**Cliente en Symmetry**

El cliente en symmetry se divide en cinco partes.

1. **proxy**

Es la capa usada por ***alergiaController.java*** del proyecto symmetry-web para guardar o modificar datos del usuario en SOLID. Llama a los servicios para ejecutar las siguientes operaciones:

1. delete(Long alergiaID): borra una alergia por su id.
2. findByPacienteId(Long userID): devuelve las alergias de un paciente por su id.
3. save(Iterable<AlergiaDto> entity) y save(AlergiaDto entity): guardan una o varias alergias.
4. update(AlergiaDto entity): actualiza una alergia.
5. find(Long alergiaId): devuelve la alergia por su id.
6. find(Long propietarioId, Long pacienteId): devuelve una lista de alergias por el id de paciente o propietario.
7. deleteAllergyFile(Long userID): borra el fichero alergias.ttl del paciente.
8. deleteUserFolder(Long userID): borra la carpeta del paciente de symmetry.
9. **service**

Es la capa de servicios que llama a las clases del paquete requests.

1. **requests**

Es el paquete que contiene las clases que ejecutan peticiones HTTP desde baryonyx.

* 1. ***get***

Gestiona dos peticiones GET.

*/symmetry/allergy/:id* : devuelve la alergia por su id.

*/symmetry/allergy/patient/:id* : devuelve las alergias del paciente por su id.

Además de ofrecer la posibilidad de obtener alergias por los ids de propietario y de paciente.

* 1. ***post***

Gestiona una petición POST.

*/symmetry/allergy* : crea una carpeta del usuario y su fichero de alergias si no existe en symmetry, y inserta en ello su alergia cuyos valores se pasan por el body en formato JSON; id alergia, id paciente, id propietario, nombre de alergia, observaciones.

* 1. ***delete***

Gestiona tres peticiones DELETE.

*/symmetry/file/allergy/:id* : borra el fichero de alergias del usuario por su id.

*/symmetry/user/:id*: borra la carpeta del usuario con el id de symmetry.

*/symmetry/allergy/:idcl/:idal* : borra la alergia por el id del paciente y el id de la alergia.

* 1. ***put***

Gestiona una petición PUT.

*/symmetry/allergy* : actualiza la alergia cuyos valores se pasan por el body; id alergia, id paciente, id propietario, nombre de alergia, observaciones.

1. **tests**

En esta carpeta, tenemos los tests que ejecutan pruebas exhaustivas del solidProxy y rutas comprobando que se guardan y se borran y se lean los datos del usuario en el pod correctamente.

1. **conf**

Contiene la clase que lee los datos del fichero config.properties que de momento contiene solamente la ip de symmetry. Es usado por la capa de requests.

**API Node**

Tenemos dividida la API en cuatro partes:

**1. Server.js**

Es el main que lanza la aplicación además de importar las dependencias usadas.

1. **app/routes**

Es la carpeta que contiene las rutas.

* 1. ***readRoutes.js***

Contiene dos rutas GET.

*/symmetry/allergy/:id* : devuelve la alergia por su id.

*/symmetry/allergy/patient/:id* : devuelve las alergias del paciente por su id.

* 1. ***writeRoutes.js***

Contiene una ruta POST.

*/symmetry/allergy* : crea una carpeta del usuario y su fichero de alergias si no existe en symmetry, y inserta en ello su alergia cuyos valores se pasan por el body; id alergia, id paciente, id propietario, nombre de alergia, observaciones.

Y una ruta PUT.

*/symmetry/allergy* : actualiza la alergia cuyos valores se pasan por el body; id alergia, id paciente, id propietario, nombre de alergia, observaciones.

* 1. ***deleteRoutes.js***

Contiene tres rutas DELETE.

*/symmetry/file/allergy/:id* : borra el fichero de alergias del usuario por su id.

*/symmetry/user/:id*: borra la carpeta del usuario con el id de symmetry.

*/symmetry/allergy/:idcl/:idal* : borra la alergia por el id del paciente y el id de la alergia.

1. **app/modules**

Contiene el fichero gestorSOLID.js que se encarga de todas las operaciones pedidas por las rutas.

Tiene los siguientes métodos:

1. *Leer :* devuelve una o varias alergias.
2. *ExistFolder :* comprueba si el usuario existe en symmetry.
3. *WriteInFolder :* inserta una o varias alergias en el pod.
4. *UpdateAllergy :* actualiza la alergia en el pod.
5. *deleteAllergy :* borra la alergia del fichero alergias.ttl en el pod.
6. *deleteAllergyFile :* borra el fichero alergias.ttl de la carpeta del usuario.
7. *deleteUserFolder :* borra la carpeta del usuario en symmetry.
8. *executeSPARQLUpdate:* ejecuta la consulta SPARQL, usado por otros métodos.
9. **tests**

En esta carpeta, tenemos el fichero routes.test.js que contiene pruebas exhaustivas de las rutas comprobando que se guardan y se borran y se lean los datos del usuario en el pod correctamente. Se ejecutan con npm test en la raíz de la API.

**Estructura de la API**

Emplea el arquetipo de Node encomendado. Se ha eliminado la librería Mongoose así como el usuario derivado, y mantenido el *healthcheck* así como la configuración para *Graylog,* según lo comentado.

En la configuración (*core/config*) se establece el puerto en que se despliega la API así como la URL en la que se encuentra el servidor de PODs y el directorio *Symmetry.* Por defecto, la API se despliega en el puerto 8440 y la URL es [*https://localhost:8443/symmetry*](https://localhost:8443/symmetry), puesto que el servidor se despliega en el mismo contenedor y en el puerto 8443, por defecto. Si fuera un servidor multiusuario, la URL sobre la que se realizan las operaciones sería del estilo *user.localhost:8443/symmetry.*

**Node Solid Server**

Se alberga en /home/node-modules/solid-server. Los siguientes subdirectorios son de especial interés:

* Bin. Se emplea el script *solid* para configurar el servidor - *./solid init –* y arrancarlo - *./solid start* -. La configuración se alberga en *config.json*, en el directorio raíz. *Nota: debemos situarnos en /bin para arrancar el servidor si queremos que utilice la configuración establecida. Si se arranca desde / empleando la ruta completa, no encontrará config.json y iniciará el servidor en el puerto por defecto, 3456.*
* Data. Aquí se alberga el sistema de archivos del servidor. Puesto que el empleado para la prueba de concepto es monousuario, consta directamente de las carpetas del mismo. Concretamente, la más relevante es *symmetry*, que contiene a su vez las carpetas de los usuarios donde se almacenan las alergias. El archivo *acl* que se encuentran en las carpetas configura la privacidad de la mismas. Un sistema multiusuario añade una capa más al sistema de archivos; bajo *data* se encuentran los directorios de cada usuario, en cada cual encontramos los correspondientes subdirectorios (*public, private, symmetry...).*
* Keys. Contiene los certificados empleados para configurar el servidor. Usar certificados autofirmados no es posible, ya que las librerías de solid bloquean las peticiones fetch; de modo que se ha empleado la herramienta *mkcert* (/home/mkcert) para crearlos.

En la configuración actual, se emplea el puerto 8443.

El contenedor *baryonyx* inicia automáticamente tanto la API como el servidor de PODs al arrancarlo.

