

compte rendu TP

4- create des entités

php bin/console make:entity restaurant

```
New property name (press <return> to stop adding fields):
> name

Field type (enter ? to see all types) [string]:
> string

Field length [255]:
> 50

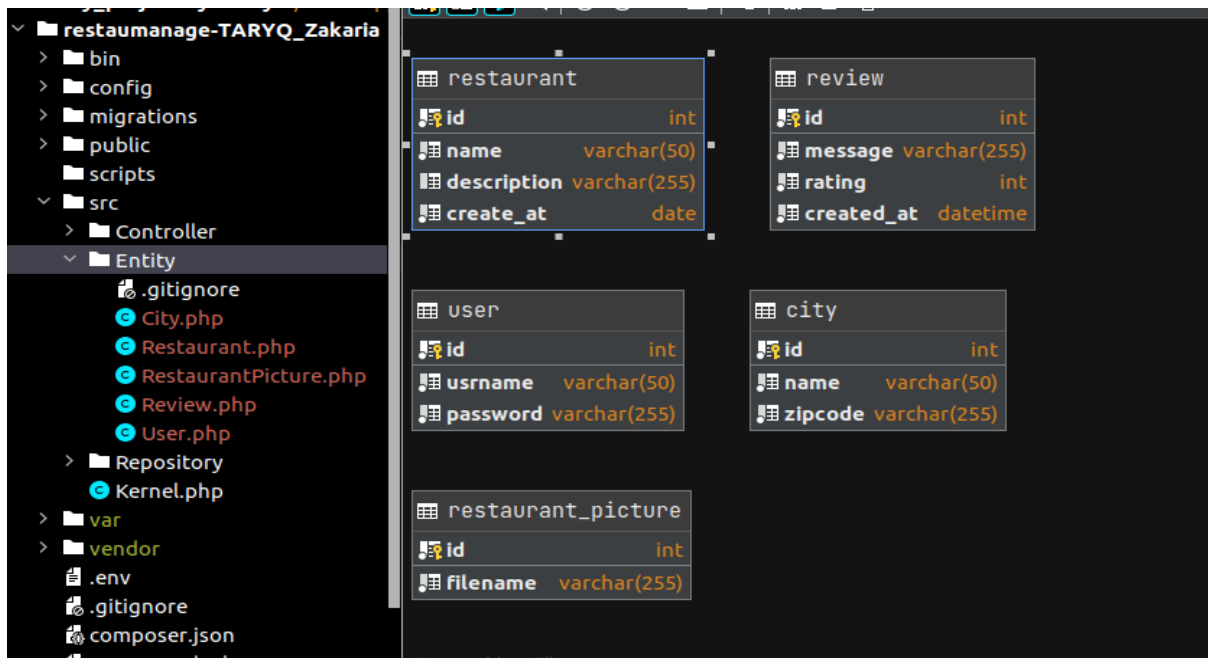
Can this field be null in the database (nullable) (yes/no) [no]:
> no

updated: src/Entity/Restaurant.php

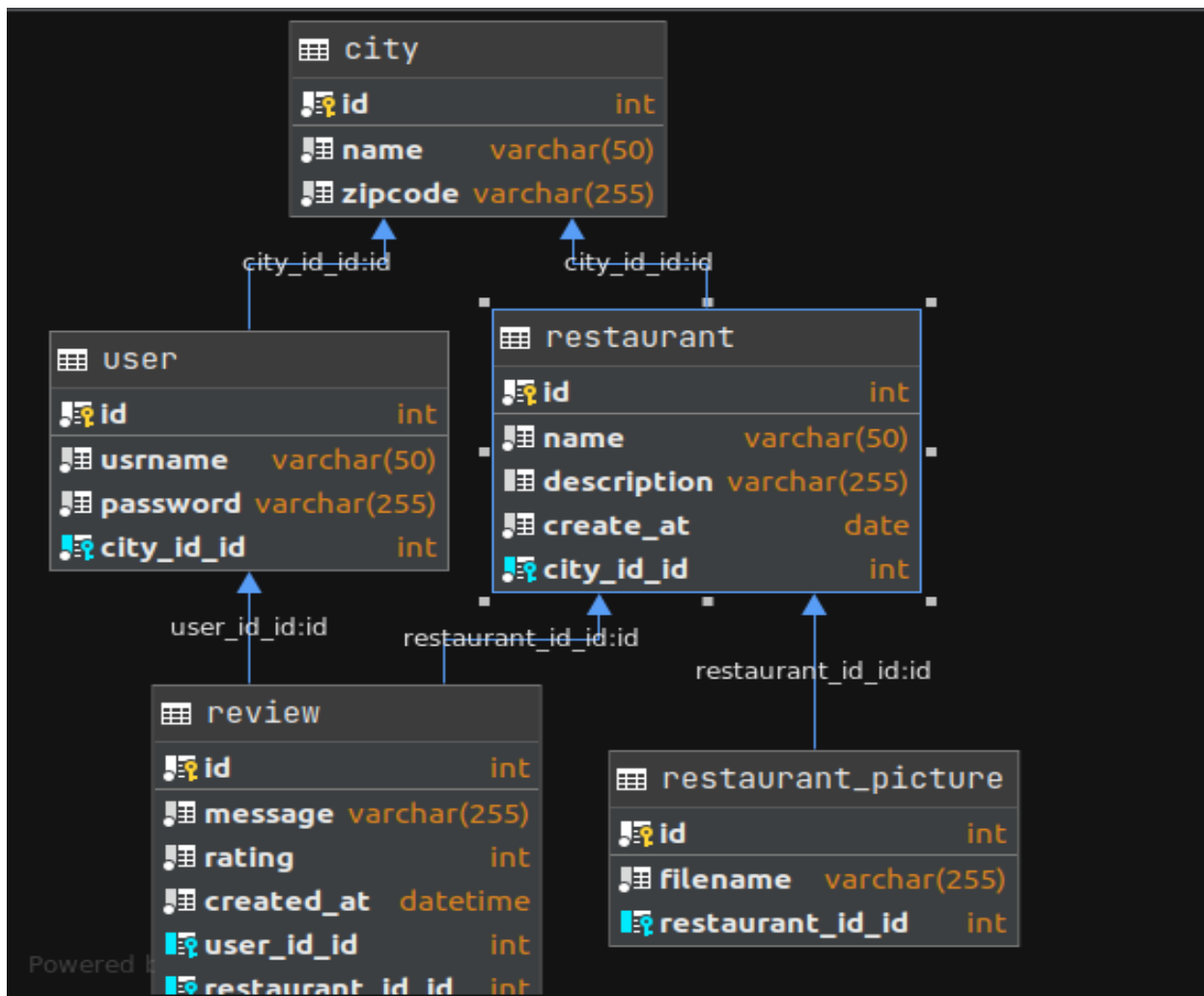
Add another property? Enter the property name (or press <return> to stop adding fields):
> description

Field type (enter ? to see all types) [string]:
> string

Field length [255]:
> 255
```



5-creation des relations



6-Fournir au champ *createdAt* une valeur par défaut .

```
/**  
 * @ORM\Column(name="create_at", type="datetime", options={"default":  
"CURRENT_TIMESTAMP"})  
 */
```

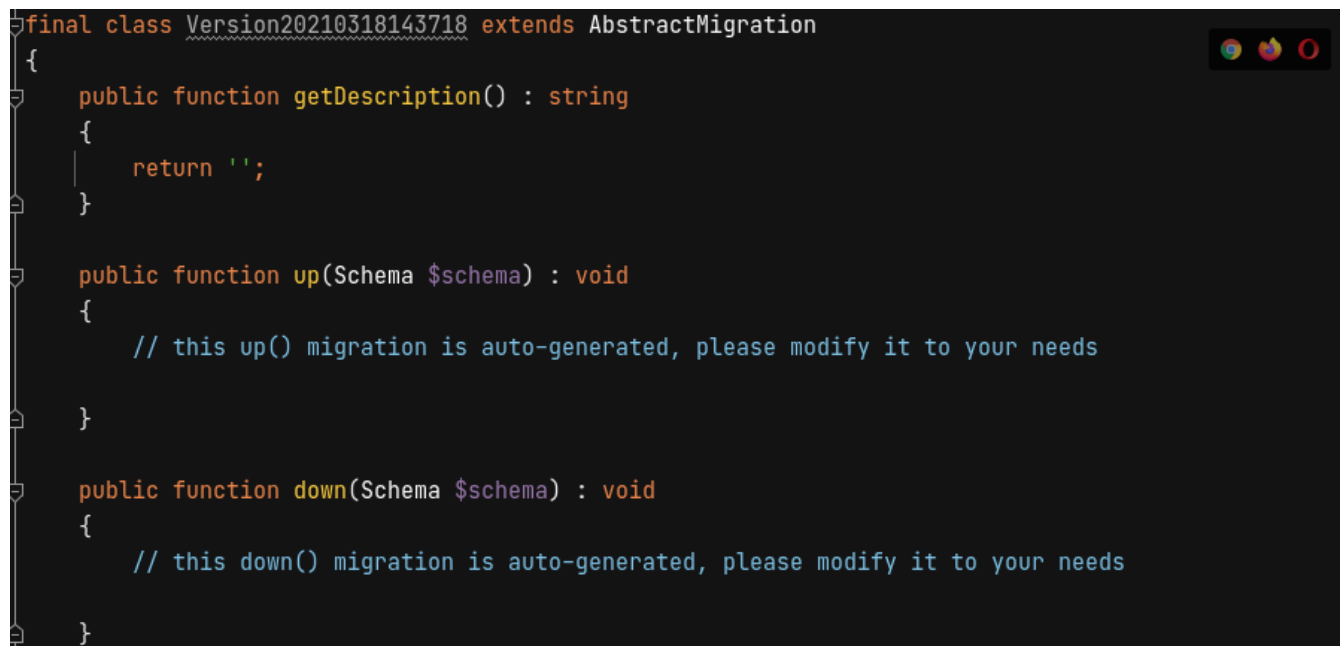
7- Créer la migration et l'exécuter, décris moi l'utilité de chaque commande et rejoins en pièce jointe la capture d'écran de la base de données après le lancement des migrations.

doctrine:migrations:execute

doctrine:migrations:migrate

doctrine:migrations:status

doctrine:migrations:generate

A screenshot of a code editor with a dark background. The code is in PHP and defines a class `Version20210318143718` that extends `AbstractMigration`. The class has three methods: `getDescription()` which returns an empty string, `up()` which has a comment indicating it is auto-generated, and `down()` which also has a comment indicating it is auto-generated. The code is color-coded: keywords like `final`, `class`, `extends`, `public`, `function`, and `void` are in orange; strings and variables are in light blue; and comments are in a lighter blue. The editor has a sidebar on the left with a file explorer and a top bar with window control icons.

```
final class Version20210318143718 extends AbstractMigration
{
    public function getDescription() : string
    {
        return '';
    }

    public function up(Schema $schema) : void
    {
        // this up() migration is auto-generated, please modify it to your needs
    }

    public function down(Schema $schema) : void
    {
        // this down() migration is auto-generated, please modify it to your needs
    }
}
```

8- Créer le contrôleur de chaque entité et rejoindre la commande de création.

Exemple :

php bin/console make:controller restaurantController

9- Migrations (script SQL) :

php bin/console doctrine:migrations:execute --up 'DoctrineMigrations\Version20210318143718'

10- Implémenter les routes suivantes et leurs méthodes :

voir le code

11- Afficher les 6 derniers restaurants créés

```
$query = $entityManager->createQuery("SELECT r FROM \App\Entity\
Restaurant r order By r.create_at Desc ")
->setMaxResults(6);
```

12- Afficher la valeur moyenne de la note d'un restaurant

```
$query = $entityManager->createQuery("SELECT AVG(rv.rating) as
moyenne FROM App\Entity\Review rv where rv.restaurant_id=".$id);
```

13- Afficher les 3 top meilleurs restaurants

```
$sql = " SELECT r.id ,r.name , r.description , r.create_at ,avg(rv.rating) as
moyenne
FROM restaurant r INNER join review rv where rv.restaurant_id_id=r.id
GROUP BY r.id ORDER By moyenne asc limit 3";
```

15- Lister les restaurants et leurs détails (review, city..)

```
$sql = "SELECT r.id ,r.name , r.description ,  
r.create_at,rv.user_id_id,rv.rating,rv.message,u.usrname  
FROM restaurant r INNER join review rv on rv.restaurant_id_id=r.id  
INNER JOIN user u on rv.user_id_id=u.id where r.id=".$id  
;
```

16- classer les restaurant pas vot

```
$sql = " SELECT r.id ,r.name , r.description , r.create_at ,avg(rv.rating) as  
moyenne  
FROM restaurant r INNER join review rv where rv.restaurant_id_id=r.id  
GROUP BY r.id ORDER By moyenne asc";
```

Des captures d'écran :