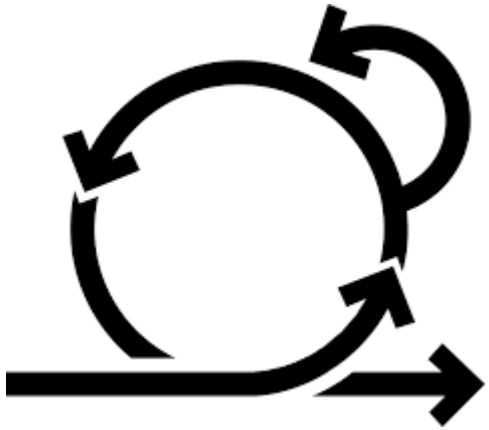
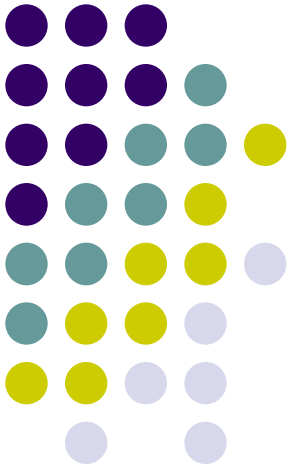
PHP

User Stories

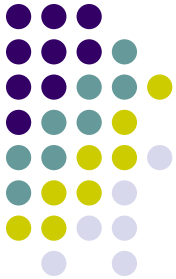




Franck LAMY – 2023/2024 – BTS SIO2

Objectifs

- Comprendre les critères d'acceptation
- Comprendre l'injection de dépendances
- Créer des tests d'intégration



User Story

Implémentation

User Stories

Formalisme

En tant que [type d'utilisateur],

je veux [une action],

afin de [bénéfice/raison].

User Stories

Exemple

En tant que bibliothécaire,

Je veux créer un adhérent,

Afin de gérer son accès et ses emprunts au sein de la bibliothèque

Critères d'acceptation

Les critères d'acceptation sont des conditions spécifiques et des vérifications qui doivent être satisfaites pour que la User Story soit considérée comme complète et fonctionnelle.

User Stories

Critères d'acceptation

Validation des données

- L'email doit être renseigné, valide et unique.
- Le nom et le prénom doivent être renseignés

Génération du numéro d'adhérent

- Un numéro d'adhérent unique doit être généré automatiquement lors de la création de l'adhérent.
- Il doit respecter le format "AD-999999"

Enregistrement en base de données

- Les informations de l'adhérent doivent être correctement enregistrées dans la base de données.

Gestion des erreurs

- Des messages d'erreur explicites doivent être retournés en cas d'informations manquantes ou incorrectes.
- Des messages d'erreur doivent être retournés en cas de problème lors de l'enregistrement en base de données.

User Stories

Implémentation

```php
class CreerAdherent {

    public function execute(PARAMETRES) : RETOUR {
        // Implémentation
    }
}
```

Dépendances

entityManager

Générateur de numéro d'adhérent

User Stories

Implémentation

```
class CreerAdherent {

    public function execute(PARAMETRES) :
RETOUR {
        // Implémentation
    }
}
```

Dépendances

SERVICES

User Stories

Implémentation

INJECTION DE DEPENDANCE

SERVICES

Constructeur

USER STORY

```php
class CreerAdherent {

    // SERVICES : attributs
    public function __construct(SERVICES) {
        // Implémentation
    }

    public function execute(PARAMETRES) :
RETOUR {
        // Implémentation
    }
}
```

Implémentation

A vos codes !

User Stories

Requête

```
class CreerAdherent {

    public function execute(PARAMETRES) :
RETOUR {
        // Implémentation
    }
}
```

Nom:
Prénom:
Email:

string $prenom,
string $nom,
string $email

OBJET

User Stories

Requête

Nom:

Prénom:

Email:

OBJET

```php
class CreerAdherentRequete
{
    public string $prenom;
    public string $nom;
    public string $email;

    public function __construct(string $prenom, string $nom, string $email)
    {
        $this->prenom = $prenom;
        $this->nom = $nom;
        $this->email = $email;
    }
}
```

Implémentation

A vos codes !

Validation

Valider les données utilisateur

Validation des données

- L'email doit être renseigné, valide et unique.
- Le nom et le prénom doivent être renseignés

```
class CreerAdherentRequete
{
    public string $prenom;
    public string $nom;
    public string $email;

    …
}
```

User Stories

Validation

Comment
Faire
?

- **Tester chaque donnée**
- **Créer un validateur de données (classe) et l'injecter en tant que service**
- **Utiliser un validateur "prêt à l'emploi" et l'injecter en tant que service**

Validation

Utiliser un validateur "prêt à l'emploi" et l'injecter en tant que service

Le composant Validation de Symfony


```
composer require symfony/validator
```

User Stories

Validation

Valider les données utilisateur

Nom:
Prénom:
Email:

```
class
CreerAdherentRequete
{
    public string
$prenom;
    public string $nom;
    public string
$email;
    ...
```

Contraintes de validation

Attributs

#[...]

User Stories

Validation

Validation des données

- L'email doit être renseigné, valide et unique.
- Le nom et le prénom doivent être renseignés

```php
use Symfony\Component\Validator\Constraints as
Assert;

class CreerAdherentRequete
{
    #[Assert\NotBlank(
        message: "Le prénom est obligatoire"
    )]
    public string $prenom;

    #[Assert\NotBlank(
        message: "Le nom est obligatoire"
    )]
    public string $nom;

    #[Assert\Email(
        message: "L'email {{ value }} n'est pas
valide"
    )]
    #[Assert\NotBlank(
        message: "L'email est obligatoire"
    )]
    public string $email;
}
```

Ces contraintes doivent être validées dans la userStory

User Stories

Validation

- Injecter un **ValidatorInterface** dans la **userStory**
- Valider les données utilisateur : méthode **validate**(...)
- Lancer une **exception** s'il y a des erreurs : certaines contraintes ne sont pas valides !

User Story

Tests

User Stories

Tests


```php
class CreerAdherent {

    public function execute(PARAMETRES) : RETOUR {
        // Implémentation
    }
}
```

Dépendances

- **entityManager**

- **Générateur de numéro d'adhérent**
- **validateur**

User Stories

Tests

Tests unitaires ?

```
class CreerAdherent {

    public function execute(PARAMETRES) : RETOUR {
        // Implémentation
    }
}
```

Dépendances

MOCK → entityManager

MOCK → Générateur de numéro d'adhérent

MOCK → validateur

User Stories

Tests

Tests unitaires ?

MOCK **entityManager**

MOCK **Générateur de numéro d'adhérent**

MOCK **validateur**

Pas "judicieux" !

Perte de temps !

Tests

Tests unitaires ?

Dépendances

entityManager

Générateur de numéro d'adhérent

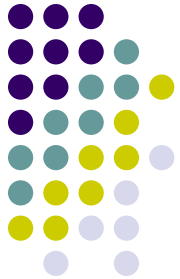
validateur

User Stories

Tests

Mettre en place des tests d'intégration

Vérifier que différentes parties du code fonctionnent ensemble correctement.



Tests

Tests d'intégration

```
class CreerAdherent {

    public function execute(PARAMETRES) : RETOUR {
        // Implémentation
    }
}
```

Dépendances

entityManager

Générateur de numéro d'adhérent

validateur

Tests

Tests d'intégration

entityManager

- Mettre en place une base de données de test (différente de la base de données de développement)
- Nouvelle configuration de tests pour l'entityManager

Tests

Tests d'intégration

```php
class CreerAdherent
{

    public function execute(CreerAdherentRequete $requete) :
bool {

        // Valider les données en entrées (de la requête)

        // Vérifier que l'email n'existe pas déjà

        // Générer un numéro d'adhérent au format AD-999999

        // Vérifier que le numéro n'existe pas déjà

        // Créer l'adhérent

        // Enregistrer l'adhérent en base de données

        return true;
    }
}
```

La BD de test doit être recréée pour chaque test

Tests

La BD de test doit être recréée pour chaque test

Tests d'intégration

Chaque test d'intégration devrait être indépendant des autres pour assurer qu'ils évaluent correctement la fonctionnalité qu'ils sont censés tester.

Cette indépendance permet aussi de s'assurer que chaque test est répétable et fiable, c'est-à-dire qu'il produira le même résultat sous les mêmes conditions, indépendamment de l'ordre d'exécution des tests ou du résultat d'autres tests.

Implémentation

A vos codes !