Devoir Maison - Semaine 4

Les tests

Lectures recommandées :

- *The Practical Test Pyramid*, Martin Fowler
- *Technical debt quadrant*, Martin Fowler
- *Mock Aren't Stubs*, Martin Fowler
- Podcast « Artisan développeur »

Mise en situation n°1

Vous êtes récemment embauché comme tech lead sur un produit qui est en retard par rapport au planning durement négocié il y a 6 mois. Une démonstration du produit prévue dans 4 semaines aura un impact sur l'avenir du produit en question. Que répondez-vous à votre sponsor qui vous propose de réduire l'effort consacré aux tests afin de sécuriser cette échéance ?

Mise en situation n°2

Vous êtes récemment embauché comme tech lead sur un application avec de vieilles stacks techniques. Votre directeur technique vous indique que l'équipe n'arrive plus à développer et mettre en production l'application. Il vous indique que vous avez carte blanche pour améliorer la situation, que proposez-vous ?

Comment définissez-vous les catégories de tests suivantes ?

- tests unitaires (unit tests)
- tests d'intégration (integration tests)
- tests d'interface utilisateur (User Interface tests)
- tests de recette (acceptance tests)
- tests manuels (manual tests)
- tests de bout en bout (end-to-end tests)
- tests de performance (benchmarks)

Décrire succinctement les patterns de tests suivants : mock, double, stub, spy, dummy, fake

Le Test-Driven Development

Lectures recommandées :

- *Test-Driven Development: by Example*, Kent Beck

Est-ce une méthode de test ou une méthode de développement ? Expliquer succinctement.

Quelle est la différence entre le TDD « outside-in » et le TDD « inside-out » ?

L’injection de dépendance

Lectures recommandées :

- *Inversion of Control Containers and the Dependency Injection pattern*, Martin Fowler
- *DIP In The Wild*, Brett L. Schuchert

A quoi ça sert l’injection de dépendance ? Quels sont les inconvénients ?

Quelle est la différence entre l'injection manuelle et l'injection automatique ?

Quelle limite voyez-vous dans l’utilisation d’un framework d'injection automatique ?

Parmi les deux exemples de code suivant, lequel permet d’écrire un test sans injection de dépendance ? Quels sont les avantages et inconvénients de chacun des exemples ?

Exemple n°1 :

```
class Dependency {
  call() {
    return "original";
  }
}

class SystemUnderTest {
  constructor(private dependency: Dependency){}
  run() {
    return this.dependency.call();
  }
}

const stub = sinon.stub(Dependency);
let systemUnderTest = new SystemUnderTest(stub);
stub.call.returns("stubbed")
console.log(systemUnderTest.run());
```

Exemple n°2 :

```
class Dependency {
  call() {
    return "original";
  }
}
const dependency = new Dependency();

class SystemUnderTest {
  run() {
    return dependency.call();
  }
}
```

```
const systemUnderTest = new SystemUnderTest();
sinon.stub(dependency, "call").returns("another stub");
console.log(systemUnderTest.run());
```