

A decorative network diagram in the top-left corner, featuring a complex web of interconnected nodes and lines. Some nodes are highlighted with blue circles, and others with blue dots.

Les mocks

A decorative network diagram in the bottom-right corner, featuring a complex web of interconnected nodes and lines. Some nodes are highlighted with blue circles, and others with blue dots.

PLAN

- 1 - Exercice sur les tests unitaires
- 2 - Le concept de mock
- 3 - Mise en pratique



A decorative network diagram in the top-left corner, consisting of various sized circles (nodes) connected by thin lines (edges). Some nodes are solid grey, while others are hollow with a grey outline. The connections form a complex, branching structure.

2. EXERCICE SUR LES TESTS UNITAIRES

A decorative network diagram in the bottom-right corner, similar to the one in the top-left. It features a cluster of nodes connected by lines, with some nodes being solid grey and others hollow with grey outlines.

EXERCICE : la classe “User”

Implémenter une classe User et sa classe de test

- Détails fonctionnels :

- Un User possède un email, un nom, un prénom, un date de naissance

- Un User est valide si son email est renseigné et a le bon format, ses nom et prénom sont renseignés et si il a 13 ans ou plus

- Opérations :

- isValid() dans la classe User



A decorative network diagram in the top-left corner, consisting of various sized circles (nodes) connected by thin lines (edges). Some nodes are solid grey, while others are hollow with a grey outline. The connections form a complex, branching structure.

2. LE CONCEPT DE MOCK

A decorative network diagram in the bottom-right corner, similar to the one in the top-left. It features a cluster of nodes connected by lines, with some nodes being solid grey and others hollow with grey outlines.

LES MOCKS

- ◎ Objet “magique” au comportement pré-câblé
- ◎ Permet de configurer le comportement de l’objet sous la forme :

*Quand on te demande ça,
tu fais ça !*

LES MOCKS

À quoi ça sert ?

LES AVANTAGES

- ◎ Ne pas s'occuper des dépendances du SUT*
- ◎ Tester des états difficiles à reproduire
- ◎ Tester facilement des cas d'erreurs
- ◎ Tester lorsque des dépendances ne sont pas encore développées
- ◎ S'éviter des initialisations très longues

* System Under Test

A QUOI RESSEMBLE UN MOCK ?

◎ Création du mock :

```
$obj = $this->createMock(SomeClass::class);
```

◎ Utilisation du mock :

```
$obj->expects($this->any())  
    ->method('doSomething')  
    ->will($this->returnValue(42));
```



A decorative network diagram in the top-left corner, featuring a complex web of interconnected nodes and lines. The nodes are represented by circles of varying sizes, some with concentric rings, and the lines are thin and grey. The diagram is partially cut off by the left edge of the frame.

3. MISE EN PRATIQUE

A decorative network diagram in the bottom-right corner, similar to the one in the top-left. It shows a cluster of interconnected nodes and lines, with nodes represented by circles of varying sizes and some having concentric rings. The lines are thin and grey. The diagram is partially cut off by the right and bottom edges of the frame.

PROJET “ToDoList”

- Seul un User valide peut créer une ToDoList

- Un User est valide si :

- Son adresse email est au bon format
- Il a bien un firstname et un lastname
- Son password (non crypté) fait entre 8 et 40 caractères et contient au moins une minuscule, une majuscule et un chiffre
- Il a 13 ans ou plus

- Vous pouvez réutiliser le IsValid() vu précédemment



PROJET “ToDoList”



- Un utilisateur peut n'avoir qu'une seule ToDoList
- Une ToDoList peut contenir de 0 à 10 items
- Un item contient un name (unique), un content (maximum 1000 caractères), une date de création
- Il faut obligatoirement respecter une période de 30 min entre la création de 2 items d'une même liste.
- L'ajout d'un item ne se fait que si l'utilisateur est valide.
- Faire au minimum une fonction `add(Item)` qui ajoute un item dans une `ToDoList`

PROJET “ToDoList”

À l’ajout du 8ème item, envoyer un mail au User (via une classe EmailSenderService) pour lui indiquer que sa ToDoList est presque remplie.



Utilisation obligatoire d’un mock pour ne pas implémenter le vrai envoi de l’email. S’assurer que la méthode est bien appelée avec les bons arguments

PROJET “ToDoList”

Faire une méthode “save” qui sera appelée pour enregistrer un item dans la ToDoList.



*Utilisation obligatoire d'un mock
pour ne pas implémenter
la vraie méthode “save”.*

Elle devra simplement lever une exception !