Universidad Tecnológica de Bolívar

Facultad de Ingeniería

Dir. programa de Ingeniería de Sistemas

Informe de prácticas investigativas

Especificación de requisitos de software para la Interfaz de Programación de Aplicaciones del laboratorio de computación de alto desempeño de la Universidad Tecnológica de Bolívar

Director:

Jairo Enrique Serrano Castañeda

Estudiante

Diego Germán Navarro Tesillo

Código estudiante: T00015910

Enero 20 de 2017

Cartagena de Indias D. T Y C

Tabla de contenidos

Introducción

Una de la característica de la mayoría de la herramienta para el procesamiento de grandes cantidades de información en paralelo es que la interfaz de usuario es en texto plano o en línea de comando, y para poder hacer uso de las herramientas, es necesario un conocimiento medio de sistemas operativos unix, y buen manejo de línea de comandos[1]. Para mejorar esto se propone el uso de una API que sirva de mediador entre HTCondor[2] y el usuario final.

Propósito

El siguiente documento va dirigido a la comunidad de estudiantes y profesores que desean usar y futuro extender los actuales componentes y requisitos usados para el diseño y construcción de la interfaz de programación de aplicaciones, abreviado (API) del Laboratorio de Computación de Alto Desempeño de la Universidad Tecnologica de Bolivar.

El diseño y construcción de esta API está centrada en cubrir la necesidad de los potenciales y actuales clientes del HPCLab, exponiendo las herramientas usadas para el procesamiento de información en el laboratorio.

Alcance

Mediante la implementación de esta nueva API se busca acceder al sistema actual, basado en HTCondor, y poder simplificar el acceso a las herramientas existentes, con el fin de disminuir la curva de aprendizaje en el uso de los recursos que provee el Laboratorio de Computación de Alto Desempeño. Con la implementación de esta API se busca diversificar los tipos de clientes que pueden acceder al sistema, tales Aplicaciones móviles, interfaces web, etc.

Para la construcción de esta API se usó la arquitectura REpresentational State Transfer o REST (Transferencia de Estado Representacional), el cual usa el estándar HTTP para la transferencia de datos. Además permite que las acciones sean categorizadas de forma más natural y uniforme y al mismo tiempo permite la separación de los recursos y su representación.

Resumen

Este documento describe el diseño de un middleware entre un sistema de computación de alto rendimiento como HTCondor y un cliente final que dispone de todas las herramientas disponibles por el sistema.

En el capítulo 1 se describe la perspectiva del producto, sus funcionalidades, las características de los usuarios, las restricciones de la API y su evolución predecible. De la misma forma en el capítulo 2 se describe los requisitos funcionales y no funcionales del sistema

Capítulo 1

Para acceder a los recursos se ha planteado usar “Tareas”, las cuales representan un conjunto de datos de entrada, con capacidad de ser procesados por las herramientas disponibles, generando así una salida la cual puede ser usada para posterior análisis.

Las tareas pueden ser procesadas de forma individual o usando un *flujos de trabajo* (workflow[3]) los cuales permiten concatenar los resultados de tareas anteriores como datos de entrada de tareas subsiguientes; estos flujos de trabajo se pueden representar usando Grafos Acíclicos Dirigidos (DAG) o Grafos no Acíclicos Dirigidos (non-DAG). Las tareas DAG se pueden dividir en *Tareas de secuencia*, *Tareas de paralelismo* y *Tareas de elección.*

El conjunto de las posibles tareas a realizar, están organizadas y catalogadas como “Proyectos”, los cuales contienen la información básica sobre el grupo de tareas a ejecutar.

Perspectiva

Esta API hace parte del sistema usado en el laboratorio de computación de alto desempeño de la Universidad Tecnológica de Bolívar y es un middleware usado una puerta de acceso a los recursos de hardware y a las herramientas de software disponibles en el laboratorio por de medio del estándar HTTP. De igual forma que ofrece una forma de organizar el trabajo a realizar por medio de proyectos y herramientas asociadas cada proyecto, como se ha descrito al inicio de este capítulo

Funcionalidad

Por medio de esta API un usuario de la comunidad estudiantil o docente de la Universidad Tecnológica de Bolívar puede, crear, editar, visualizar y eliminar proyectos personales y al mismo tiempo que puede crear, editar, visualizar, ejecutar y asociar a un proyecto tareas.

Características de los usuarios

Los usuarios de esta API podrán autenticarse usando sus códigos y contraseña de estudiante de la plataforma SIRIUS, obteniendo así un token el cual permitirá el uso seguro de esta API. de las misma forma los usuarios estarán categorizados como logrando así jerarquizar las acciones de la API en roles.

Evolución previsible del sistema

Estado actual del sistema

Es indispensable pensar en una arquitectura que permita hacer todo esto de la forma más sencilla y ágil posible. A este punto esta API está construida con las bases de una

aplicación monolítica. Esto se traduce en que un gran conjunto de código es el encargado de suplir los requerimientos funcionales del sistema, así el gran conjunto de código es el encargado de comunicarse con HTCondor, subir archivos, administrar proyectos, etc.

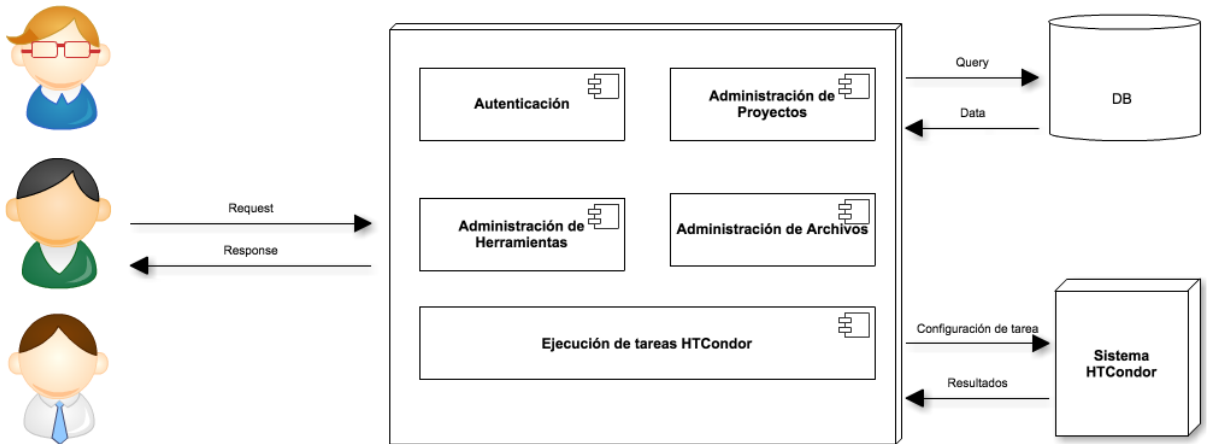


Figura 1. Arquitectura actual del sistema

Los problema con esta arquitectura van desde alto riesgo de dependencia entre los componentes, alto riesgo de comprometer la escalabilidad, debido a que la información viaja de forma horizontal yendo desde la Base de datos, pasando por una o varias partes del sistema antes de llegar al cliente final; se puede llegar a ver comprometido el rendimiento del sistema por un error en algún componente que no esté bien aislado. A nivel de hardware vemos que todos los componentes de la API comparten los mismo recursos en una misma máquina, podríamos seguir nombrando los riesgos de usar una arquitectura monolítica al momento de expandir el sistema, sin embargo hay arquitecturas alternativas que permiten explorar y barajar mejores formas de distribuir y escalar los servicios de un sistema; La “Arquitectura de Microservicios”.

Arquitectura de Microservicios

El término "Arquitectura de Microservicios" ha surgido en los últimos años para describir una forma particular de diseñar aplicaciones de software como pequeños servicios desplegables de forma independiente. Si bien no existe una definición precisa de este estilo arquitectónico, existen ciertas características comunes alrededor del despliegue automatizado, la inteligencia en los puntos finales y el control descentralizado de lenguajes y datos.

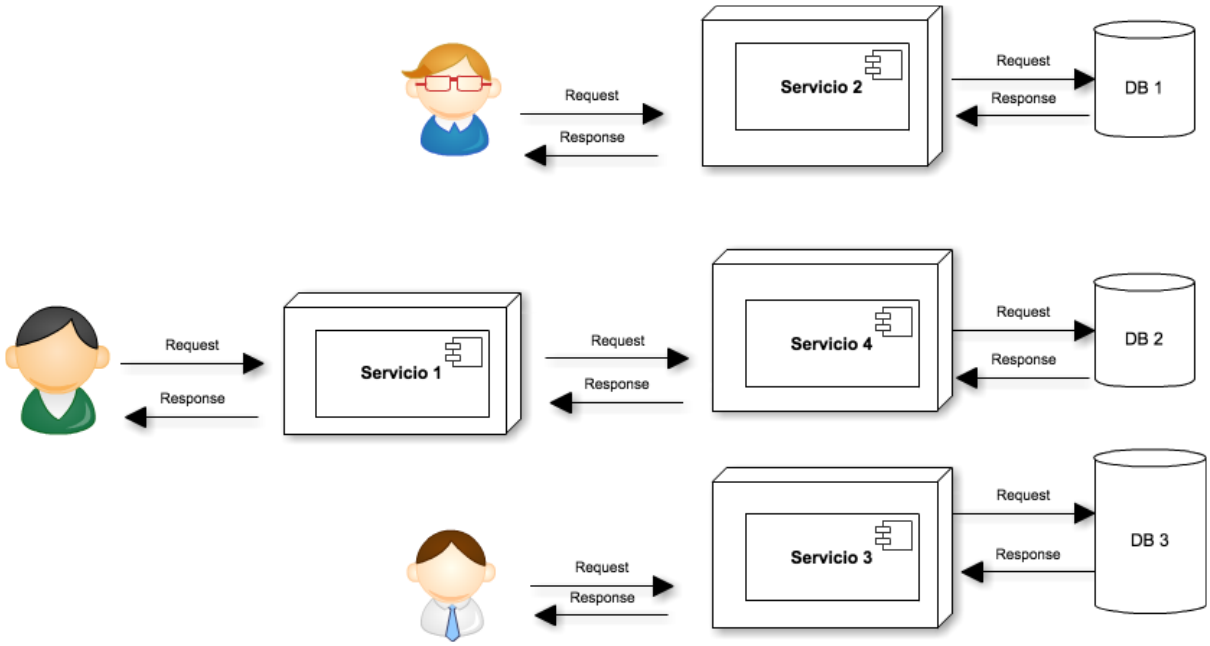


Figura 2. Representación de arquitectura de Microservicios

En un sistema basado en la "Arquitectura de Microservicios" la información viaje de forma vertical, aquí, el punto clave es que un tipo de información puede ser consultada a un servicio específico, este servicio a su vez, es independiente en recursos y es capaz de responder la solicitud sin depender de otros servicios. La ventaja principal de migrar a esta arquitectura es la posibilidad de independizar todo el sistema en servicios independientes interconectados entre sí; de esta forma por ejemplo, puede existir un servicio que se encargue únicamente de administrar los proyectos, un servicio para administrar las herramientas y otro servicio para ejecutar las tareas en HTCondor. alojarlo

Descripción del sistema usando Microservicios

En la actualidad tenemos un sistema que consta de varias características orientadas a suplir los requerimientos funcionales, estas se pueden expresar en forma de microservicios: servicio de administración de de proyectos, servicios de administración de herramientas, servicios de administración de tareas, servicio de autorización.

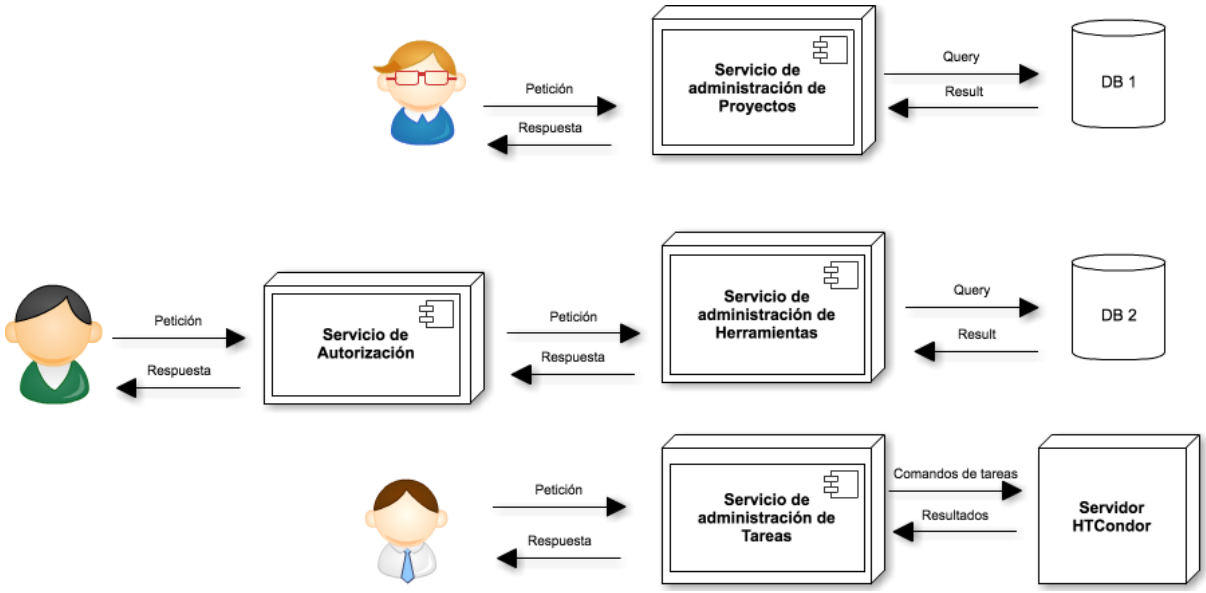


Figura 3. Posible representación del sistema actual como microservicios

Servicio de administración de proyectos

Este servicio albergará todo lo concerniente a la administración de los proyectos que el sistema tendrá, es un servicio pequeño con tareas muy específicas, crear, modificar y eliminar proyectos, puede tener una base de datos solo para él y no requiere de una máquina poderosa.

Servicios de administración de herramientas

Este servicio integra toda la administración de la información y archivos relacionados a las herramientas disponibles en el sistema, este servicio además de necesitar de una base de datos, necesita de almacenamiento disponibles para albergar los archivos relacionados a las herramientas.

Servicio de administración de tareas

Este servicio integra toda la administración de la información y archivos relacionados a las tareas y los resultados asociados a cada una de ellas, este servicio es el alma de sistema debido a que por medio de este sistema se establece comunicación directa con HTCondor y así conocer de primera mano toda la información disponible sobre las tareas en ejecución, recursos y resultados.

Servicio de autorización

Es necesario tener un servicio que se encargue únicamente de la autorización o usar un patrón que permita autorizar al usuario una sola vez para poder acceder así los diferentes servicios del sistemas, para este fin es posible implementar procedimientos tales como “Identificación única” o SSO[6] por sus siglas en inglés, por medio del estándar abierto OAuth 2.0[5]. Con SSO es posible acceder a todos los microservicios del sistema usando una instancia única de identificación; por su parte OAuth 2.0 es la segunda versión del protocolo OAuth 2.0 el cual implementa los conceptos de SSO, permitiendo compartir con

los usuarios recursos del sistema o acceder a los servicios por medio de una sola instancia de identificación.

OAuth 2.0 ha sido implementado por varias de las más grandes compañías de internet, tal es el caso de Google y Facebook quienes permiten a un usuario usar su servicio de autorización a aplicaciones propias y de terceros. En nuestro caso por medio de una servidor OAuth 2.0 es posible permitir a una aplicación web o una aplicación móvil acceder a cada uno de los servicios del sistema de la forma más sencilla posible.

Para entender cómo funciona el protocolo OAuth 2.0 debemos indagar en su terminología y conocer los actores que intervienen en el proceso. Tenemos al *Servidor de recursos*, el cual o los cuales albergan el o los servicios a los que deseamos acceder, tenemos a *El cliente*, el cual no es más que la aplicación web, de escritorio o móvil que desea acceder a los contenidos de nuestro *Servidor de recursos, Servidor de autorización* o *Proveedor de identidad,* este, como ya se ha hablado antes, es el encargado de autorizar a los clientes y de proveerlos de una identidad válida.

En un alto nivel y de forma simple este sería el flujo que tendría el sistema con microservicios implementado un servidor de autorización con OAuth 2.0. A modo de ejemplo usaremos el servicio de herramientas y el caso de un usuario sin autorización de usarlo.

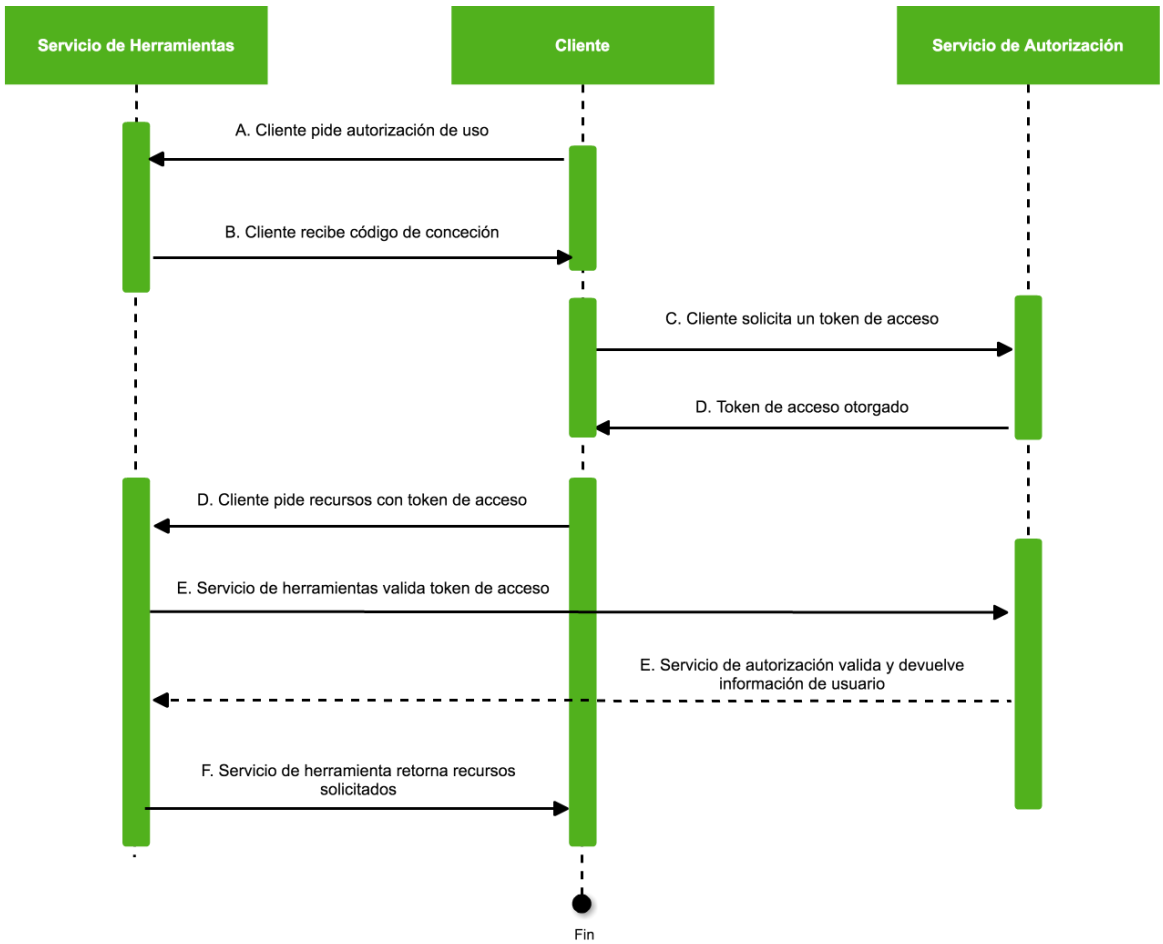


Figura 3. Flujo de sistema de microservicio implementando un flujo OAuth 2.0

A. El cliente intenta acceder al *servicio de herramientas*, pero como este servicio no es el encargado de la autenticación, es redirigido al servicio de autorización con una solicitud de autorización, el usuario presenta su credenciales de SIRIUS y se le pregunta si desea aprobar al *servicio de herramientas* a actuar en su nombre. Y es redirigido al servicio de herramientas.
B. El usuario obtiene un código de concesión de autorización como parte del redireccionamiento, el cual es transferido al cliente.
C. El cliente usa el código de concesión para pedir al servicio de autorización un token de acceso.
D. Si el código de concesión es válido, entonces *el servicio de autorizacion* retorna al cliente un token de acceso, el cual será usado por el cliente para acceder al servicio de herramientas y los demás.
E. *El servicio de herramienta* recibe una petición con token de acceso. El servicio pasa entonces a validar el token en *el servicio de autorización*. Si el token es válido, el *servicio de autorización* devuelve la información del usuario al *servicio de herramientas*.
F. Luego de validar la información del usuario *el servicio de herramientas* devuelve la información requerida por el cliente.

Hay otro flujos de trabajo con OAuth 2.0, los cuales están diseñados para aplicaciones móviles, aplicaciones de escritorio y aplicaciones del lado del servidor donde el cliente no está involucrado de forma directa.

La ventaja principal de los flujos de trabajo de OAuth 2.0 es que la comunicación entre el servicio de autenticación, el cliente y los servicios de recursos en sobre el protocolo HTTP.

Capítulo 2

En este capítulo se revisan los requisitos funcionales tienen la API.

Requisitos funcionales

Todas y cada unas de las funciones que un usuario puede realizar por medio de API están condensadas en este aparte. Para una idea general, tenemos a continuación los diagramas de Casos de Uso por roles de usuario

Diagrama 1: Casos de uso para el administrador del sistema

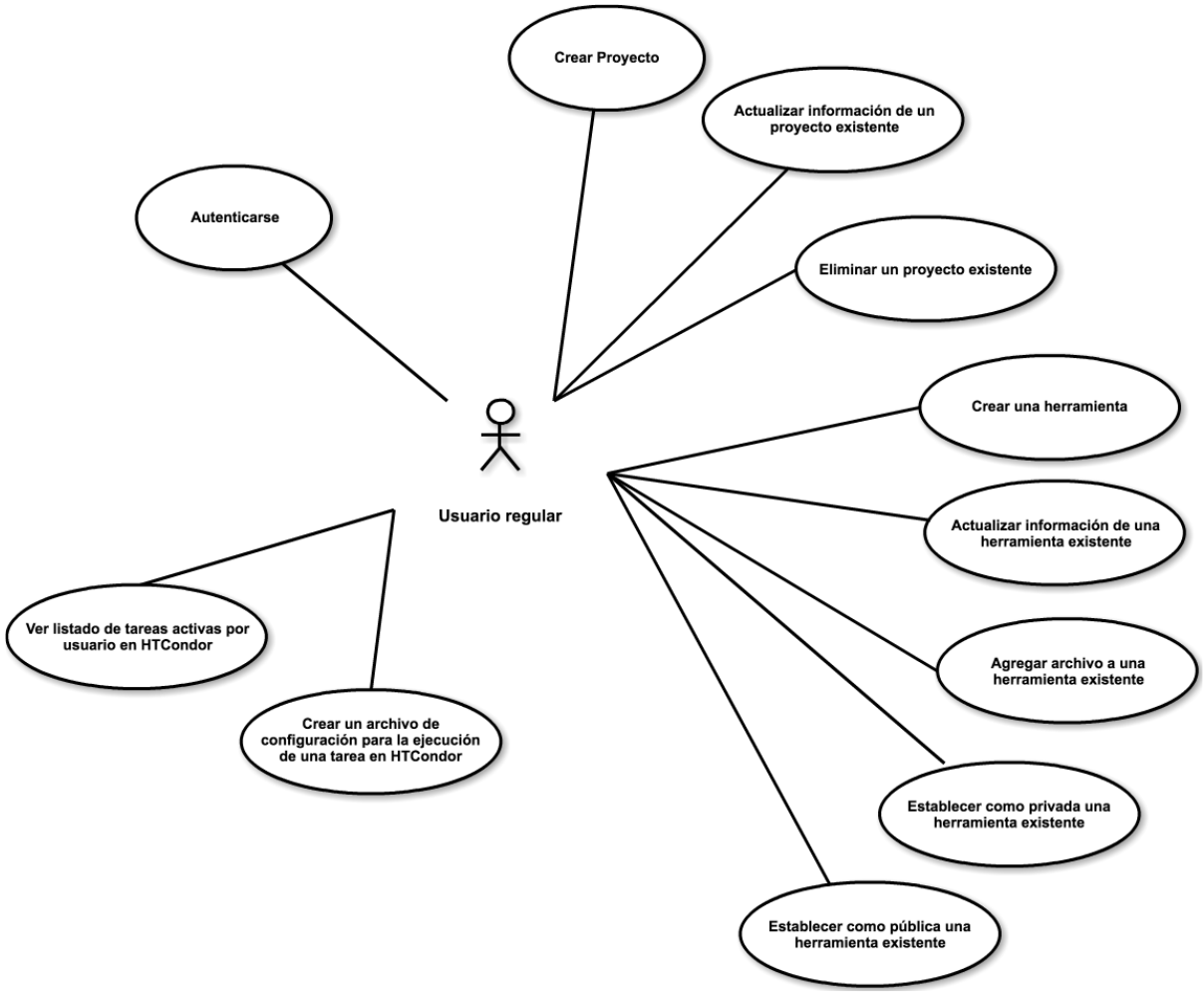


Diagrama 2: Casos de uso para un usuario regular

Autenticación

Por medio de una petición POST a una URL los usuarios del sistema podrán autenticarse y así tener un token JWT[4] de acceso para ejecutar las tareas propias del sistema.

Código

RF-0001

Actores

Usuario no autenticados en el sistema

Escenario

Un usuario desea usar el sistema.

Condiciones de éxito

Ser un usuario activo de la Universidad de Tecnológica de Bolívar y tener un código de usuario válido.

Condiciones de fracaso

No ser un usuario activo de la Universidad de Tecnológica de Bolívar o no tener un código de usuario válido.

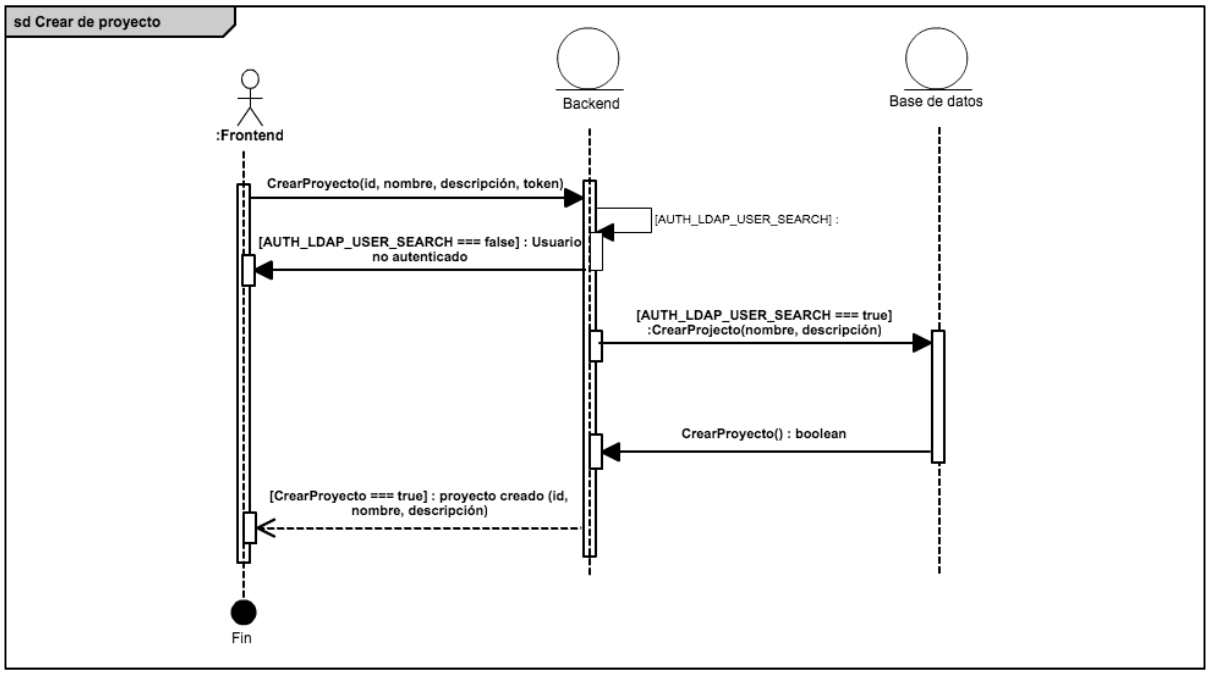
Resultado en caso de éxitos

- Respuesta HTTP con estatus 200 con un token de usuario.

Resultado en caso de fracaso

- Respuesta HTTP con estatus 40x con mensaje de error.

Diagrama de secuencia



Crear proyecto

Por medio de una petición POST a una URL los usuarios autenticados pueden crear proyectos para organizar las tareas y herramientas.

Código

RF-0002

Actores

Usuarios del sistema.

Escenario

Usuarios autenticados que desean usar el sistema e iniciar a organizar sus tareas y herramientas.

Condiciones de éxito

Los usuarios deben proveer la información necesaria para crear un proyectos, tal información es:

- Nombre del proyecto.
- Descripción del proyecto.
- Token de acceso.

Condiciones de fracaso

- No enviar el token de acceso.
- No enviar el nombre del proyecto.
- No enviar las propiedades como el nombre correcto.

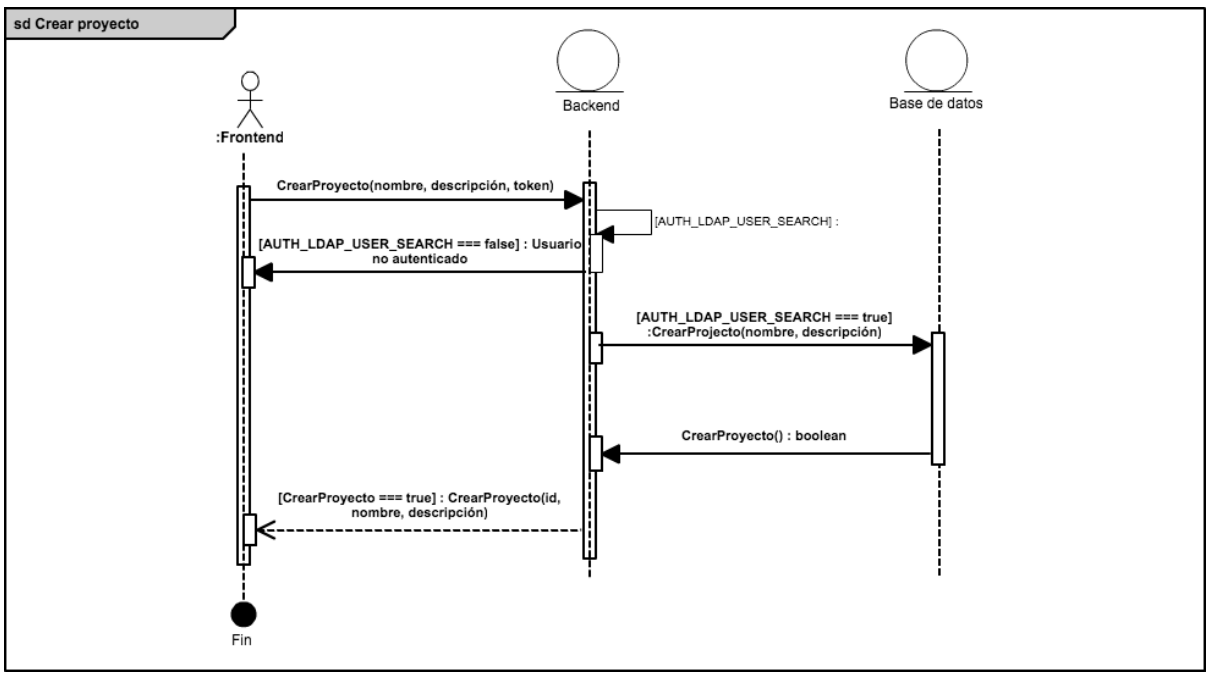
Resultado en caso de éxitos

- Respuesta HTTP con estatus 200 con la información del proyecto creado junto con un identificador único para ese proyecto.

Resultado en caso de fracaso

- Respuesta HTTP con estatus 4xx con mensaje de error de información faltante.
- Respuesta HTTP con estatus 5xx con mensaje de error por propiedad con nombre erróneo.
- Respuesta HTTP con estatus 50x con mensaje de error de usuario no autenticado.

Diagrama de secuencia



Actualizar la información de un proyecto existente

Por medio de una petición PUT a una URL los usuarios autenticados pueden actualizar la información de un proyecto ya existente.

Código

RF-0003

Actores

Usuarios autenticados.

Escenario

Un usuario autenticado desea actualizar la información de un proyecto ya existente.

Condiciones de éxito

Los usuarios deben proveer la información necesaria para actualizar un proyecto, tal información es:

- Nuevo nombre para el proyecto.
- Descripción para el nuevo proyecto.
- Identificador único del proyecto.
- Proyecto a actualizar esté asociado al usuario que ejecuta la acción.
- Token de acceso.

Condiciones de fracaso

- No enviar el token de acceso.
- Proyecto a actualizar no está asociado al usuario que ejecuta la acción.
- No enviar las propiedades correctas, tales como, el nombre.

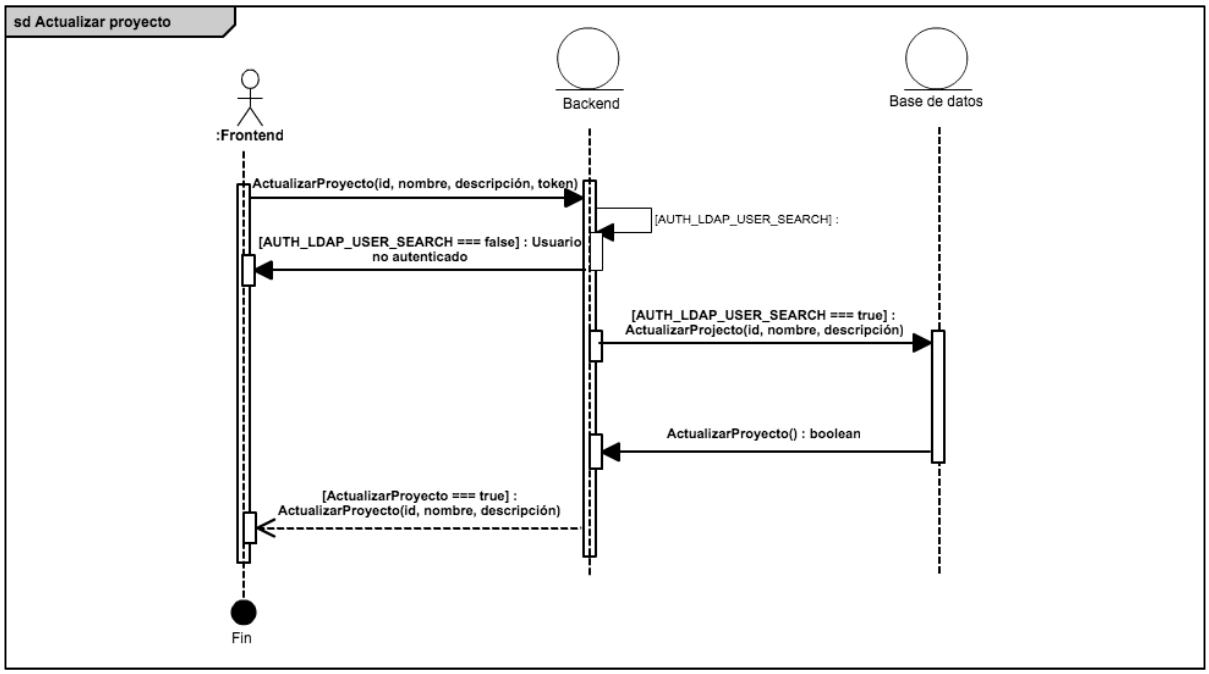
Resultado en caso de éxitos

- Respuesta HTTP con estatus 200 con la información actualizada del proyecto a actualizar.

Resultado en caso de fracaso

- Respuesta HTTP con estatus 5xx con mensaje de error de nombre de propiedades erróneos.
- Respuesta HTTP con estatus 4xx con mensaje de error de usuario no autenticado.
- Respuesta HTTP con estatus 4xx con mensaje de error de usuario no asociado al proyecto.
- Respuesta HTTP con estatus 5xx con mensaje de error de proyecto no encontrado.

Diagrama de secuencia



Eliminar un proyecto existente

Por medio de una petición DELETE a una URL, los usuarios autenticados pueden eliminar un proyecto existente.

Código

RF-0004

Actores

Usuario autenticado.

Escenario

Un usuario autenticado desea eliminar un proyecto existente.

Condiciones de éxito

El usuario administrador debe proveer la información necesaria para eliminar un proyecto existente, tal información es:

- Identificador único de un proyecto.
- Proyecto a eliminar está asociado al usuario que ejecuta la acción.
- Token de acceso.

Condiciones de fracaso

- No enviar el token de acceso.
- Proyecto a eliminar no está asociado al usuario que ejecuta la acción.
- No enviar identificador único de un proyecto.

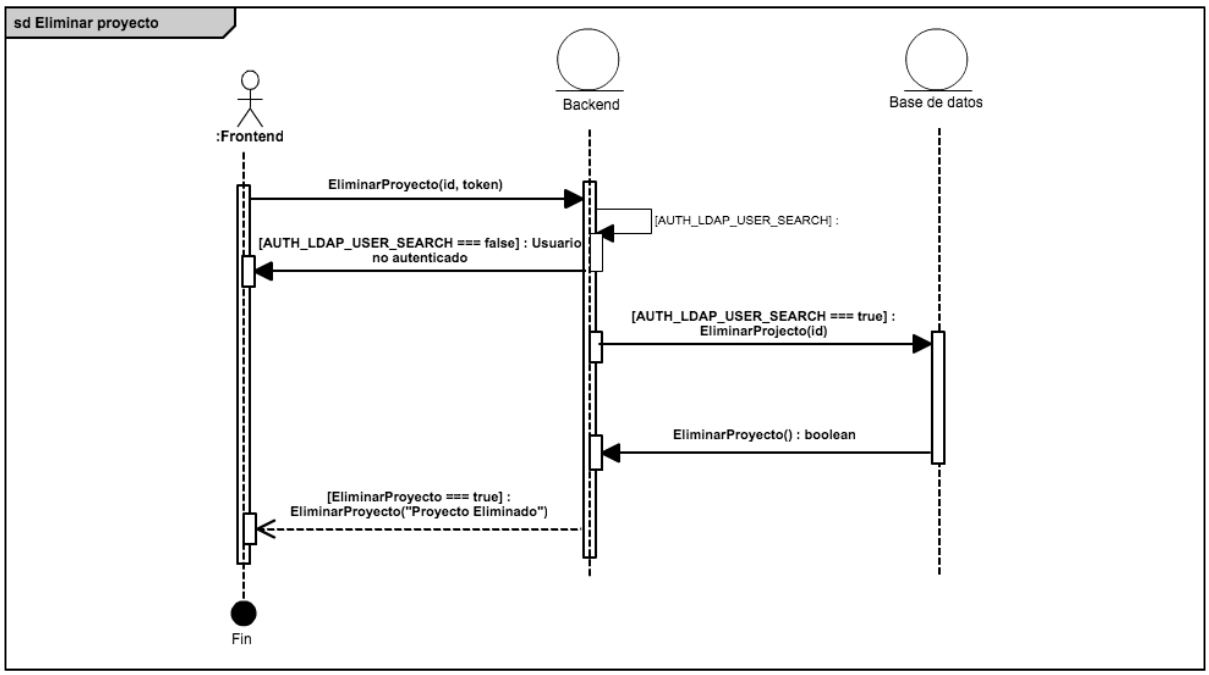
Resultado en caso de éxitos

- Respuesta HTTP con estatus 200 con el mensaje de proyecto eliminado.

Resultado en caso de fracaso

- Respuesta HTTP con estatus 4xx con mensaje de error de usuario no autenticado.
- Respuesta HTTP con estatus 5xx con mensaje de error de proyecto no encontrado.
- Respuesta HTTP con estatus 4xx con mensaje de error de usuario no asociado al proyecto.
- Respuesta HTTP con estatus 5xx con mensaje de error de proyecto no eliminada.

Diagrama de secuencia



Crear herramientas

Por medio de una petición POST a una URL, un usuario con rol de *Administrador* y autenticado puede crear una herramienta.

Código

RF-0005

Actores

Usuario con rol de *Administrador* autenticado.

Escenario

Un administrador del sistema desea crear una nueva herramienta para ser usada posteriormente por los demás usuarios al momento de ejecutar una tarea.

Condiciones de éxito

Los usuarios deben proveer la información necesaria para crear una herramienta, tal información es:

- Nombre
- Descripción
- Parámetros
- Token de acceso

Condiciones de fracaso

- No enviar el nombre de la nueva herramienta.
- No enviar token de acceso
- No enviar parametros

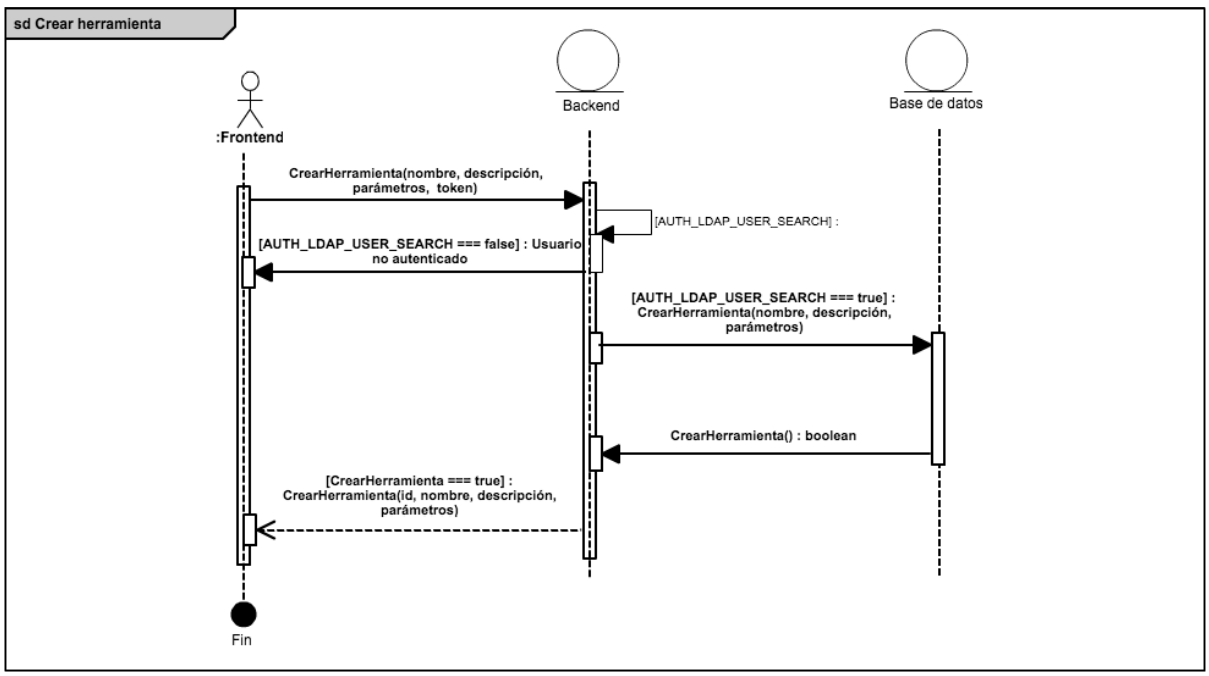
Resultado en caso de éxitos

- Respuesta HTTP con estatus 200 con la información de la nueva herramienta.

Resultado en caso de fracaso

- Respuesta HTTP con estatus 4xx con mensaje de error de información faltante.
- Respuesta HTTP con estatus 4xx con mensaje de error por propiedad con nombre erróneo.
- Respuesta HTTP con estatus 4xx con mensaje de error de usuario no autenticado.

Diagrama de secuencia



Actualizar información de una herramienta existente

Por medio de una petición PUT a una URL, un usuario con rol de *Administrador* y autenticado puede actualizar la información de una herramienta existente.

Código

RF-0006

Actores

Usuario con rol de *Administrador*.

Escenario

Un administrador del sistema desea actualizar una nueva herramienta ya existente para ser usada posteriormente por los demás usuarios al momento de ejecutar una tarea.

Condiciones de éxito

Los usuarios deben proveer la información necesaria para actualizar una herramienta, tal información es:

- Nuevo nombre.
- Nueva descripción.
- Nuevos parámetros.
- Token de acceso.
- Identificador único de la herramienta

Condiciones de fracaso

- No enviar el token de acceso.
- No enviar las propiedades correctas, tales como, el nombre.

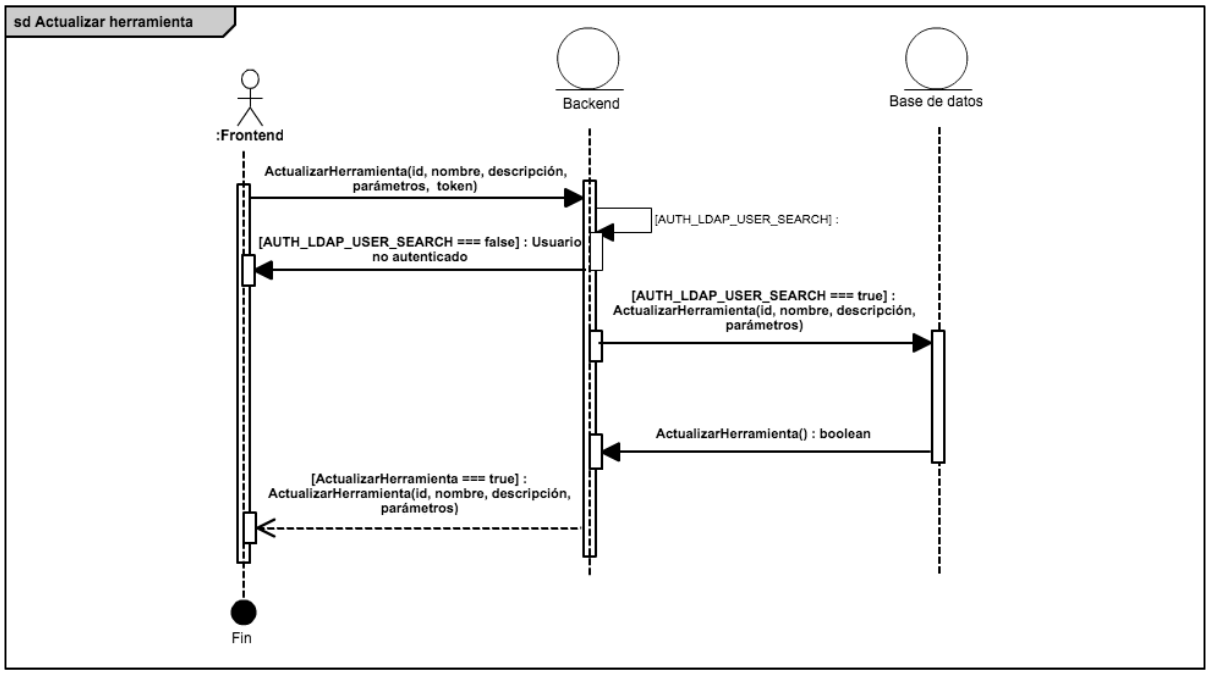
Resultado en caso de éxitos

- Respuesta HTTP con estatus 200 con la información actualizada de la herramienta a actualizar.

Resultado en caso de fracaso

- Respuesta HTTP con estatus 5xx con mensaje de error de nombre de propiedades erróneos.
- Respuesta HTTP con estatus 4xx con mensaje de error de usuario no autenticado.
- Respuesta HTTP con estatus 5xx con mensaje de error de herramienta no encontrada.

Diagrama de secuencia



Eliminar herramienta existente

Por medio de una petición DELETE a una URL, un usuario con rol de *Administrador* autenticado puede eliminar una herramienta existente.

Código

RF-0007

Actores

Usuario con rol de *Administrador* autenticado.

Escenario

Un administrador del sistema desea eliminar una herramienta ya existente para que no pueda ser usada por los demás usuarios al momento de ejecutar una tarea.

Condiciones de éxito

El usuario administrador debe proveer la información necesaria para eliminar una herramienta existente, tal información es:

- Identificador único de una herramienta.
- Token de acceso.

Condiciones de fracaso

- No enviar el token de acceso.
- No enviar identificador único de una herramienta.

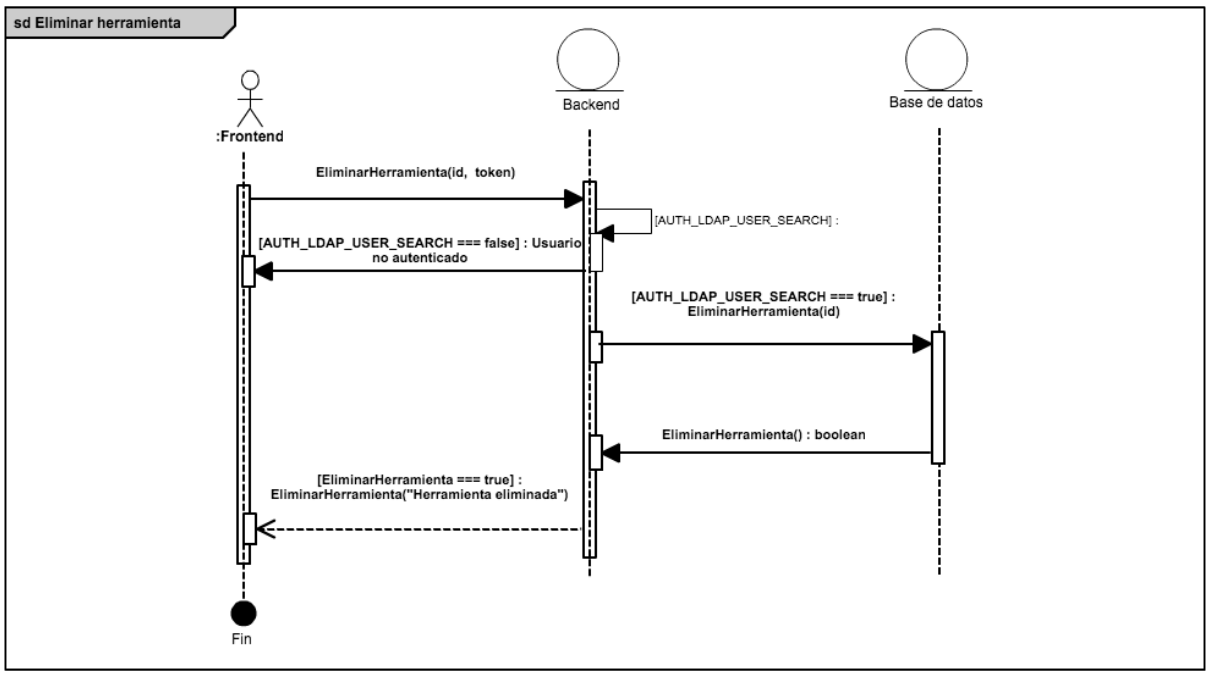
Resultado en caso de éxitos

- Respuesta HTTP con estatus 200 con el mensaje de herramienta eliminada.

Resultado en caso de fracaso

- Respuesta HTTP con estatus 40x con mensaje de error de usuario no autenticado.
- Respuesta HTTP con estatus 50x con mensaje de error de herramienta no encontrada.
- Respuesta HTTP con estatus 50x con mensaje de error de herramienta no eliminado.

Diagrama de secuencia



Agregar archivo a una herramienta existente

Por medio de una petición POST a una URL, un usuario puede agregar de forma individual los archivos que crea necesario para ejecutar su tarea en HTCondor usando una herramienta existente.

Código

RF-0008

Actores

- Usuario con rol de *Administrador*.
- Usuario regular.

Escenario

Un usuario autenticado desea agregar archivos que a consideración crea necesarios para una herramienta existente.

Condiciones de éxito

- ID herramienta a usar.
- Archivo a subir.

Condiciones de fracaso

- No enviar el token de acceso.
- Enviar archivos con un formato no admitido
- Enviar un archivo con un nombre

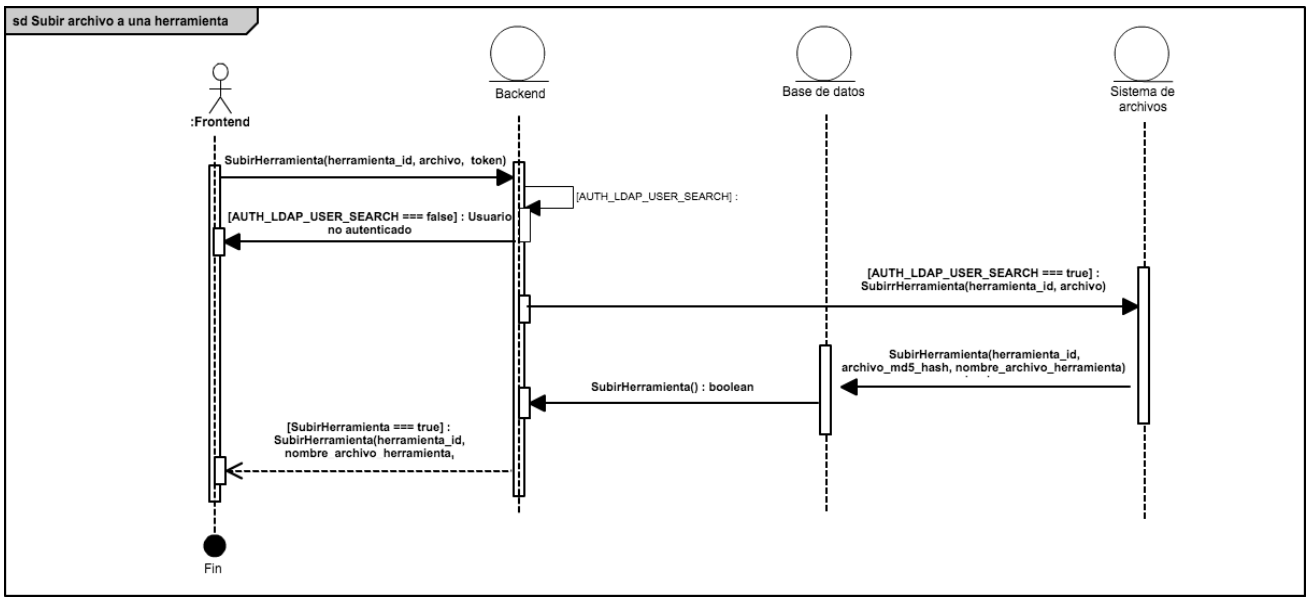
Resultado en caso de éxitos

- Respuesta HTTP con estatus 200 y mensaje de éxito.

Resultado en caso de fracaso

- Respuesta HTTP con estatus 5xx con mensaje de error por mal formato de archivo
- Respuesta HTTP con estatus 4xx con mensaje de error de usuario no autenticado.

Diagrama de secuencia



Establecer una herramienta como pública

Por medio de una petición PUT a una URL, un usuario con rol de *Administrador* autenticado puede establecer una herramienta como pública.

Código

RF-0009

Actores

Usuario con rol de *Administrador*.

Escenario

Un usuario administrador desea colocar como pública una herramienta existente en el sistema.

Condiciones de éxito

- Herramienta existente en el sistema

Condiciones de fracaso

- No enviar el token de acceso.
- Enviar el ID de una herramienta no existente

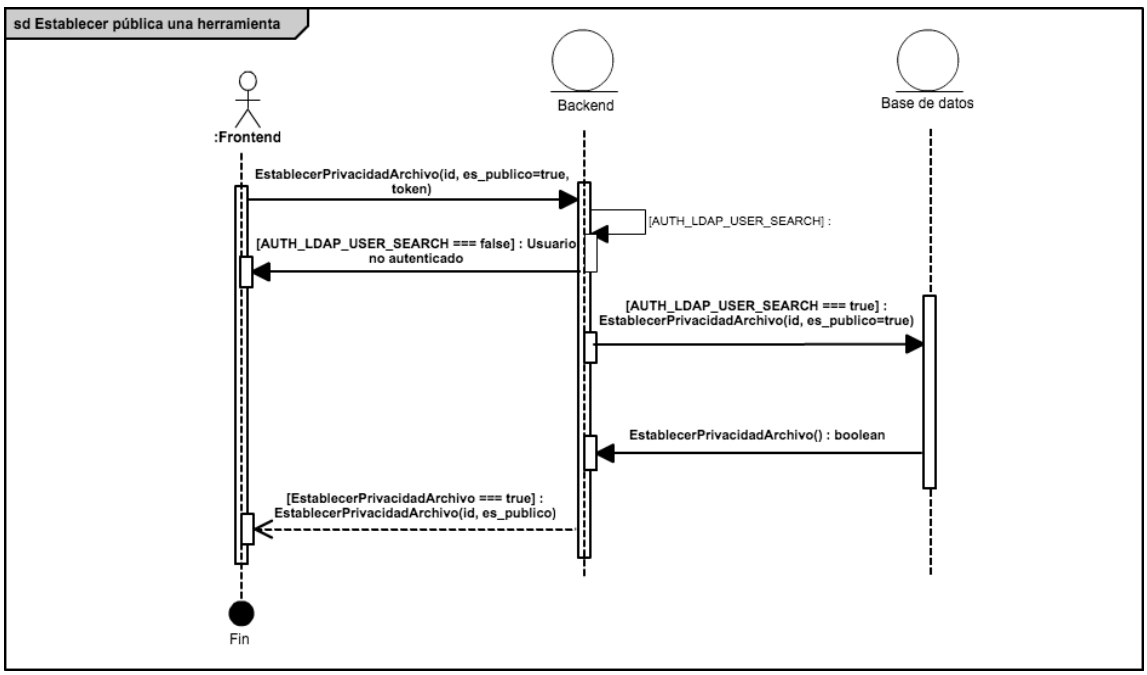
Resultado en caso de éxitos

- Respuesta HTTP con estatus 200 y la información actualizada de la herramienta a establecer como pública

Resultado en caso de fracaso

- Respuesta HTTP con estatus 4xx y mensaje de error de usuario no autenticado.
- Respuesta HTTP con estatus 5xx y mensaje de error de herramienta no encontrada.

Diagrama de secuencia



Establecer una herramienta como privada

Por medio de una petición PUT a una URL, un usuario con rol de *Administrador* autenticado puede establecer una herramienta como privada.

Código

RF-0010

Escenario

Un usuario administrador desea colocar como privada una herramienta existente en el sistema.

Actores

Usuario con rol de *Administrador*.

Condiciones de éxito

- Herramienta existente en el sistema

Condiciones de fracaso

- No enviar el token de acceso.
- Enviar el ID de una herramienta no existente

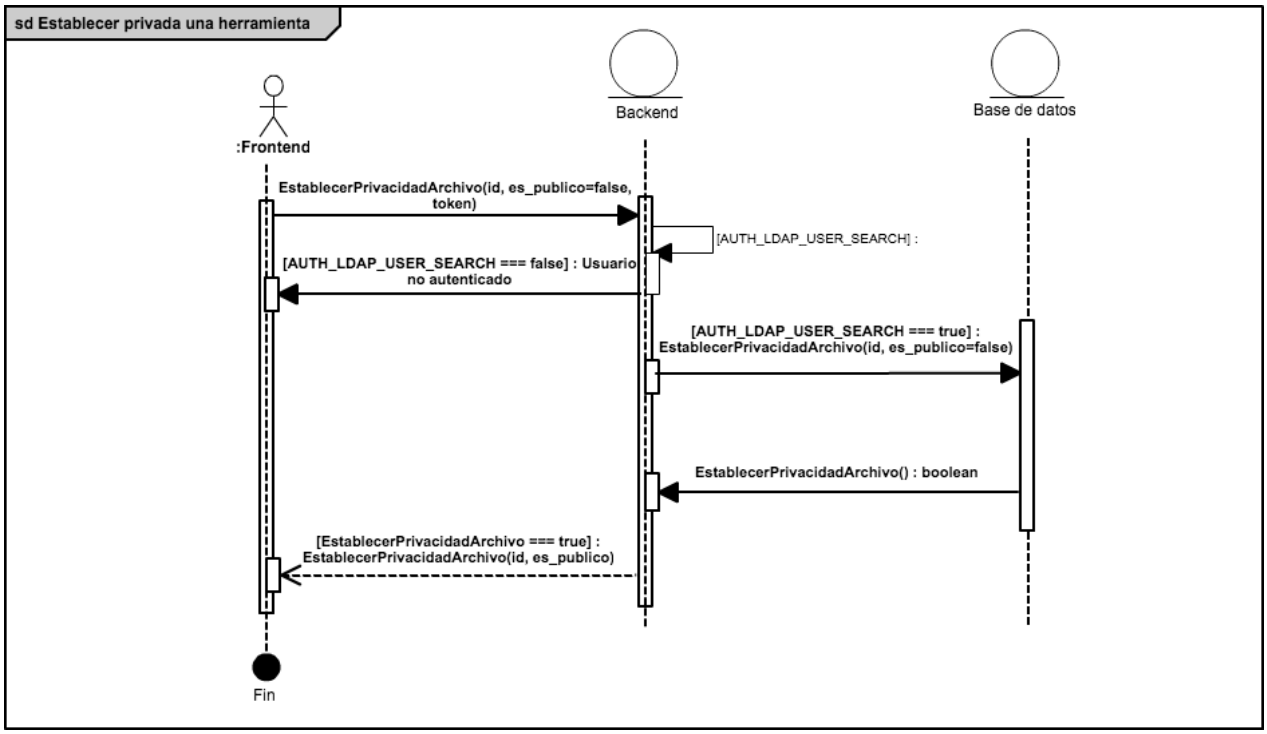
Resultado en caso de éxitos

- Respuesta HTTP con estatus 200 y la información actualizada de la herramienta a establecer como privada

Resultado en caso de fracaso

- Respuesta HTTP con estatus 4xx y mensaje de error de usuario no autenticado.
- Respuesta HTTP con estatus 5xx y mensaje de error de herramienta no encontrada.

Diagrama de secuencia



Crear de archivos de configuración para ejecutar tareas tipo “HTCondor” en HTCondor

Por medio de una petición POST a una URL, un usuario con rol de *Administrador* o *Regular*, puede crear un archivo de configuración que se usará para la ejecución de una tarea tipo “HTCondor” en HTCondor a partir de la información suministrada por el usuario, proyecto, herramientas y archivos a ejecutar.

Código

RF-0011

Actores

- Usuario con rol de *Administrador*.
- Usuario con rol *Regular*.

Escenario

Un usuario del sistema desea crear un archivo de configuración para pasar a ejecutar una o varias tareas en HTCondor

Condiciones de éxito

- ID de las herramientas a usar
- ID proyecto al que pertenece
- Token de acceso

Condiciones de fracaso

- No enviar el token de acceso.
- Enviar el ID de una herramienta no existente
- No enviar token de acceso

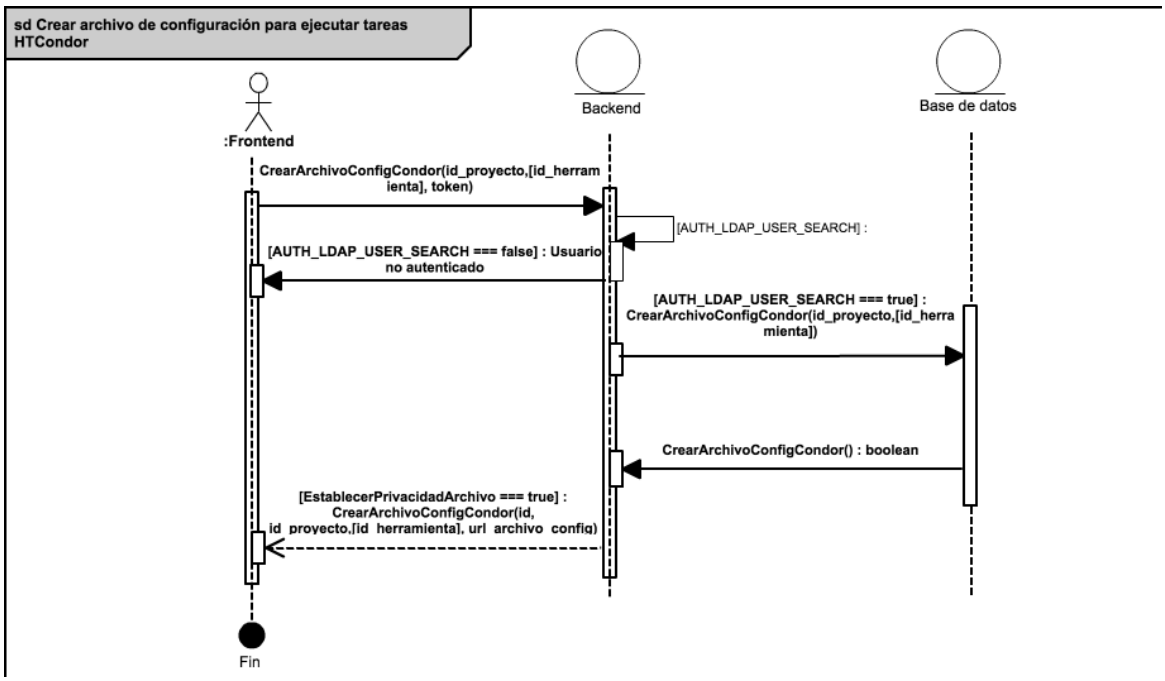
Resultado en caso de éxitos

- Respuesta HTTP con estatus 200 y la información del archivo recién creado.

Resultado en caso de fracaso

- Respuesta HTTP con estatus 4xx y mensaje de error de usuario no autenticado.
- Respuesta HTTP con estatus 5xx y mensaje de error de herramienta no encontrada.

Diagrama de secuencia



Ejecutar tareas usando archivos de configuración en HTcondor

Por medio de una petición POST, un usuario con rol de *Administrador* o *Regular*, puede ejecutar tareas en HTCondor usan un archivo de configuración existente.

Código

RF-0012

Actores

- Usuario con rol de *Administrador*.
- Usuario con rol *Regular*.

Escenario

Un usuario del sistema desea ejecutar una tarea, usando un archivo de configuración existente

Condiciones de éxito

- Archivo de configuración de la tarea a ejecutar
- ID del usuario que ejecuta la tarea.
- Token de acceso

Condiciones de fracaso

- Enviar archivo de configuración de tarea no existente
- No enviar token de acceso

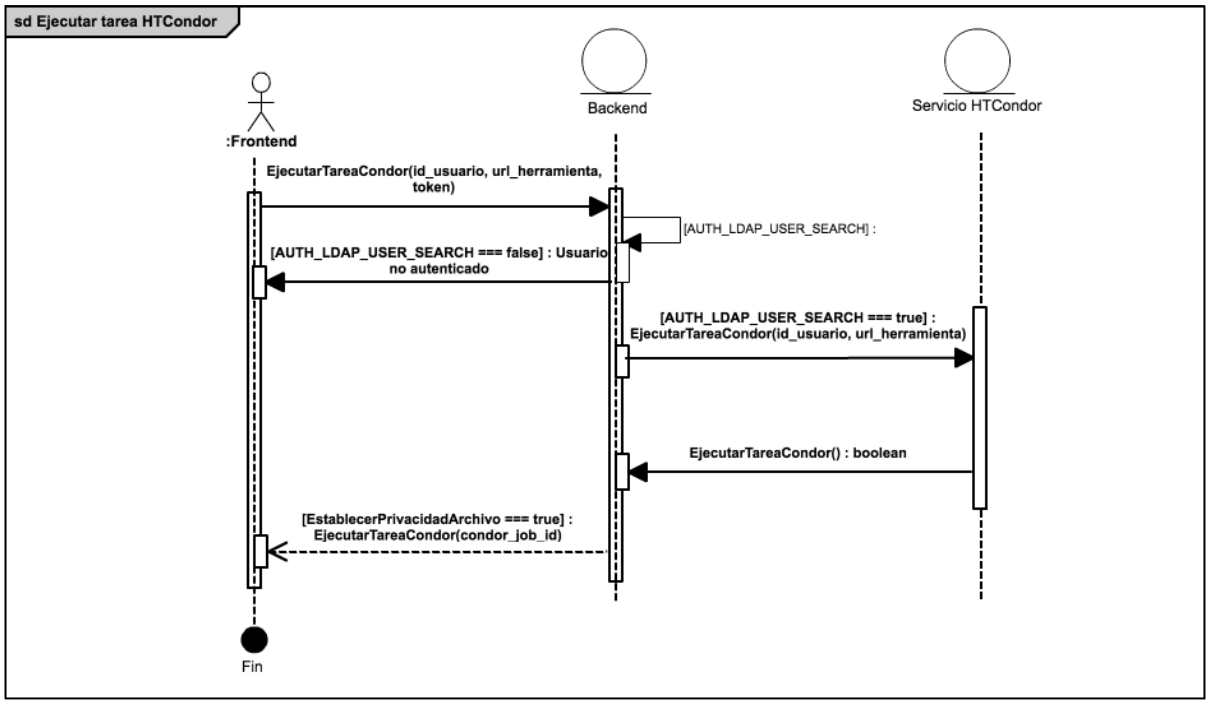
Resultado en caso de éxitos

- Respuesta HTTP con estatus 200 e identificador de la tarea HTCondor job en ejecución

Resultado en caso de fracaso

- Respuesta HTTP con estatus 4xx y mensaje de error de usuario no autenticado.
- Respuesta HTTP con estatus 5xx y mensaje de error de archivo de configuración.
- Respuesta HTTP con estatus 5xx y mensaje de error de HTCondor.

Diagrama de secuencia



Ver listado de tareas activas en HTCondor

Por medio de una petición GET, un usuario con rol de *Administrador*, puede obtener una lista de tareas activas en HTCondor.

Código

RF-0013

Actores

- Usuario con rol de *Administrador*.

Escenario

Un usuario con rol de *Administrador*, puede obtener una lista de tareas activas en HTCondor.

Condiciones de éxito

- Token de acceso

Condiciones de fracaso

- No enviar token de acceso

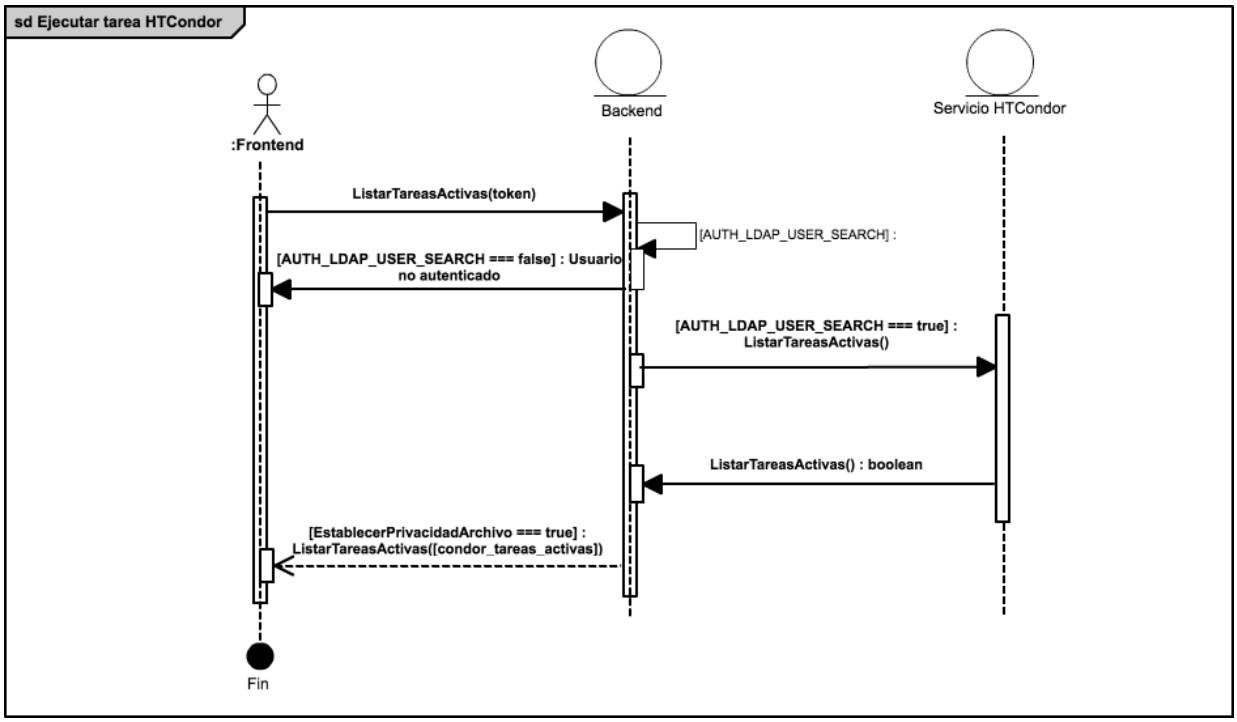
Resultado en caso de éxitos

- Respuesta HTTP con estatus 200 con un listado de las tareas activas y su información básica

Resultado en caso de fracaso

- Respuesta HTTP con estatus 4xx y mensaje de error de usuario no autenticado.
- Respuesta HTTP con estatus 5xx y mensaje de error de HTCondor.

Diagrama de secuencia



Ver listado de tareas por usuario HTcondor

Por medio de una petición GET, un usuario con rol de *Administrador* o rol *Regular*, puede obtener una lista de sus tareas activas en HTCondor.

Código

RF-0014

Actores

- Usuario con rol de *Administrador*.
- Usuario con rol *Regular*.

Escenario

Un usuario con rol de *Administrador*, puede obtener una lista de tareas activas en HTCondor.

Condiciones de éxito

- Token de acceso

Condiciones de fracaso

- No enviar token de acceso

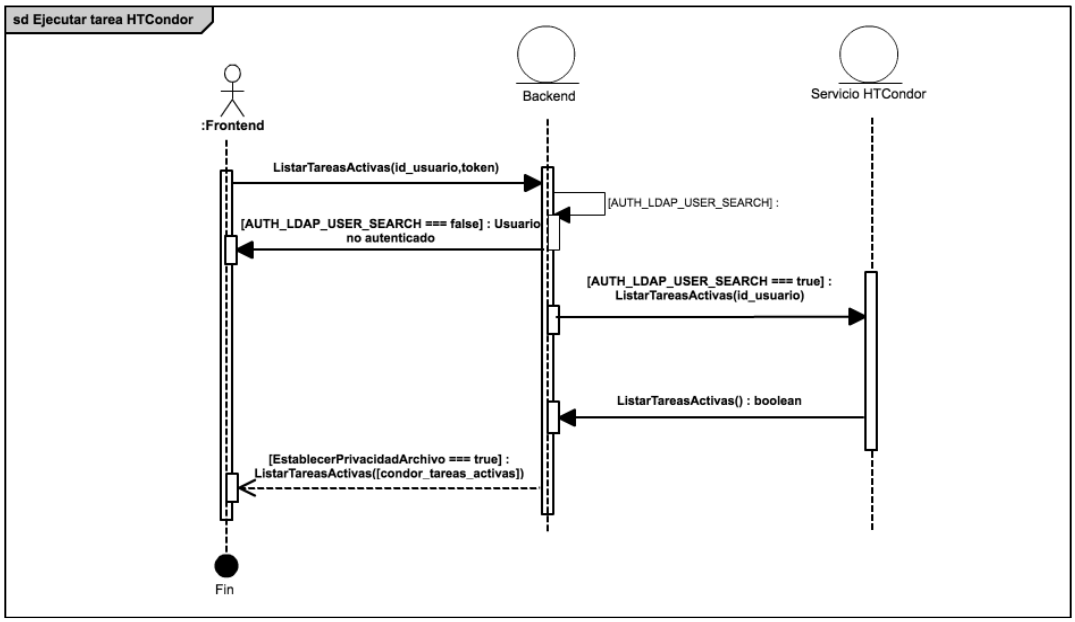
Resultado en caso de éxitos

- Respuesta HTTP con estatus 200 con un listado de las tareas activas y su información básica

Resultado en caso de fracaso

- Respuesta HTTP con estatus 4xx y mensaje de error de usuario no autenticado.
- Respuesta HTTP con estatus 5xx y mensaje de error de HTCondor.

Diagrama de secuencia



Cómo acceder al API

Como ya se ha mencionado muchas veces en los requerimientos funcionales, se accede los componentes de la API por medio de un *Localizador Uniforme de Recursos* o *URL[7]* (por su siglas en inglés) es la forma estándar por la cual se acceden a recursos en las redes, por ejemplo en internet. Para este caso se usará los principios usados en la metodología REST[8], por ejemplo, los objetos serán tratados como recursos, los cuales podrán ser aprovechados y accedidos usando cada uno de los verbos HTTP disponibles según sea el caso.

Requerimiento Funcional	**Verbo HTTP**	**URL**
RF-001	POST	/authentication/attempt
RF-002	GET	/Projects/
FR-002	GET	/Projects/[id],[id],[id]

RF-002	POST	/Proyectos/
RF-003	PUT	/Projects/[id]/update
RF-004	DELETE	/Projects/[id]/destroy
RF-004	DELET	/Projects/[id],[id],[id]/destroy
RF-005	POST	/Tools/
RF-006	PUT	/Tools/[id]/update
RF-007	DELETE	/Tools/[id]/destroy
RF-007	DELET	/Tools/[id],[id],[id]/destroy
RF-008	PUT	/Tools/[id]/uploadFile
RF-009	PUT	/Tools/[id]/updatePrivacy
RF-010	PUT	/Tools/[id]/updatePrivacy
RF-011	POST	/WorkflowFile/
RF-012	POST	Jobs/execute/WorkflowFile/[workflow_file_id]
RF-013	GET	Jobs/
RF-013	GET	Jobs/[id],[id],[id]
RF-014	GET	Users/[user_id]/jobs/
RF-014	GET	Users/[user_id]/jobs/[id],[id],[id]

Capítulo 3

Requisitos no funcionales

Rendimiento

Se espera que toda la comunidad de estudiantes de la UTB pueda hacer uso constante de esta API por ende se esperan aproximadamente 3000 usuarios entre estudiantes y profesores de forma no concurrente. Para esto hay habilitado un servidor con 4GB de memoria RAM, un disco duro de 20GB compartido con otros servicios.

Seguridad

Se usará un árbol LDAP de igual forma como se hace con el sistema SAVIO.

Mantenibilidad

Para asegurar el correcto funcionamiento del sistema, se utiliza actualmente Munin, por medio de esta herramienta, la capacidad de monitorear todas y cada una de las características dentro de los servidores que usa la API es un trabajo de análisis de los datos entregados por Munin. Es importante que la persona encargada tenga conocimientos en esta herramienta con el fin de dar un buen soporte e informar de forma eficaz sobre el rendimiento y las necesidades de las máquinas usadas por el sistema.

Apéndice

Bibliografía

[1]. G.~ Tel-zur. Pdc education in the bgu ece department. In Proceedings of the Workshop on Education for High-Performance Computing, EduHPC'14, pages 15-20, Piscataway, NJ, USA, 2014. IEEE Press.

[2]. M.~ Litzkow, M.~Livny , and M.~Mutka. Condor - a hunter of idle workstations. In proceeding of the 8th International Conference of Distributed Computing Systems, June 1988.

[3]. J.~ Yu and R.~ Buyya. A taxonomy of scientific workflow systems for grid computing. SIGMOD Rec., 34(3):44-49, Sept. 2005.

[4]. Tools.ietf.org. Rfc7519-json web token (jwt), 2015.

[5]. tools.ietf.org. rfc6749 The OAuth 2.0 Authorization Framework, 2012.

[6] en.wikipedia.org, Single sign-on, 2016.

[7] Tools.ietf.org. rfc1736 Functional Recommendations for Internet Resource Locators, 1995

[8] en.wikipedia.org, Representational state transfer, 2016.