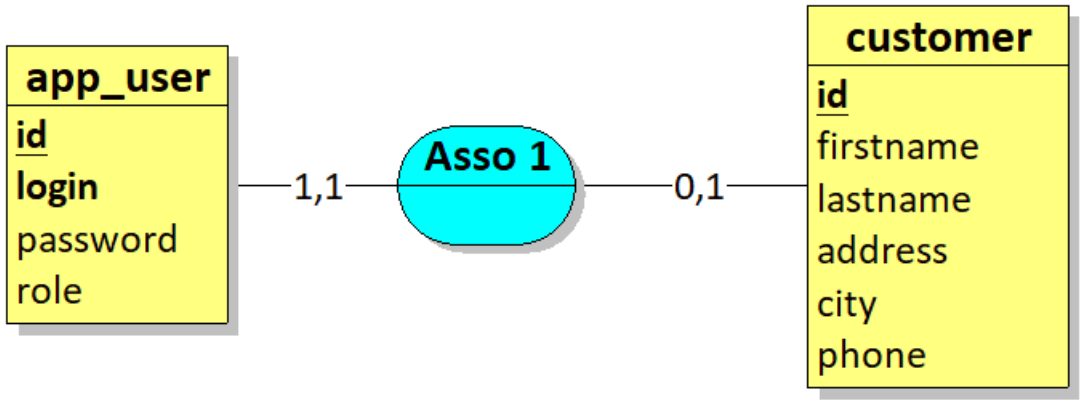
TD PHP MVC

Étape 17 : Préparation de l'authentification et de la création de compte

Nous allons dans cette étape mettre en place ce qui est nécessaire à la création de comptes utilisateurs et à l'authentification.

Nous allons créer 2 tables supplémentaires en base de données : app_user et customer. Voici le MCD.



Un client n'a pas nécessairement de compte utilisateur, il peut l'avoir supprimé.
Voici le script SQL de création des 2 tables :

```sql
DROP TABLE IF EXISTS `customer`;
CREATE TABLE IF NOT EXISTS `customer` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `firstname` varchar(255) COLLATE utf8mb4_general_ci DEFAULT NULL,
  `lastname` varchar(255) COLLATE utf8mb4_general_ci DEFAULT NULL,
  `address` text COLLATE utf8mb4_general_ci,
  `city` varchar(255) COLLATE utf8mb4_general_ci DEFAULT NULL,
  `phone` varchar(255) COLLATE utf8mb4_general_ci DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

DROP TABLE IF EXISTS `app_user`;
CREATE TABLE IF NOT EXISTS `app_user` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `login` varchar(255) COLLATE utf8mb4_general_ci NOT NULL,
  `password` varchar(255) COLLATE utf8mb4_general_ci NOT NULL,
  `role` tinyint(4) DEFAULT NULL,
  `customer_id` int(11) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `login` (`login`),
  UNIQUE KEY `customer_id` (`customer_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

Nous ajoutons une contrainte sur la table app_user vers la table customer :

```sql
ALTER TABLE `app_user` ADD CONSTRAINT `app_user_ibfk_1` FOREIGN KEY (`customer_id`) REFERENCES `customer` (`id`);
```

Créons une ligne dans chaque table afin de tester par la suite :

```
INSERT INTO `customer` (`id`, `firstname`, `lastname`, `address`, `city`, `phone`) VALUES
(1, 'Laurent', 'Bédu', '20 rue du Luxembourg', 'Roubaix', '0123456789');

INSERT INTO `app_user` (`id`, `login`, `password`, `role`, `customer_id`) VALUES
(1, 'admin@admin.fr', 'admin', 1, 1);
```

Nous allons ensuite créer les entités correspondantes avec la relation entre elles.

entity > app_user.entity.php > App_user

```php
<?php

class App_user extends Model{

    static $relations =
    ["Customer" => ['type'=>'hasOne',
                    'table'=>'customer',
                    'attribute'=>'customer',
                    'foreignKey'=>'customer_id']
    ];

}
```

entity > customer.entity.php > Customer

```php
<?php

class Customer extends Model{

    static $relations =
    ["User" => ['type'=>'isOne',
                    'table'=>'app_user',
                    'attribute'=>'user',
                    'foreignKey'=>'customer_id']
    ];

}
```

Nous avons ici un nouveau type de relation OneToOne. Le code que nous avons écrit dans le MainRepository (getOne, getAll) pour la relation hasOne fonctionne pour l'entité App_user (en effet une clé étrangère vers customer se trouve dans app_user)
Par contre nous allons devoir ajouter du code pour faire la relation dans l'autre sens : isOne. Ce code ressemble à hasMany sauf que nous ne récupérerons pas un tableau mais une seule entité, si elle existe, sinon null.
Nous modifions donc nos méthodes getOne et getAll dans le MainRepository

repository > main.repository.php > MainRepository > getAll

```php
        }
        if($relation['type'] == 'isOne'){
            $repo = new MainRepository($relation['table']);
            $results = $repo->getAll();
            foreach($rows as $row){
                $currentResults = array_filter($results, function($item) use ($row, $relation){
                    return $item->{$relation['foreignKey']} == $row->id;
                });
                $row->{$relation['attribute']} =
                    count($currentResults) == 1 ? array_shift($currentResults) : null;
            }
        }
    }
    return $rows;
}
```

repository > main.repository.php > MainRepository > getOne

```php
        }
        if($relation['type'] == 'isOne'){
            $repo = new MainRepository($relation['table']);
            $results = $repo->getAll($relation['foreignKey']." = $row->id");
            $row->{$relation['attribute']} =
                count($results) == 1 ? array_shift($results) : null;
        }
    }

    return $row;
}
```

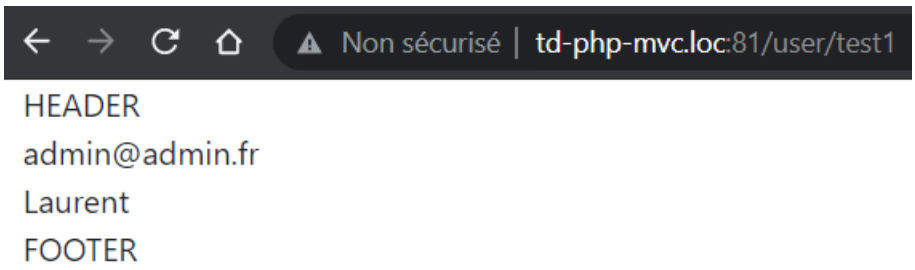
Nous allons tester ces modifications avant de continuer …
Nous créons donc un controller UserController avec une première méthode test1 ainsi que le template user/test1.view.php puis nous testons la route /user/test1 (nous parton du user pour récuperer le customer)

controller > user.controller.php > UserController

```php
<?php

class UserController extends BaseController
{
    public function test1()
    {
        $repository = new MainRepository("app_user");
        $repository->with("Customer");
        $user = $repository->getOne(1);

        $this->entities = ['user' => $user];

        $this->render();
    }
```

template > user > test1.view.php

```php
<?= $user->login ?><br>
<?= $user->customer->firstname ?>
```



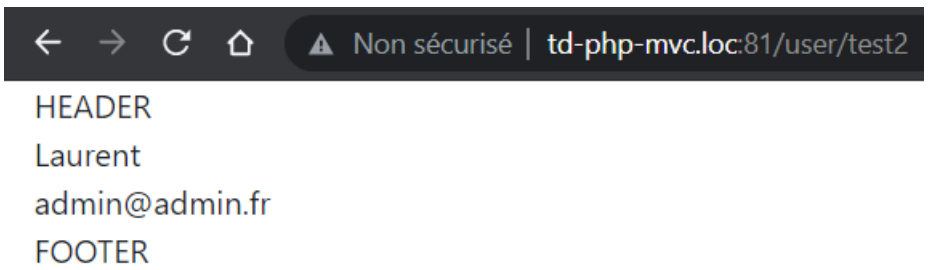
Testons la relation dans l'autre sens avec test2 (du customer vers le user)

controller > user.controller.php > UserController

```php
public function test2()
{
    $repository = new MainRepository("customer");
    $repository->with("User");
    $customer = $repository->getOne(1);

    $this->entities = ['customer' => $customer];

    $this->render();
}
```

template > user > test2.view.php

```php
<?= $customer->firstname ?><br>
<?= $customer->user->login ?>
```



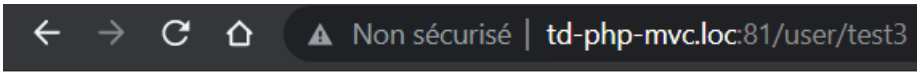
Avec test3 et test4 nous testons la relation dans les 2 sens pour le getAll cette fois ci.

controller > user.controller.php > UserController

```php
26
27      public function test3()
28      {
29          $repository = new MainRepository("app_user");
30          $repository->with("Customer");
31          $users = $repository->getAll();
32
33          $this->entities = ['users' => $users];
34
35          $this->render();
36      }
```

template > user > test3.view.php

```php
1   <?php foreach($users as $user){ ?>
2       <?= $user->login ?><br>
3       <?= $user->customer->firstname ?>
4   <?php } ?>
```



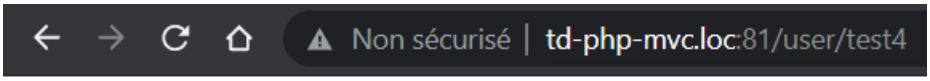
HEADER
admin@admin.fr
Laurent
FOOTER

controller > user.controller.php > UserController

```php
38      public function test4()
39      {
40          $repository = new MainRepository("customer");
41          $repository->with("User");
42          $customers = $repository->getAll();
43
44          $this->entities = ['customers' => $customers];
45
46          $this->render();
47      }
48  }
```

template > user > test4.view.php

```php
<?php foreach($customers as $customer){ ?>
    <?= $customer->firstname ?><br>
    <?= $customer->user->login ?>
<?php } ?>
```



HEADER
Laurent
admin@admin.fr
FOOTER