Conventions du projet annuel

Partie 1 : Git

a) La branche principale est « **main** ».
C'est la branche de production.

b) Chaque nouveau développement devra être dans la branche « **develop** ».
Exemple : **develop/homepage**
Les tests seront faits sur cette branche avant de merge avec la branche de production.

c) Chaque nouvelle fonctionnalité devra être dans la branche « **feature** ».
Exemple : **feature/login**
Une fois une fonctionnalité finalisée, elle sera merge avec la branche « **develop** ».

d) Chaque nouvelle version devra être dans la branche « **release** ».
Exemple : **release/1.0**
La version s'incrémente dès qu'il y a une mise en production (merge avec « **main** »).

e) Chaque correction de bug devra être dans la branche « **hotfix** ».
Exemple : **hotfix/login**
Une fois un bug résolu, il faudra merge avec « **develop** ».

Partie 2 : Développement de l'application web

Pour développer l'application web nous utiliserons le Framework **Symfony**.
Pensez à se documenter régulièrement sur https://symfony.com/doc/current/index.html.

Variables

Toutes les variables seront déclarées en **anglais** et en **camelCase**.
Utiliser des noms de variables logiques.
Déclaration des attributs public et privés sous la forme « **$firstName** ».
Déclaration des attributs protégés sous la forme « **$_firstName** ».
Déclaration des constantes en **SCREAMING_SNAKE_CASE**.
Exemple : « **MY_VAR** ».

Classes

Déclaration des classes en **PascalCase**.
Le nom du fichier doit être identique au nom de la classe.
Toujours une classe par fichier (même si héritage).
Utiliser les « **namespace** ».

Méthodes

Typer les paramètres des méthodes ainsi que le retour (return).
L'annotation doit bien être à jour avec la méthode (la mise à jour peut se faire automatiquement grâce à un raccourci disponible sur de nombreux IDE).
Exemple :

```php
/**
 * @Route("/signup", name="signup")
 * @param Request $request
 * @return Response
 */
public function signup(Request $request): Response {}
```

Rendre les méthodes les plus génériques possibles afin de ne pas faire de la duplication de code.
Limiter le nombre de paramètres : s'il y a trop de paramètres, utiliser un tableau.
Déclaration des méthodes publiques sous la forme « **public function myFuntion() {}** ».
Déclaration des méthodes privées et protégées sous la forme « **private function _myFuntion() {}** ».
Les « **getters** » et les « **setters** » doivent être déclarés sous la forme « **getSomething** » et « **setSomething** ».
Veillez à bien sauter 2 lignes entre chaque méthode.

Routing

Chaque « **route** » sera déclarée grâce à une annotation au-dessus de chaque « **controller** ».

Indentations et espaces

Ne pas utiliser d'espaces entre des parenthèses et du code.
Exemple, à ne pas reproduire :

```php
if( $this->getUser( ) )
```

Exemple, à reproduire :

```php
if($this->getUser())
```

Mettre les accolades à la ligne lors de la déclaration d'une méthode.
Exemple :

```php
public function login(Request $request): Response
{
    echo "test";
}
```

Bien indenter les tableaux associatifs.
Exemple :

```php
[
    'user' => $user,
    'form' => $form->createView(),
    'current_page' => 'signup'
]
```

Base de données

Toutes les informations dans la base de données seront en anglais.
Les tables seront nommées en **SCREAMING_SNAKE_CASE** et toujours au singulier.
Les colonnes seront nommées en **snake_case**.

Divers

Lors d'une condition, utiliser les **comparaisons de types** (« === »), si possible.
Bien documenter ses développements.
Essayez d'avoir un code le plus lisible possible.