Installation

1. Récupérer les fichiers distribués sur le réseau.
Installer vos fichiers dans :

```
I:\UwAmp-www
        |-🗀 alphacorp /
            ├ composer.json
            ├🗀 vendor /
            ├🗀 public /
            |   ├ index.php
            |   ├.htaccess
            |   └🗀 assets /
            |         ├🗀 css /
            |         ├🗀 js /
            |         └🗀 img /
            └🗀 app /
                |-  boostrap.php
                |-  boostrap-router.php
                |-  routes.php
                |-🗀 Config /
                |-🗀 Views /
                |-🗀 Models /
                |-🗀 Controllers /
                |-🗀 Libs /
                └🗀 Storage /
```

2. Vérifier la présence de votre base de données nommée **s3.devweb.alphacorp** et qui contient 8 tables. Ajouter **un préfixe** pour chacune de vos tables.

- les administrateurs du site
  `admin (id, name, password, createdAt )`
- les classes inscrites
  `classroom ( id, description, teachername, code, createdAt )`
- les enfants des classes
  `children ( id, pseudo, avatar, classroom_id, createdAt )`
- la liste des défis possibles
  `challenge( id, description, code, collection_id, createdAt, updateAt )`
- la liste des défis associés à une classe
  `challenge_classroom(id, classroom id, challenge id, createdAt )`
- la liste des collections qui peuvent être utilisées dans les défis
  `collection(id, name )`
- la liste des images appartenant à une collection
  `collection_image(id, letter, image, collection_id, createdAt )`

Le fichier des routes est le suivant:

```
$route->addRoute('GET', '/', 'HomeController@index');

// interface enseignant
$route->addRoute('GET', '/classe/{code}/config', 'HomeController@config');
$route->addRoute('GET', '/classe/{code}/defi', 'HomeController@defi');

// interface enfant
$route->addRoute('GET', '/jouer/{idchildren}/{idchallenge}', 'HomeController@jouer');

// backoffice


$route->addRoute('GET', '/backoffice', 'BoController@index');

$route->addRoute('GET', '/backoffice/admins', 'AdminController@index');

$route->addRoute('GET', '/backoffice/challenges', 'ChallengeController@index');
$route->addRoute('GET', '/backoffice/challenges/add', 'ChallengeController@add');
$route->addRoute('POST', '/backoffice/challenges/save', 'ChallengeController@save');
$route->addRoute('GET', '/backoffice/challenges/edit/{id:[0-9]+}', 'ChallengeController@edit');
$route->addRoute('POST', '/backoffice/challenges/edit/{id:[0-9]+}', 'ChallengeController@update');
$route->addRoute('GET', '/backoffice/challenges/delete/{id:[0-9]+}', 'ChallengeController@delete');
```

Tester les différentes routes pour s'assurer du bon fonctionnement des services déjà développés.
localhost/alphacorp/
localhost/alphacorp/backoffice
localhost/alphacorp/ backoffice/admins
localhost/alphacorp/backoffice/challenges

localhost/alphacorp/backoffice/challenges/add

Modèles de base de données

Plusieurs fichiers modèles sont déjà présents :

Fichier	Méthode spécifique
\Models\Base.php	add($datas) , get($id) , gellAll(), update($id, $data) , delete ($id)
\Models\Admin.php	check($admin, $password) :: Indique si le login existe dans la base.
\Models\Classroom.php	get($code) :: Retourne les informations d'une classe à partir de son code. getIdFromCode($code) :: Retourne l'identifiant d'une classe à partir de son code. exists($code) :: Indique si le code de classe existe. addChallenge($challenge_id, $classroom_id) :: Ajoute un défi pour une classe. deleteChallenges($classroom_id) :: Effacer tous les défis pour une classe. getChildrens($code) :: Retourne la liste des enfants d'une classe.

1. Ajouter une route **/test** qui permet de tester le bon fonctionnement des **6 méthodes** du modèle Classroom

cette route doit être associé à une méthode nommée **test()** dans le contrôleur **HomeController.php**

```
public function test( )
{
  r( App\Models\Classroom::getInstance()->get( 'a124a' ) );
  // !!! à compléter !!!!
}
```

2. Ajouter les fichiers modèles associés aux autres tables:

- Collection.php pour la table **collection**
  pas de méthode spécifique
- CollectionImages.php pour la table **collection_image**
  cette classe contient la méthode spécifique

```
public function getFromCollection( $idcollection )
{
      $sql = "SELECT *
              FROM {$this->tableName}
              WHERE collection_id = $idcollection";
      return self::$dbh->query($sql)->fetchAll();
}
```

- Children.php pour la table **children**
  cette classe contient la méthode spécifique

```
/**
 * Retourne les informations sur tous les enfants ainsi que
 * le nom de la classe associé et le pseudo de l'enseignant associé.
 */   public function getAll()
{
      $sql = "!!! à compléter !!!";
      return self::$dbh->query( $sql)->fetchAll();
}
```

- Challenge.php pour la table **chalenge**

cette classe contient la méthode spécifique

```
/**
 * Retourne les informations détaillées sur un défi.
 * @param  integer  $idchallenge   identifiant du défi
 * @return array (id, description, code, collection_id,
                  createdAt, updatedAt, images )
 */
public function getDetailled ( $idchallenge )
{
      $sql = "!!! à compléter !!!";
      return self::$dbh->query( $sql)->fetchAll();
}
```

Processus de login

1. Pour entrer dans le backoffice, il faut passer la phase d'authentification
Ajouter les 3 routes pour ce traitement

```
$route->addRoute('GET', '/login', 'BoController@login');
$route->addRoute('POST', '/login', 'BoController@loginCheck');
$route->addRoute('GET', '/delog', 'BoController@delog');
```

La première route **/login** en méthode GET permettra d'afficher le formulaire de login.
La seconde est celle qui est déclenchée quand le formulaire est validé.
La dernière déclenchera le processus de fermeture de la session.

2. Dans le contrôleur **BoController.php**, ajouter la méthode login() :

```
/**
* Login dans l'application BackOffice.
*/
public function login()
{
      if (isLogin()) {
            redirect('/backoffice');
      } else {
            $this->display(
                  'backoffice/login.html.twig'
            );
      }

}
```

Remarque: la fonction isLogin() (présente dans le fichier \Libs\functions) indique si une session est déjà ouverte. Attendez la question suivante pour mieux comprendre…

3. Ajouter une vue `\Views\backoffice\login.html.twig`. Le formulaire de cette vue doit transmettre à la route **/login** en méthode post les paramètres `name` et `password`

4. Dans le contrôleur **BoController.php**, ajouter la méthode loginCheck() :

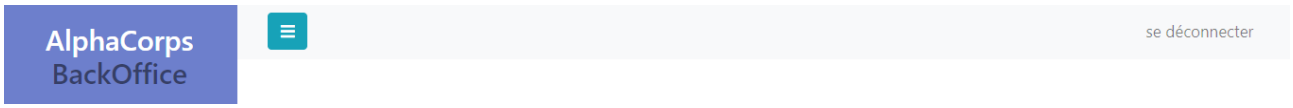
```
/**
* Vérifier le login.
*/
public function loginCheck()
{
  if ( !!! à compléter !!! ) ) {
    // ouvrir une session en mode ADMIN
    !!! à compléter !!!
    $_SESSION[ 'role' ] = 'admin';
    // redirection vers le backoffice
    redirect('/backoffice');
  } else {
    redirect('/login');
  }
}
```

Quand l'authentification est correcte, il faut ouvrir une session.
Pour gérer les variables de session dans l'application, il faut que l'instruction **session_start()** soit déclenchée.
Dans le fichier `bootstrap.php`, ajouter au tout début l'écriture

```
session_start()
```

Dans la fonction `loginCheck()`, ajouter 2 variables de session.
La première nommée **login** passe à true pour indiquer que la session est ouverte.
La seconde nommée **role** prend la valeur 'admin' pour indiquer le rôle de l'internaute qui vient de se connecter.
Remarque: plus tard, quand l'enseignant travaillera, il ouvrira aussi une session mais en mode enseignant (donc avec un role 'ens').

5. Dans la vue `\Views\abckoffice\template.html.twig`, ajouter le HTML qui permet de placer un lien; vous devez produire le visuel suivant :



Le lien est associé à la route `\delog` (déjà mise en place) qui lance la méthode **delog()** de `BoController`.

6. Un internaute qui ne s'est pas authentifié peut être tenté d'utiliser la route : `localhost/alphacorp/backoffice`

Pour le moment, il verrait s'afficher la page d'accueil du backOffice … bonjour la faille de sécurité! Pour gérer cela, il faut ajouter dans la méthode `BoController@index` un test qui s'assure que les pages ne sont affichées que pour un internaute déjà authentifié; dans le cas défavorable, l'internaute doit être redirigé vers la route de login.