Token JWT

Le principe est simple :

- L'utilisateur se connecte, Symfony génère un Token JWT.
- Android le stocke
- Android envoie le token dans les requêtes APi
- Symfony le vérifie

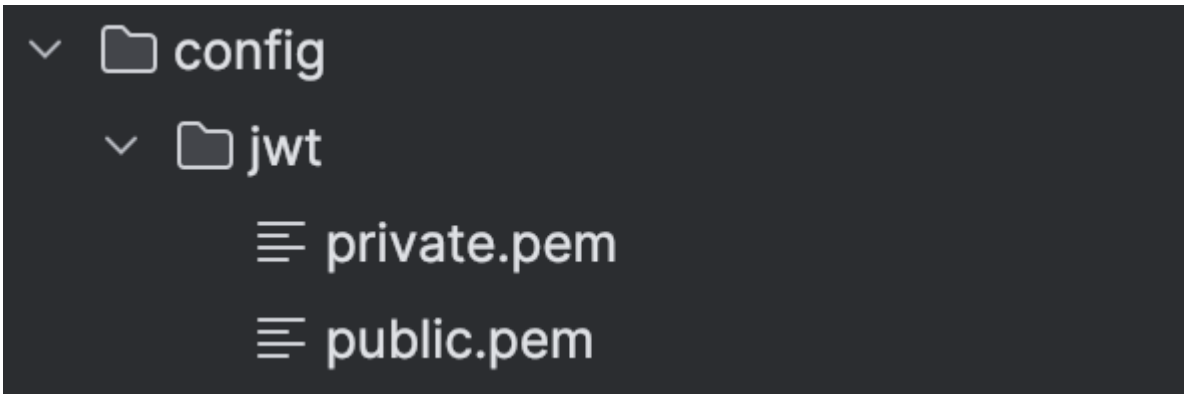
Tout d'abord sur Symfony, installer :
composer require lexik/jwt-authentication-bundle
C'est le bundle de génération de tokens

```
lea@mac notes-de-frais % composer require lexik/jwt-authentication-bundle
./composer.json has been updated
Running composer update lexik/jwt-authentication-bundle
Loading composer repositories with package information
Updating dependencies
Nothing to modify in lock file
Writing lock file
Installing dependencies from lock file (including require-dev)
Nothing to install, update or remove
```

Il faut ensuite générer des clés
php bin/console lexik:jwt:generate-keypair

```
lea@mac notes-de-frais % php bin/console lexik:jwt:generate-keypair
```

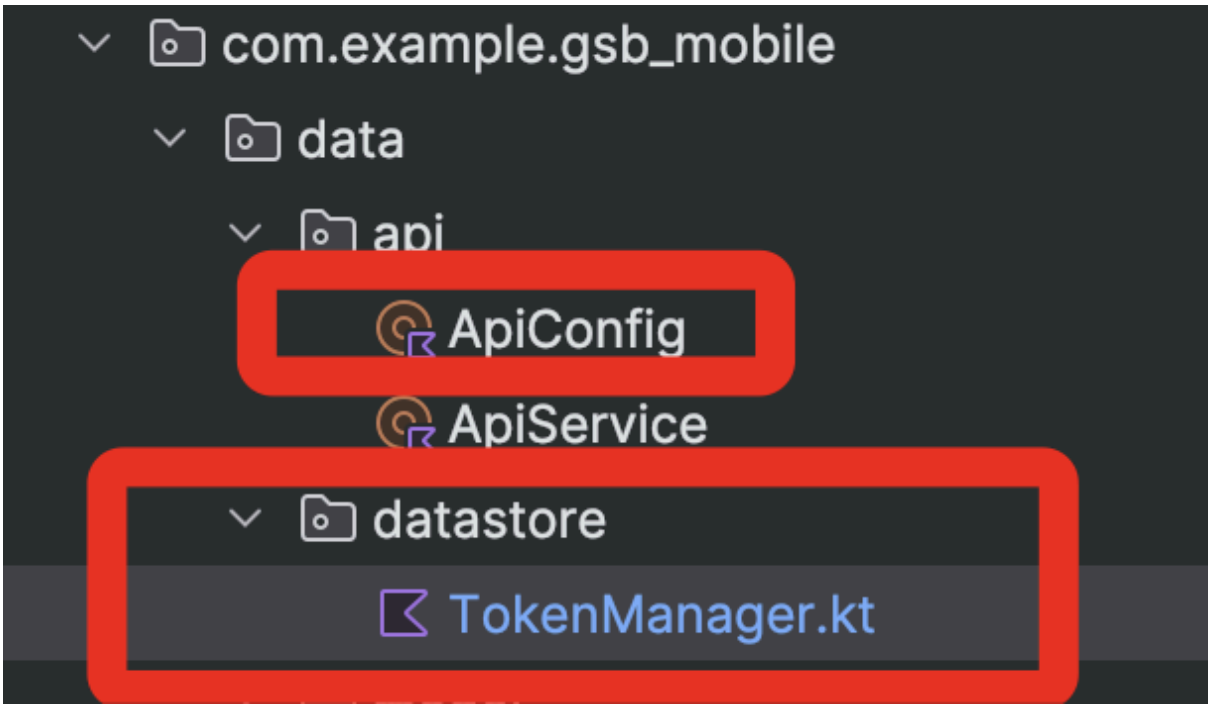
Cela va créer deux fichiers avec ces clés
config/jwt/private.pem
config/jwt/public.pem



On indique ensuite au fichier d'environnement la configuration des token JWT :
JWT_SECRET_KEY=%kernel.project_dir%/config/jwt/private.pem
JWT_PUBLIC_KEY=%kernel.project_dir%/config/jwt/public.pem
JWT_PASSPHRASE=tonpassphrase
le passphrase est normalement demandé à la génération des keypair, sinon elle peut être au choix de l'utilisateur

```
###> lexik/jwt-authentication-bundle ###
JWT_SECRET_KEY=%kernel.project_dir%/config/jwt/private.pem
JWT_PUBLIC_KEY=%kernel.project_dir%/config/jwt/public.pem
###< lexik/jwt-authentication-bundle ###
```

Côté projet Kotlin:
2 fichiers vont être importants à créer
data/api/ApiConfig
data/datastore/TokenManager.kit



ApiConfig :
Le fichier va stocker l'IP de la machine hébergeant le projet web

ApiConfig.kt

```kotlin
package com.example.gsb_mobile.data.api

10 Usages
object ApiConfig {
    10 Usages
    const val BASE_URL = "http://192.168.107.224"
}
```

De cette manière, il suffit juste d'appeler la constante "BASE_URL" lorsque ceci est nécessaire. Ici on récupère l'URL pour pointer sur l'endpoint côté Symfony pour afficher les barèmes/référentiels.

```
val request = Request.Builder()
    .url( url = "${ApiConfig.BASE_URL}/api/bareme")
    .addHeader( name = "Authorization", value = "Bearer $token")
    .build()
```

TokenManager

Le fichier va stocker le token dans une configuration qu'on pourra appeler à la connexion :

```
package com.example.gsb_mobile.data.datastore

import android.content.Context
import androidx.datastore.preferences.core.*
import androidx.datastore.preferences.preferencesDataStore
import kotlinx.coroutines.flow.first

val Context.dataStore by preferencesDataStore(name =
"user_prefs")

object TokenManager {

    private val TOKEN_KEY = stringPreferencesKey("jwt_token")

    suspend fun saveToken(context: Context, token: String) {
        context.dataStore.edit { it[TOKEN_KEY] = token }
    }

    suspend fun getToken(context: Context): String? {
        return context.dataStore.data.first()[TOKEN_KEY]
    }

    suspend fun clearToken(context: Context) {
        context.dataStore.edit { it.remove(TOKEN_KEY) }
    }
}
```

Ici, dans la fonction connexion on récupère le token

```
val token = JSONObject(bodyString).getString("token")

CoroutineScope(Dispatchers.IO).launch {
   TokenManager.saveToken(context, token)
   Session.clear()
   ApiService.fetchCurrentUser(context)
   withContext(Dispatchers.Main) {
       callback(true)
   }
}
```