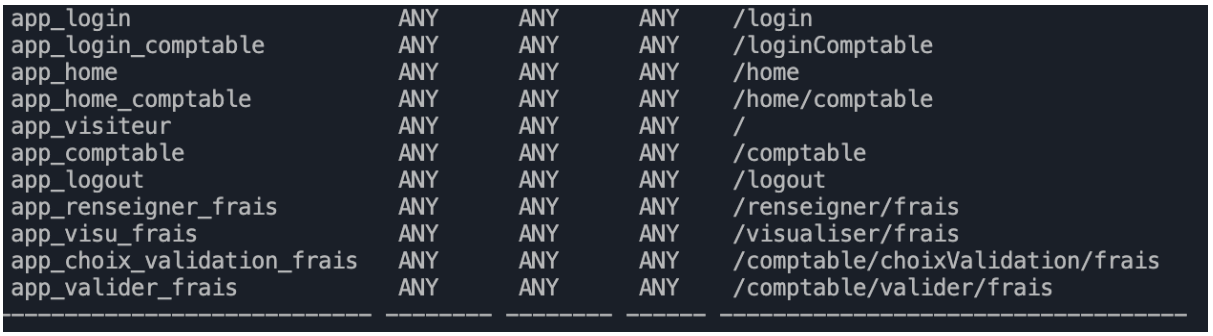
GSB-Frais

Documentation technique

Table des matières

1 - Routes de l'application

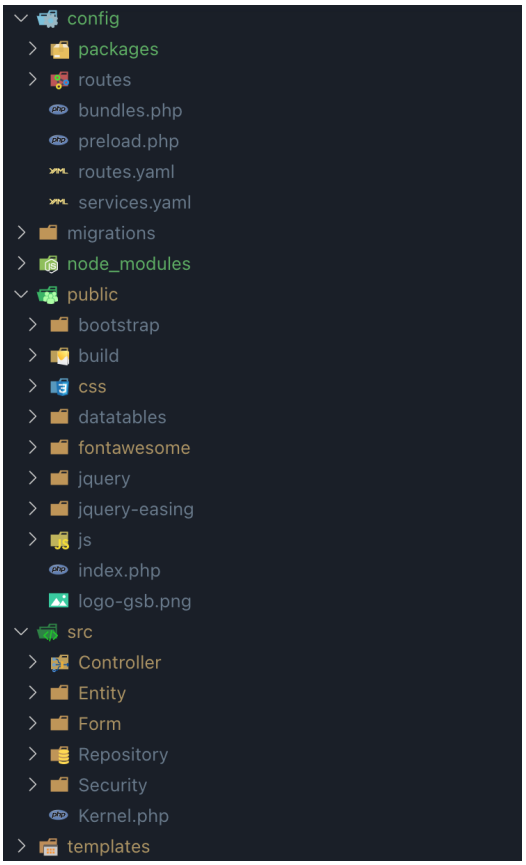


```
app_login                    ANY      ANY      ANY    /login
app_login_comptable          ANY      ANY      ANY    /loginComptable
app_home                     ANY      ANY      ANY    /home
app_home_comptable           ANY      ANY      ANY    /home/comptable
app_visiteur                 ANY      ANY      ANY    /
app_comptable                ANY      ANY      ANY    /comptable
app_logout                   ANY      ANY      ANY    /logout
app_renseigner_frais         ANY      ANY      ANY    /renseigner/frais
app_visu_frais               ANY      ANY      ANY    /visualiser/frais
app_choix_validation_frais   ANY      ANY      ANY    /comptable/choixValidation/frais
app_valider_frais            ANY      ANY      ANY    /comptable/valider/frais
-------------------------- -------- -------- ------ ----------------------------------
```

app_login = SecurityController (contrôleur de connexion)
app_login_comptable = SecurityController (contrôleur de connexion)
app_home = page d'accueil Visiteur
app_home_comptable = page d'accueil Comptable
app_visiteur = page de connexion Visiteur
app_comptable = page de connexion Comptable
app_logout = déconnexion
app_renseigner_frais = formulaire de création ficheFrais, fraisHorsForfait, ligneFraisForfait
app_visu_frais = page de visualisation des fichesFrais etc…
app_choix_validation_frais = page de choix d'une ficheFrais pour validation
app_valider_frais = page de validation d'une ficheFrais

En lançant l'application, nous sommes dirigés directement vers la page **app_visiteur** ou **app_comptable**. La vérification du matricule et mdp se fait auprès du controller de connexion **app_login** ou **app_login_comptable.** Après connexion, nous arrivons vers la page d'accueil **app_home** ou **app_home_comptable**. Nous pouvons ensuite choisir d'ajouter (**app_renseigner_frais)** ou de visualiser (**app_visualiser_frais),** le comptable peut lui, valider les fiches de frais en passant par la vue **app_choix_validation_frais** puis **app_valider_frais.** La déconnexion se fait grâce à la fonction contenue dans **app_logout** qui redirige vers la page de connexion **app_login**.

2 - Architecture globale



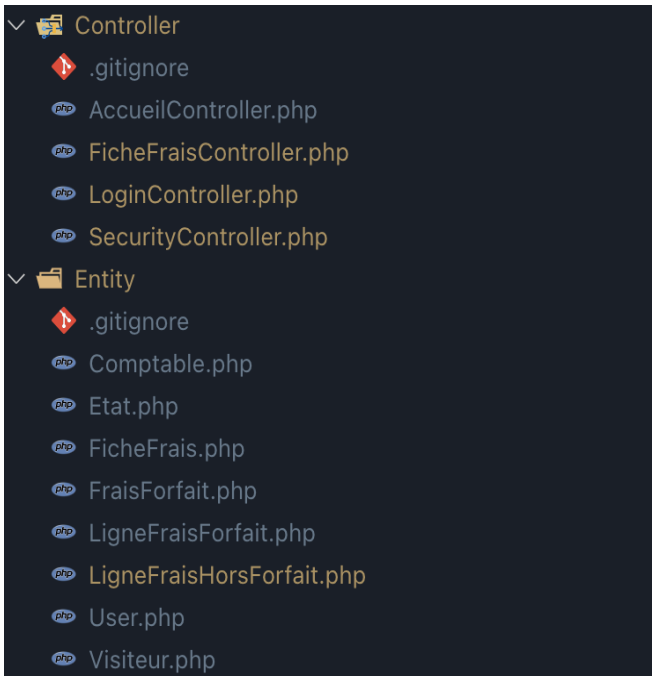
config -> routes + packages

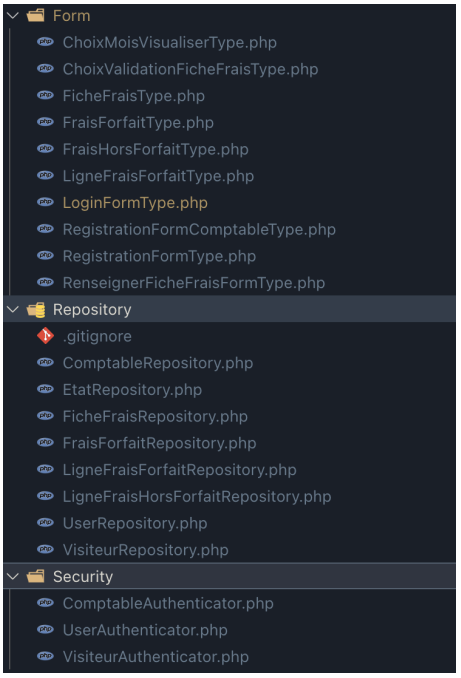
migrations -> version migration BDD

public -> css, bootstrap…

src -> Controller, Form, Entity, Repository

templates -> vues twig





Dossier **templates** contenant les vues de l'application.

3 - Technologies

3.1 - Technologies utilisées

Base de données ⇒ MariaDB-10.11.2

Relation Entités - Base de données ⇒ ORM Doctrine

PHP 8.2.3

Framework PHP ⇒ Symfony 6

Mise en page + stylisation CSS ⇒ bootstrap5 + template sb-admin-2

Icons ⇒ FontAwesome

3.2 - Fonctions utilisées

3.2.1 - Bash

php bin/console **make:form** → créer un formulaire
php bin/console **make:entity** → créer une entité
php bin/console **make:controller** → créer un controller
php bin/console **debug:router** → affiche les routes
php bin/console **make:user** → créer un nouveau User
php bin/console **make:migration** → créer une version de migration
php bin/console **doctrine:migrations:migrate** → effectue la migration
composer update → met à jour les dépendances Symfony

3.2.1 - Controllers (src/Controller)

FONCTIONS

Création d'un formulaire

```
$form→createForm(nomForm::class);
$form→handleRequest($request));
```

Obtenir les données d'un formulaire

```
$form→get['nomChamp']→getData();
$form→getData();
```

Envoie des données après validation

```
$entityManager→persist($nomEntité);
$entityManager→flush();
```

Obtenir les données d'une fonction précédente

```
$var = $request→get('nomObjet');
```

Obtenir les données d'une entité

```
$var = $repository→findOneBy([paramètres]);
$var = $repositor→findBy([paramètres]);
$var = $repository→nomFonction([paramètres]);
```

Si le formulaire est validé et soumis

```
if (isset($_POST[‘submit’]))
==
if ($form->isSubmitted && $form->isValid())
```

Rediriger vers une route après un formulaire

```
return $this→redirectToRoute(‘nomRoute’, [paramètres]);
```

Rediriger vers une page twig

```
return $this→render(‘chemin/fichier.html.twig’, [paramètres]);
```

3.2.2 - Formulaires (src/Form)

Les différents types d’input :

IntegerType::class → type int

TextType::class → type string

DateType::class → type date (jj-mm-aaaa)

PasswordType::class → type mot de passe (*****)

CheckboxType::class → type checkbox (case à cocher)

EntityType::class → type Entité (affiche tous les enregistrements)
→ ‘class’ ⇒ *Entité*::class

ChoiceType::class → type choix (liste avec différents choix)
→ ‘choices’ ⇒ ‘*choix1*’ ⇒ *val1*,
‘*choix2*’ ⇒ *val2*…

Les différentes configurations :

Ajoute un message si le champ n'est pas renseigné.

```
'constraints' ⇒ [
      new NotBlank([
            'message' ⇒ 'msg',
      ]),
],
```

Créer un label.

```
'label' ⇒ 'label',
```

Ajoute une classe **CSS** au champ.

```
'attr' ⇒ [
      'class' ⇒ 'class',
],
```

Ajoute une classe **CSS** au libellé.

```
'label_attr' ⇒ [
      'class' ⇒ 'class',
],
```

3.2.3 - Repository (src/Repository)

```
public function nomFonction(paramètres) {
      →select('champs/table')
      →where('conditions' / 'idVisiteur = ?1')
      →setParameter(numéro, $var / 1, $id)
      →getQuery()
      →getOneOrNullResult();
}
```

3.2.4 - Templates (templates)

{% extends base.html.twig %} ⇒ permet d'étendre la vue (base.html.twig) en gardant ses styles etc…
exemple : navbar statique sur toute l'application

{% block stylesheets %}
{{ parent() }} ⇒ permet d'obtenir les styles (fichier css) du parent.

{% block javascripts %}
{{ parent() }} ⇒ permet d'obtenir les scripts (fichier js) du parent.

{{ form_start(*form*) }}
{{ form_end(*form*) }} ⇒ balises début/fin d'un formulaire (src/Form)

{{ form_row(*form_champ*) }} ⇒ affiche le champ + libellé

href="{{ path('*nomRoute*') }}" ⇒ permet de rediriger directement vers une route (définit dans config/routes).

{{ app.user.login }} ⇒ permet d'afficher le login de l'utilisateur connecté.