(/)

(/users/204) (/users/202) (/users/274) (/users/199) (/users/200) (/users/203) (/users/259)

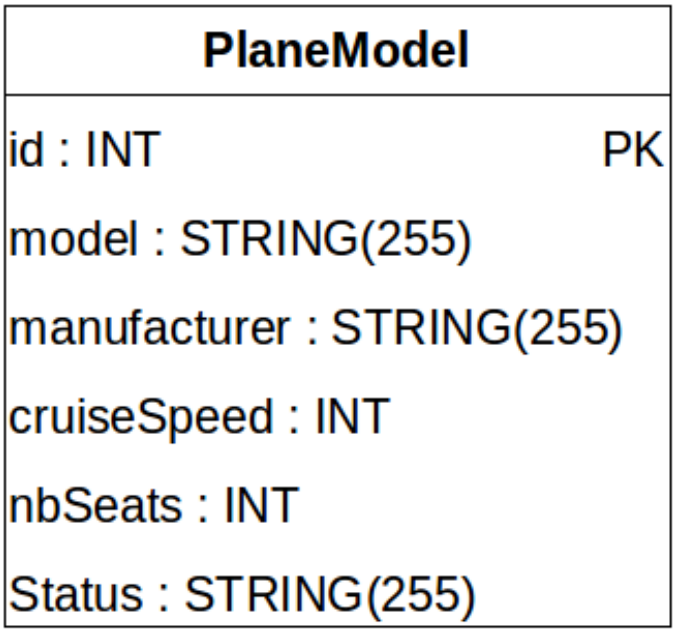
Progression sur la quête **Générer le CRUD**

(/tasks/7172) (/tasks/7173) (/tasks/7174)

33%

Générer le CRUD | Génération de l'entité

Dans le monde de Symfony les modèles du MVC sont appelés entités. Plus exactement les entités sont les modèles que l'on souhaite enregistrer en BDD. C'est le cas des avions que nous allons aussi générer avec la console SF2. Je te propose de générer l'entité PlaneModel avec les champs suivants :

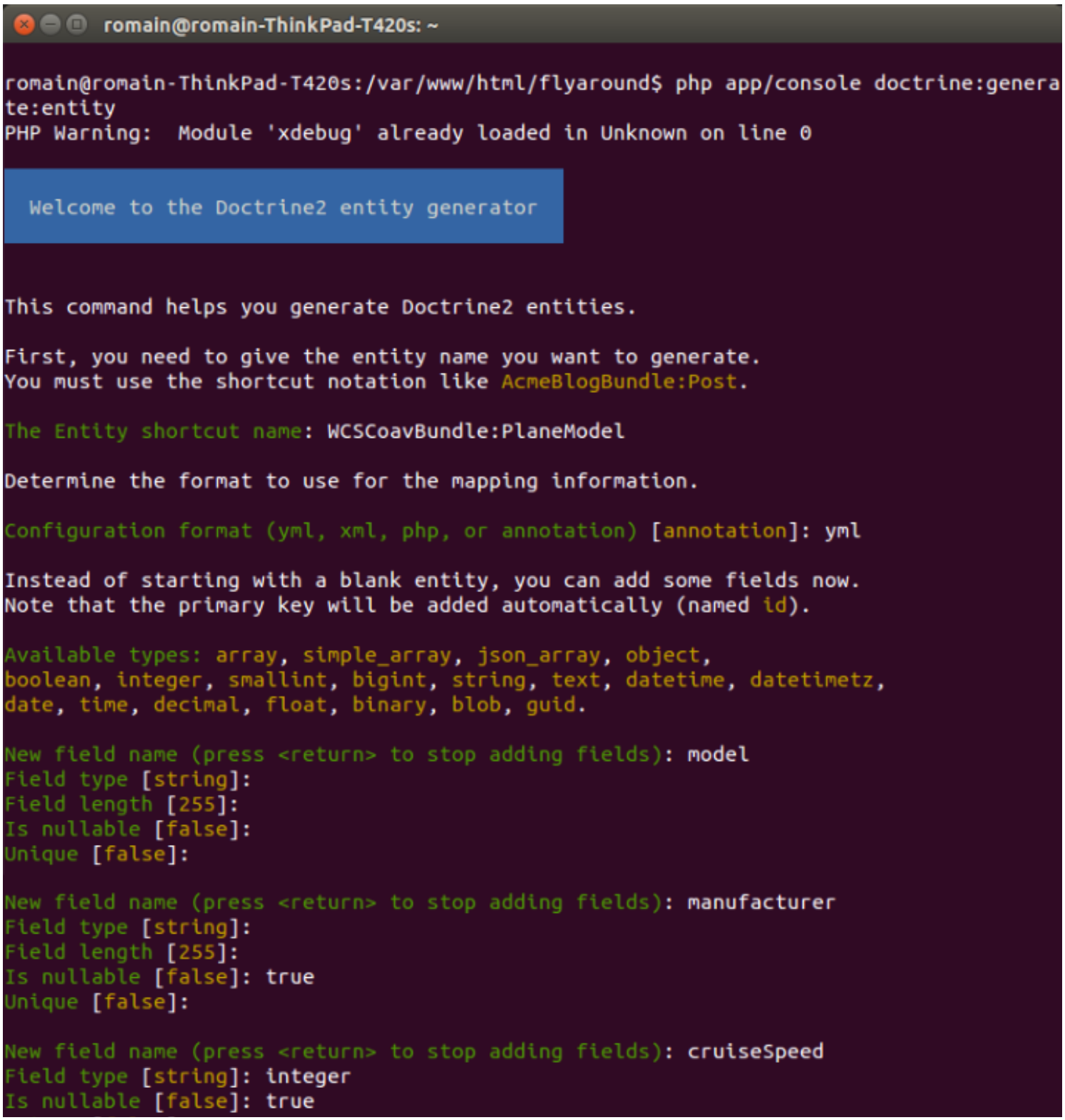


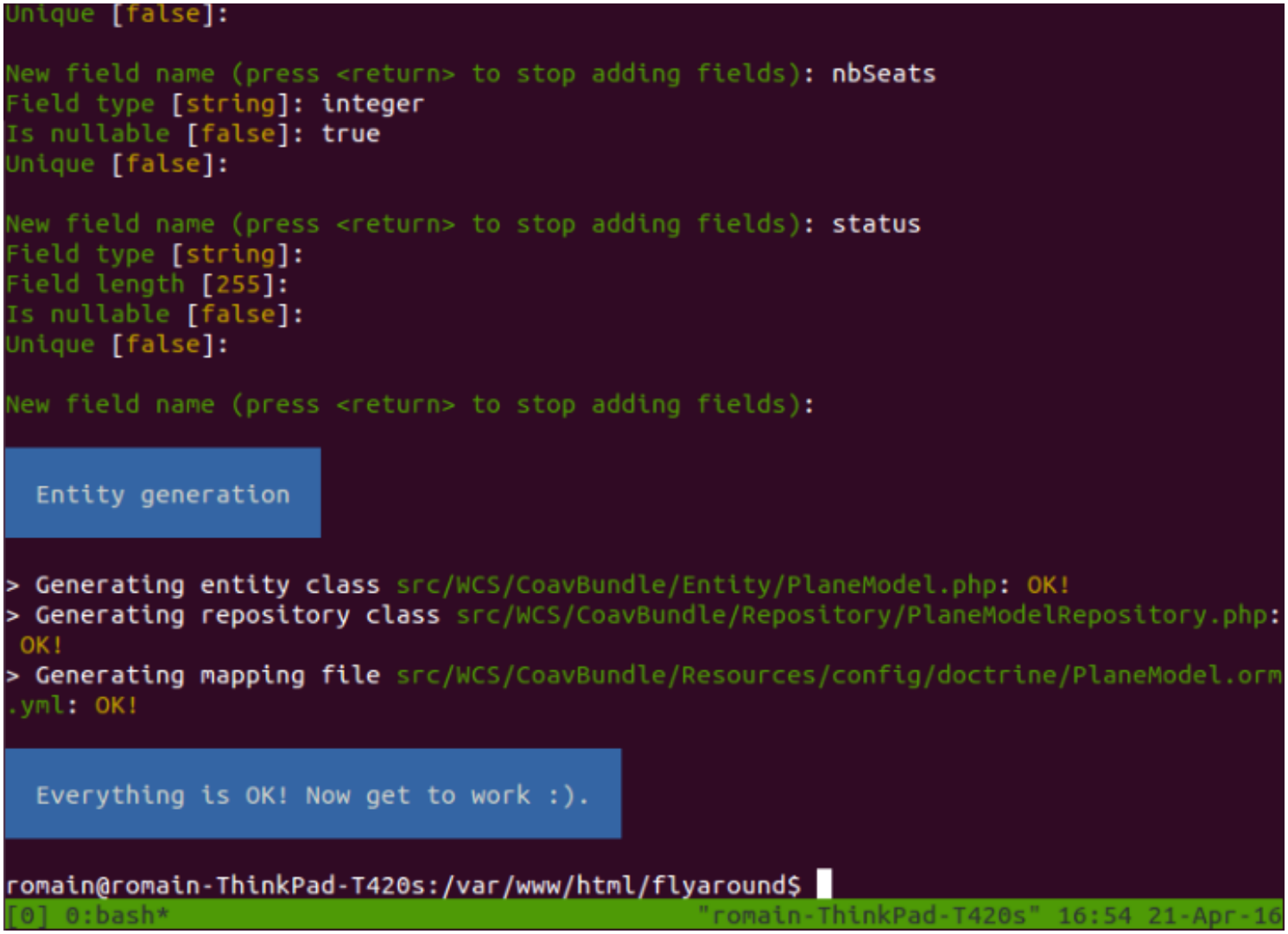
C'est Doctrine qui va faire le travail de générer l'entité :

```
php app/console doctrine:generate:entity
```

élèves connectés

Doctrine commence par te demander de donner un nom à ton entité. Comme chaque entité est enregistrée dans un bundle, on l'indique à Doctrine dans le nommage de l'entité. Pour la configuration je te propose de garder le YAML pour rester cohérent. Ensuite on attaque la saisie des champs. Pas la peine de lui indiquer qu'il y a un attribut id. Doctrine le crée tout seul et en fait une clé primaire. Tu remarqueras que la convention de nommage des attributs est du camelCase. C'est une sorte de CamelCase dans lequel le première lettre du premier mot est une minuscule.





Résultat, trois fichiers ont été générés : * PlaneModel.php : Il contient la classe PlaneModel avec les attributs demandés. Chaque attribut est déclaré en private et vient avec un getter et un setter sauf l'id qui n'a qu'un getter. L'id est géré automatiquement par Doctrine et ne doit pas être saisi à la main. * PlaneModelRepository.php : Le repository sert à créer des requêtes sur l'ensemble de la table PlaneModel. On pourra par exemple y créer une méthode qui nous renvoie tous les avions de n places. La classe est vide pour le moment mais étend EntityRepository qui fournit quelques méthodes très utiles. * PlaneModel.orm.yml : C'est le fichier de config de l'ORM, en l'occurence Doctrine, qui va faire le lien entre l'entité en tant que classe ou objet et la table PlaneModel de la base de donnée.

> Quand on utilise un ORM une entité est d'un côté une classe qui s'intancie en objets et de l'autre une table dans une BDD. Ainsi les attributs de la classe correspondent aux champs de la table. Attention aux confusions.

En créant cette entité nous avons modifié le schéma de la base de données. Il faut donc indiquer à doctrine qu'il doit le mettre à jour. Utilise la commande suivante

```
php app/console doctrine:schema:update --force
```

> Il ne peut pas faire ça tout seul ce traine savate ?

La mise à jour du schéma de BDD est une opération qui peut s'avérer sensible sur une base de données de production. Des données pourraient etre perdues définitivement. C'est aussi la raison pour laquelle doctrine nous impose de préciser –force pour que l'on prenne bien conscience de ce que l'on est en train de faire.

1. Le model de données (http://jobeet.thuau.fr/le-modele-de-donnees)
   Ce tuto propose une autre méthode pour générer les données. Nous l'utiliserons plus tard pour mettre à jour nos entités

2. Les entités avec OpenClassrooms (https://openclassrooms.com/courses/developpez-votre-site-web-avec-le-framework-symfony2/la-couche-metier-les-entites)
   Chez OpenClassrooms on configure l'ORM avec des annotations

---

Retour à la quête (/quests/49)

Valider + Suivante
>
(/tasks/7173/validate?
redirect=%2Ftasks%2F71