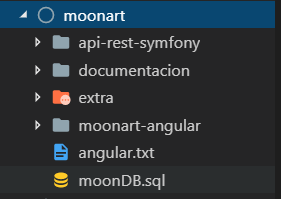
Estructura del proyecto



Notas: Si ves en el código moonart-api.io, es un host virtual de la api rest sólo.

Para ejecutar la web hay que seguir unos pasos, pero no tengo muy claro del todo cómo se haría, ya que no te vale con tener Angular CLI instalado, tienes que importar el proyecto. A mí me va porque lo he creado desde mi pc, pero todos los módulos que importé no los tendrías instalados en el proyecto. Creo que todo eso se guarda en el angular.json, y que al importar el proyecto te leerá el angular.json y te instalará todo lo que necesita, pero no estoy seguro. Eso es mejor mirarlo en un tutorial 😊.

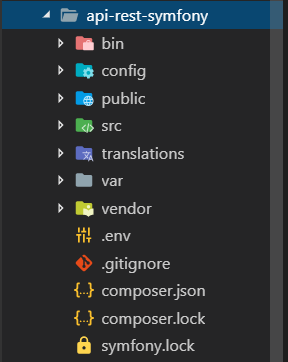
Luego tendrías que definir un vhost con tu localhost donde vayas a ejecutar la api rest (yo usé wamp), referenciándolo a moonart-api.io.

Angular lo tienes que lanzar desde la línea de comando, yendo al path donde está el proyecto angular (dentro).

Se ejecuta con el comando “ng serve”. Pero para que vaya bien del todo en mi caso, hay que hacer “ng serve –live-reload false”, para evitar la recarga automática, que lo hace angular por defecto en el modo developer.

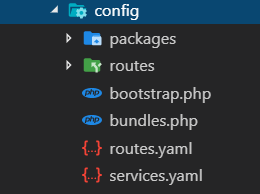
A la hora de conectar la api rest con el frontend, igual te da problemas con el CORS (Cross-Origin Request no se qué). No me acuerdo muy bien de cómo lo solucioné, pero sí que me acuerdo de que me costó. Así que si te pasa y no lo consigues, intento ayudarte. ☻

Backend con Symfony



Carpetas importantes:

* config



En routes.yaml están los enrutados que conectan las funciones con sus direcciones.

user:

path: /user

controller: App\Controller\UserController::index

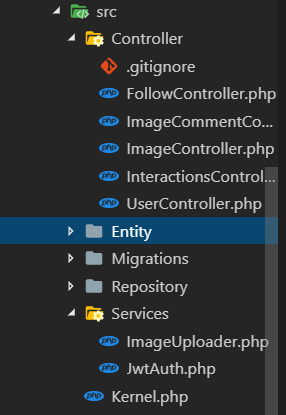
methods: [POST]

Aquí por ejemplo decimos que la ruta apirest.com/user va a mostrar aquello que mostremos con nuestra función index dentro de nuestro controlador UserController.

Debajo le indicamos que los parámetros se los pasaremos por POST.

Es la ruta de una llamada a la api rest, por tanto desde Angular tendremos que hacer una petición AJAX por POST. Si no coinciden, no va a ir.

* public: Es donde almaceno las imágenes que se suben
* src:



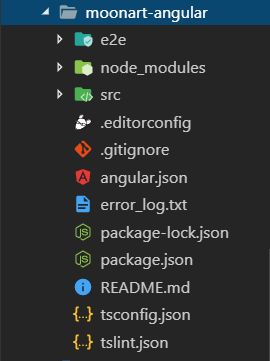
En Controller guardamos los controladores con sus funciones.

En Entity es donde se generan nuestras clases con Doctrine.

En Services hay dos servicios, el de subir imágenes y el de autentificación por tokens.

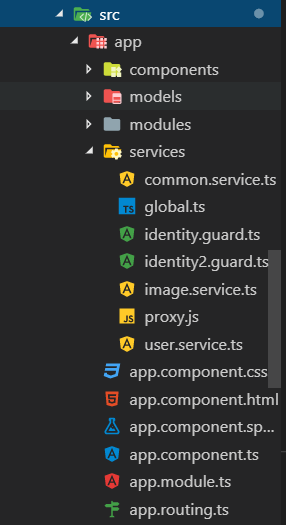
(Esos son casi todos los que toqué, el resto no me hizo falta, y lo generaba Symfony ya todo)

Frontend con Angular



Carpetas importantes:

Aquí la carpeta más importante es src.



* app/components: Aquí es donde guardo los “bundles”. En Angular se llaman components creo.

Los componentes son lo que forman la página web, la vista. Puedes tener varios componentes en una misma vista, pero creo que lo normal es un componente por vista. Cada componente generado con línea de comandos tiene su html, su typescript y su css.

* app/models: El modelo de nuestros objetos. Éste creo que se tiene que hacer manualmente, ya no me acuerdo. Tiene que concordar con nuestra base de datos, o al menos el objeto que le pasemos debe tener campos válidos.
* app/services: Aquí es donde conectamos a la api rest con la api. (Mirar image.service.ts por ejemplo).

checkInteractions(token, data): Observable<any> {

let json = JSON.stringify(data);

let params = 'json=' + json;

let headers = new HttpHeaders().set('Content-Type', 'application/x-www-form-urlencoded')

.set('Authorization', token);

return this.\_http.post(this.url + 'check', params, { headers: headers });

}

Creamos la función checkInteractions por ejemplo (en typescript se declaran sin el function delante). En este caso le pasamos el token (queremos que mire interacciones del usuario, por lo que le pasamos el usuario, que va en el token junto con otros datos que hemos definido en el JwtAuth.php). Definimos el json que le vamos a pasar con los datos, y la cabecera, indicándole qué tipo de dato es.

El return this.\_http.post… es la llamada AJAX. A estas funciones las llamaremos desde nuestros componentes. Es importante en Angular al menos llamarlas con el método subscribe (checkInteractions(this.token, this.data).subscribe()), porque si no es como que lo hace, pero no te devuelve nada, o igual ni lo hace.

Los métodos y sus usos más comunes son:

**POST**: Cuando queremos hacer un INSERT, o subir algo, como una imagen.

**GET**: Cuando queramos recoger datos de la api rest.

**PUT**: Cuando queramos hacer un UPDATE.

**DELETE**: Cuando queramos hacer un DELETE.

La página se mostrará desde el index.html de src. Éste llamará a <app-root>, que está en src/app/app.component.html, y ya desde aquí iremos cargando los componentes.

En app.module.ts cargamos los módulos (creo que lo iba haciendo automáticamente Angular)

En app.routing.ts definimos las rutas, y en mi caso creé IdentityGuards, para definir quién podía entrar donde. (Un usuario no registrado no puede subir imágenes, así que se le redirige al login, y un usuario registrado no puede ir a la ruta de login, así que le devuelve a home)