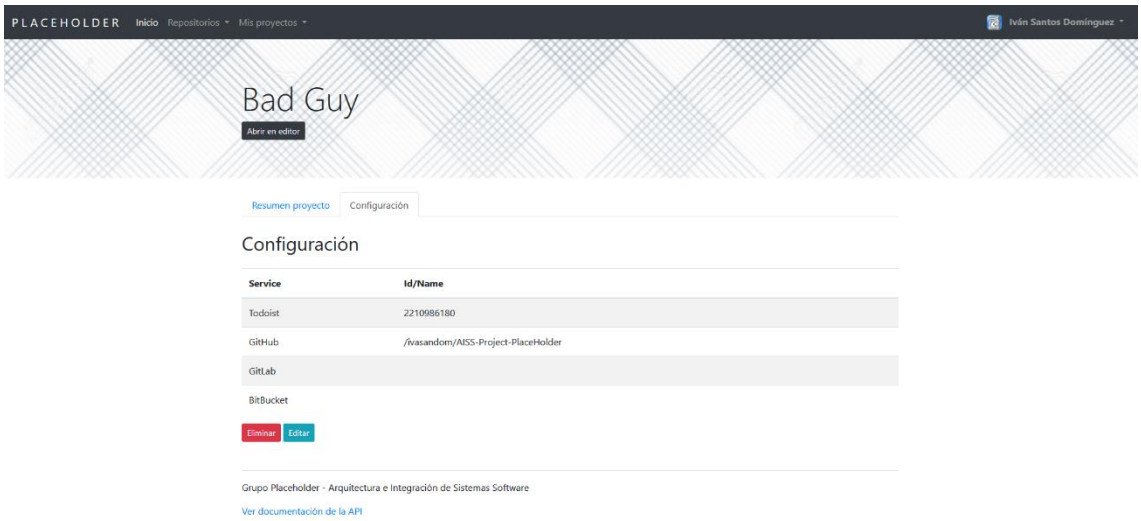
Si queremos crear un nuevo proyecto, en la pestaña “Mis proyectos” seleccionamos “create new project”, y nos llevará a la vista para crearlo. Elegimos las opciones de creación tal y como nos guían e incluso podemos enlazar repositorios al proyecto.



Una vez creado podemos acceder a él y nos encontraremos con dos pestañas. En “Resumen proyecto” podemos ver, añadir y eliminar tareas, y en el botón “Abrir en editor” podemos acceder al editor de repositorios y ver y editar los repositorios asignados al proyecto.

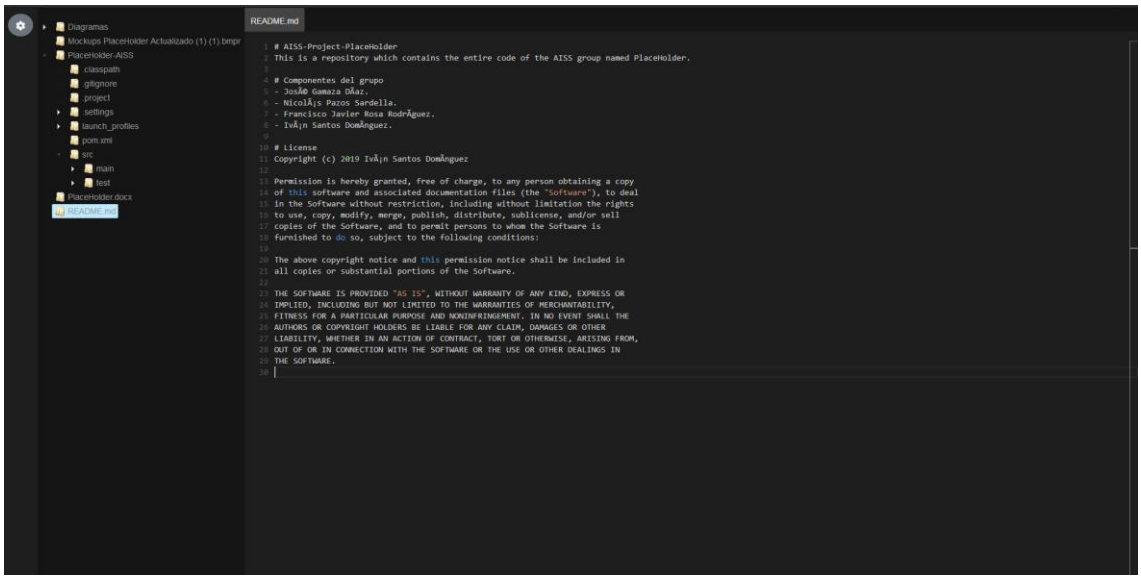


En la pestaña “Configuración” podemos actualizar la información del proyecto o incluso eliminarlo, y ver las direcciones de los repositorios asociados al proyecto.



Y, por último, al abrir el editor, seleccionamos uno de los repositorios que tenemos enlazados al proyecto, aunque si no hemos hecho log anteriormente en la aplicación de repositorios nos saltará un error explicando eso.

Dentro del editor podemos navegar entre los archivos del repositorio e incluso editarlos, y cuando hayamos terminado podemos hacer commit para guardar los cambios del repositorio.



6.2 API REST

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

Recurso task

HTTP	URI	Descripción
GET	/tasks	Devuelve todas las tareas. Si se reciben correctamente, devuelve un “200 sucessful operation”. Si no se reciben, devuelve “unexpected error”. Opcionalmente se le puede añadir a la URI ?name={letra} para filtrar todas las tareas que empiecen por esa letra. Si no hay ninguna que coincida no devuelve ninguna tarea. Opcionalmente se le puede añadir a la URI ?start={comienzo de página}&size={tamaño de página} para que muestre las tareas a partir de la posición start y el número de tareas mostradas a partir de esa posición está definido por size.
GET	/tasks/{taskId}	Devuelve la tarea con id={taskId}. En caso de que no exista una tarea guardada que coincida devuelve un “404 Not Found”. Si la tarea no tiene una id no válida se devuelve un error “400 Invalid ID supplied”. Si se recibe correctamente, devuelve un “200 sucessful operation”.
PUT	/tasks/{taskId}	Actualiza una tarea con id={taskId}. En caso de que no exista una tarea guardada que coincida devuelve un “404 Not Found”. Si la tarea no tiene una id no válida se devuelve un error “400 Invalid ID supplied”. Si se actualiza correctamente, devuelve un “204 No Content”.
POST	/tasks	Se añade una tarea. Si la tarea no es válida (null o vacío) se devuelve un error “400 Bad Request”. Si se añade correctamente, se devuelve un “201 Created”. Si ya existe una tarea con el mismo nombre devuelve un error “409 an existing task already exists”.
DELETE	/tasks/{taskId}	Se elimina una tarea con id={taskId}. Si el id de la tareano existe, devuelve un “404 Not Found’”. Si la tarea no tiene una id no válida se

		devuelve un error “400 Invalid ID supplied”. Si se elimina correctamente, devuelve un “204 No Content”.

Recurso Project

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”. Opcionalmente se le puede añadir a la URI ?name={letra} para filtrar todos los proyectos que empiecen por esa letra. Si no hay ninguno que coincida no devuelve ningún proyecto. Opcionalmente se le puede añadir a la URI ?start={comienzo de página}&size={tamaño de página} para que muestre los proyectos a partir de la posición start y el número de proyectos mostrados a partir de esa posición está definido por size.
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”.
POST	/projects/{projectId}/{taskId}	Añade la tarea con id={taskId} al proyecto con id={projectId}. Si el proyecto o la tarea no existe, devuelve un “404 Not Found”. Si la tarea ya está incluida en el proyecto devuelve un “400 Bad Request”. Si se añade satisfactoriamente, devuelve “201 Created” con la referencia a la URI y el contenido del proyecto.
DELETE	/projects/{projectId}/{taskId}	Elimina la tarea con id={taskId} del proyecto con id={projectId}. Si el proyecto o la tarea no existe, devuelve un “404 Not Found”. Si se realiza correctamente, devuelve “204 No Content”.

Todas las pruebas de la API se han realizado satisfactoriamente con la extensión de mozilla RESTClient.

```
{
        "id": "p1",
        "name": "Tareas de casa",
        "tasks": [
        {
                "id": "t1",
                "name": "Hacer la cena",
        },
        {
                "id": "t2",
                //...
        }
        ]
}
```