



Université de Paris Dauphine - Paris

Spécialité : MIAGE SITN M2

Thème

Compte Rendu de la fusion du module Agilité

Réalisé par

-Mr. NADJEM Nadir

-Mr. MASDOUA Manil

Introduction:

Présentation des projets , fusion impact et évolution des storytelling

Projet Avengers:



Les Avengers sont des personnages de super-héros de l'univers Marvel qui protègent la planète terre des forces du mal. Un Avenger est ainsi caractérisé par une puissance et des superpouvoirs.

Le projet Avengers, se décompose principalement en deux classes:

Avenger: Qui comporte un attribut String nom, un attribut int puissance maximisé de 100 et une liste de superpouvoirs, avec tous les getters et setters.

SuperPower: Qui a un attribut String nom et un Avenger, avec tous les getters et setters.

Projet Virus:



Les virus sont des micro-organismes porteurs de maladies, ils sèment et répandent les maladies au sein de la planète terre. Un virus est caractérisé par une liste de maladies.

Le projet Virus, se décompose principalement en deux classes:

Virus: Qui comporte un attribut String nom, un attribut int gentillesse et une liste de maladies, avec tous les getters et setters.

Disease: Qui a un attribut String nom et un Virus, avec tous les getters et setters.

Storytelling et mise en oeuvre:

La beauté de la terre ne cesse de faire des envieux qui veulent s'en emparer , un grand nombre de créatures extraterrestres ont tenté de s'approprier cette dernière , mais heureusement que les Avengers sont là pour protéger notre chère planète . Hélas en 2045 des créatures surpuissantes menacent la sécurité de la planète bleue , les Avengers à eux seuls ne font pas le poids, alors que tout semblait perdu , et voilà que l'une des créatures les plus malveillantes dans ce monde décide d'aider les Avengers pour sauver cette pauvre terre , mais hélas les virus n'ont pas les compétences pour aider , ils sont habitués à faire du mal uniquement , comment faire ? À l'aide d'une machine construite par notre cher Iron Man sous le nom de "virus 2 Avenger" permet de transformer un virus en un super Avenger prêt pour aider et sauver la planète.



User Story:1 Combattre les intrus !

Scenario: Virus to Avenger

Given La création du Virus "X" qui a "4" gentillesse et "a" et "b" sont des maladies

When Le virus "X" avec gentillesse "4" et maladies "a" et "b" est transformé en Avenger

Then Le Avenger "X" a "96" de puissance et "a" et "b" sont des superpouvoirs

Priorité: Haute

Estimation: 2 semaines

Pour répondre à cette User Story, nous avons décidé d'utiliser le design pattern “**Adapter**”, et cela en convertissant la classe **Virus** en **Avenger**, sous le nom de classe **VirusToAvenger**, comme le démontre le code ci-dessous.

```
public class VirusToAvengerAdapter extends Avenger {  
    private Virus virus;  
  
    public VirusToAvengerAdapter(Virus virus ) {  
        this.virus=virus;  
    }  
  
    public void setName(String nom) {  
        this.virus.setName(nom);  
    }  
    @Override  
    public int getPower() {  
        return (100-this.virus.getKidness());  
    }  
    @Override  
    public void setPower(int value) {  
        this.virus.setKindness(100-value);  
    }  
    public String getName() {  
        return this.virus.getName();  
    }  
    public void addSuperPouvoir(SuperPower superpouvoir) {  
        this.addSuperPouvoir(superpouvoir);  
        this.virus.addDisease(convertSuperPouvoirToDisease(superpouvoir));  
    }  
  
    public ArrayList<SuperPower> getAllSuperPower(){  
        ArrayList<SuperPower> tempList = new ArrayList<SuperPower>();  
        for(Disease disease: this.virus.getDiseases() {  
            tempList.add(convertDiseaseToSuperPouvoir(disease));  
        }  
        return tempList;  
    }  
  
    public static Disease convertSuperPouvoirToDisease(SuperPower superpouvoir )  
    {  
        return new Disease(superpouvoir.getName());  
    }  
  
    public static SuperPower convertDiseaseToSuperPouvoir(Disease disease)  
    {  
        return new SuperPower(disease.getName());  
    }  
}
```

Etant donné que moins un virus est gentil, plus est puissant, il est donc logique que la puissance d'un virus transformé en Avenger soit de 100 - la gentillesse du virus.

Une maladie faisant partie d'un virus et un superpouvoir faisant aussi partie d'un Avenger, ainsi en ajoutant une maladie à un virus, ce dernier transformé en Avenger, gardera toujours ses maladies transformées en superpowers, il est donc plus évident de faire une **conversion** via des méthodes plutôt que de faire deux adapter bidirectionnelles de maladie à superpouvoir et vice versa, comme le démontre le code suivant:

```
public static Disease convertSuperPouvoirToDisease(SuperPower superpouvoir )
{
    return new Disease(superpouvoir.getName());
}

public static SuperPower convertDiseaseToSuperPouvoir(Disease disease)
{
    return new SuperPower(disease.getName());
}
```

Tests Unitaires:

Ayant utilisé la méthodologie TDD, on a donc utilisé toute une panoplie de tests unitaires.

```
public class TestAdapter {
    @Test
    public void testVirusToAvengerGetPower(){
        Virus virus = new Virus(10,"test1");
        VirusToAvengerAdapter virusToAvenger = new VirusToAvengerAdapter(virus);
        assertEquals(90,virusToAvenger.getPower());
    }

    @Test
    public void testVirusToAvengerGetName(){
        Virus virus = new Virus(10,"test2");
        VirusToAvengerAdapter virusToAvenger = new VirusToAvengerAdapter(virus);
        assertEquals(virusToAvenger.getName(),virus.getName());
    }

    @Test
    public void testVirusToAvengerSetPower(){
        Virus virus = new Virus(10,"test3");
        VirusToAvengerAdapter virusToAvenger = new VirusToAvengerAdapter(virus);
        virusToAvenger.setPower(5);
        assertEquals(100-virusToAvenger.getPower(),virus.getKidness());
    }

    @Test
    public void testVirusToAvengerSetName(){
        Virus virus = new Virus(10,"test4");
        VirusToAvengerAdapter virusToAvenger = new VirusToAvengerAdapter(virus);
        virusToAvenger.setName("mister test 4");
        assertEquals(virusToAvenger.getName(),virus.getName());
    }

    @Test
    public void testMaladieToSuperPouvoirGetNom(){
        Disease disease = new Disease("test5");
        SuperPower superPouvoir = convertDiseaseToSuperPouvoir(disease);
        assertEquals(disease.getName(),superPouvoir.getName());
    }
}
```

```

@Test
public void testMaladieToSuperPouvoirGetAvenger(){
    Disease disease = new Disease("disease test7");
    SuperPower superPouvoir = convertDiseaseToSuperPouvoir(disease);
    assertEquals(disease.getName(),superPouvoir.getName());
}

/*
 *
 * Tester uniquement avec le virus devenu avenger
 * on teste la fusion pas l'ancien code !
 */
@Test
public void testGetAllSuperPower(){

    SuperPower superPouvoir = new SuperPower("1");
    SuperPower superPouvoir2 = new SuperPower("2");
    Avenger avenger = new Avenger("nadir",1);
    avenger.addSuperPower(superPouvoir);
    avenger.addSuperPower(superPouvoir2);
    ArrayList<SuperPower> superPouvoirs = new ArrayList<SuperPower>();
    superPouvoirs.add(superPouvoir);
    superPouvoirs.add(superPouvoir2);
    assertEquals(superPouvoirs,avenger.getAllSuperPower());
}
}

```

Cucumber:

Afin de valider notre user story:

```

public class Feature2 {

    @Given("La creation du Virus {string} qui a {string} gentillesse et {string} et {string} sont des maladies")
    public Virus creation(String string, String string2, String string3, String string4) {
        Virus virus = new Virus(Integer.parseInt(string2),string);
        Disease disease=new Disease(string3);
        Disease disease1 = new Disease(string4);
        virus.addDisease(disease);
        virus.addDisease(disease1);
        return virus;
    }

    @When("Le virus {string} avec gentil {string} et maladies {string} et {string} est transforme en avenger")
    public VirusToAvengerAdapter virusAvenger(String string, String string1, String string2,
                                              String string3) {
        return new VirusToAvengerAdapter(creation(string,string1,string2,string3));
    }

    @Then("Le Avenger {string} a {string} de puissance et {string} et {string} sont des superpouvoirs")
    public void classeDuVirus(String string, String string1, String string2, String string3) {
        // Write code here that turns the phrase above into concrete actions
        VirusToAvengerAdapter av= virusAvenger(string,string1,string2,string3);
        Virus virus = creation(string,string1,string2,string3);
        assertEquals(100-av.getPower(),virus.getKidness());
        assertEquals(av.getName(),virus.getName());
        for(int i =0; i<virus.getDiseases().size();i++)
            assertEquals(virus.getDiseases().get(i).getName(),av.getAllSuperPower().get(i).getName());
    }
}

```

Tous les virus se sont alliés aux Avengers sauf un seul Virus "sars-cov-19".

le plus dangereux des virus , pas une nuance de gentillesse dans le coeur de celui-ci (la valeur de sa gentillesse est de -1) .Avide de pouvoir il ne recule devant rien , il n'a épargné aucun virus et n'a cessé de voler des maladies à ces confrères ,c'est ainsi qu'il est devenu le virus le plus troublant de l'univers .



Sans "sars-cov-19" le combat sera dur , mais nos Avengers et nos virus ne lâchent rien ,le combat fait rage, on dénombre de nombreux blessés et morts des deux côtés . Après 5 ans de guerre, les Avengers ont finalement gagné cette guerre !

Pendant que les Avengers et les virus étaient en train de sauver la planète terre du désastre."sars-cov-19" en a profité pour bidouiller la machine et c'est comme ça que ce fourbe est devenu l'être le plus puissant de la galaxie , Thanos est né.

La machine détruite, aucun autre Thanos ne peut être Créer , Thanos est unique !

La machine n'étant plus fonctionnelle les virus ne peuvent plus revenir à leur nature de virus et reprendre le cours de leur vie , Thanos a tout manigancé pour éliminer tout le monde, et cette attaque extraterrestre 'arrange bien car avant cette transformation il ne pouvait s'attaquer qu'aux virus uniquement, mais avec cette transformation il pourra également tuer les Avengers ! .

User Story:2 Naissance de Thanos !

Scenario: Un seul Thanos

Given La creation virus "sars-cov-19"

When Le virus "Thanos" se transforme en supermechant

Then Le supermechant "Thanos" est unique

Priorité: Haute

Estimation: 1 semaine

Pour répondre à cette User Story, nous avons décidé d'utiliser le design pattern "**Singleton**", et cela en créant une classe que l'on ne peut instancier qu'une seule fois un objet, sous le nom de "Thanos" et donc le virus doit avoir "-1" en gentillesse.

L'implémentation du singleton est thread safe et cela pour garantir l'unicité de l'instance même en cas de multi-threading.

```

public class SuperEvil extends VirusToAvengerAdapter{

    private static SuperEvil instance;

    private SuperEvil(Virus virus) {
        super(virus);
        this.power=400;
        this.name="Thanos";
        //this.addSuperPower(new SuperPouvoir("Destruction"));
    }

    public String getName(){
        return this.name;
    }

    public void setName(String s){

    }

    /**
     * Thread Safe singelton
     * @param virus
     * @return SuperEvil instance
     */
    public static synchronized SuperEvil getInstance(Virus virus) {
        if(virus.getKidness() != -1 && virus.getName().equals("")) {
            //Ce virus n'est pas assez puissante pour devenir Thanos !
            return null;
        }
        else if (instance==null) {
            instance = new SuperEvil(virus);
            return instance;
        }
        else return instance;
    }

    @Override
    public int getPower() {
        return this.power;
    }
}

```

Cette classe hérite bien évidemment de **VirusToAvenger**, et on a une seule objet que l'on peut instancier, ayant puissance 400 (très puissant !) et dont le nom est **Thanos**.

Pour instancier ce **SuperEvil**, le virus doit avoir **-1** en gentillesse (vraiment pas gentil le virus hein).

Tests Unitaires:

```

public class TestSingleton {
    @Test
    public void testSuperMechantName(){
        Virus virus = new Virus(-1,"tha");
        SuperEvil superMechant = SuperEvil.getInstance(virus);
        assertEquals("Thanos",superMechant.getName());
    }

    @Test
    public void testSuperMechantSetName(){
        Virus virus = new Virus(-1,"string");
        SuperEvil superMechant = SuperEvil.getInstance(virus);
        superMechant.setName("nadir el 3amiq");
        assertEquals("Thanos",superMechant.getName());
    }

    @Test
    public void testPuissanceSuperMechant(){

```

```

        SuperEvil superMechant = SuperEvil.getInstance(new Virus(-1," "));
        assertEquals(400,superMechant.getPower());
    }

    @Test
    public void testUniciteSuperMechant(){
        SuperEvil superMechant = SuperEvil.getInstance(new Virus(-1," 1"));
        SuperEvil superMechant1 = SuperEvil.getInstance(new Virus(-1," 2"));
        assertEquals(superMechant1,superMechant);
    }

}

```

Cucumber:

```

public class Feature3 {
    @Given("La creation virus {string}")
    public Virus creationDunVirus(String string) {
        return new Virus(5,string);
    }

    @When("Le virus {string} se transforme en supermechant")
    public SuperEvil virusMechant(String string) {
        SuperEvil superMechant = SuperEvil.getInstance(creationDunVirus(string));

        return superMechant;
    }

    @Then("Le supermechant {string} est unique")
    public void virusMechantOuPas(String string) {
        SuperEvil superMechant = virusMechant(string);
        SuperEvil superMechant2 = SuperEvil.getInstance(new Virus());
        assertEquals(superMechant2,superMechant);
    }
}

```

Comment combattre ce Thanos ? Encore une fois, notre chère Iron Man a eu une idée de génie ! fusionner les Avengers pour donner naissance à un **FusionAvenger** qui lui seul pourra rivaliser avec Thanos ,comme le dit si bien le proverbe “l’union fait la force”.



User story: 3 L’union fait la force

Scenario: Un seul Thanos

Given La creation virus "sars-cov-19"

When Le virus "Thanos" se transforme en supermechant

Then Le supermechant "Thanos" est unique

Priorité: Haute

Estimation: 1 semaine

Pour répondre à cette User Story, nous avons décidé d'utiliser le design pattern “**Composite**”, et cela en créant une classe où l'on va regrouper plusieurs Avengers, avec tous les superpouvoirs de ces derniers, et la somme de leurs puissances.

L'implémentation a été faite de manière à ce qu'elle permette aussi de faire une fusion d'une autre fusion d'Avengers.

```
public class FusionAvenger extends Avenger{
    private List<Avenger> listeAvenger;
    /**
     * Class model for fusionAvenger
     * @param name
     * @param avengers
     */
    public FusionAvenger(String name,Avenger...avengers) {
        this.name=name;
        listeAvenger = new ArrayList<Avenger>();
        for(Avenger avenger: avengers) {
            this.listeAvenger.add(avenger);
        }
    }

    @Override
    public int getPower() {
        int sum=0;
        for(Avenger avenger : listeAvenger) {
            if (avenger instanceof FusionAvenger) {
                //to delete
                sum=sum+avenger.getPower();
            } else {
                sum=sum+avenger.getPower();
            }
        }
        return sum;
    }

    public List<SuperPower> getAllSuperPowerFusion() {
        ArrayList<SuperPower> allTheSuperPower = new ArrayList<SuperPower>();
        for(int i = 0 ; i < listeAvenger.size(); i++) {
            for(int j=0; j<listeAvenger.get(i).getAllSuperPower().size();j++){
                allTheSuperPower.add(listeAvenger.get(i).getAllSuperPower().get(j));
            }
        }
        return allTheSuperPower;
    }
}
```

Nous avons la liste de tous les Avengers participant à la fusion, une méthode pour récupérer la puissance de la fusion en sommant la puissance des Avengers, et une méthode pour récupérer tous les superpouvoirs de cette fusion.

Tests Unitaires:

```
public class TestFusionAvenger {  
    @Test  
    public void testFusionGetPouvoir(){  
        Avenger avenger1 = new Avenger("gerald",50);  
        Avenger avenger2 = new Avenger("sawsaw",50);  
        SuperPower superPouvoir1 = new SuperPower("Magic");  
        SuperPower superPouvoir2 = new SuperPower("Sword");  
        SuperPower superPouvoir3 = new SuperPower("Lying");  
        avenger1.addSuperPower(superPouvoir1);  
        avenger1.addSuperPower(superPouvoir2);  
        avenger2.addSuperPower(superPouvoir3);  
  
        ArrayList<SuperPower> superPouvoirs = new ArrayList<SuperPower>();  
        superPouvoirs.add(superPouvoir1);  
        superPouvoirs.add(superPouvoir2);  
        superPouvoirs.add(superPouvoir3);  
        FusionAvenger fusionAvenger = new FusionAvenger("la fusion",avenger1,avenger2);  
        boolean test=true;  
        for(int i = 0;i< superPouvoirs.size();i++) {  
            if (fusionAvenger.getAllSuperPowerFusion().get(i)!=superPouvoirs.get(i))  
                test=false;  
        }  
        assert(test);  
    }  
  
    @Test  
    public void testFusionGetPower(){  
        Avenger avenger1 = new Avenger("gerald",50);  
        Avenger avenger2 = new Avenger("sawsaw",35);  
        FusionAvenger fusionAvenger = new FusionAvenger("la fusion",avenger1,avenger2);  
        assertEquals(avenger1.getPower()+avenger2.getPower(), fusionAvenger.getPower());  
  
    }  
    @Test  
    public void testFusionOfFusionGetPower(){  
        Avenger avenger1 = new Avenger("gerald",50);  
        Avenger avenger2 = new Avenger("sawsaw",35);  
        FusionAvenger fusionAvenger = new FusionAvenger("la fusion",avenger1,avenger2);  
        FusionAvenger fusionAvenger2 = new FusionAvenger("La fusion d'une fusion" ,fusionAvenger,fusionAvenger);  
        assertEquals(fusionAvenger.getPower()+fusionAvenger.getPower(), fusionAvenger2.getPower());  
    }  
}
```

Cucumber:

```
public class Feature4 {  
    @Given("La creation de fusion {string} qui se compose du avenger {string} et" +  
        " {string} qui ont du pouvoir {string} et {string}")  
    public FusionAvenger creationFusion(String string, String string1, String string2,  
        String string3, String string4) {  
        FusionAvenger fusionAvenger = new FusionAvenger(string,new Avenger(string1,Integer.parseInt(string3)),  
            new Avenger(string3,Integer.parseInt(string4)));  
        return fusionAvenger;  
    }  
  
    @When("La fusion happens")  
    public FusionAvenger fusion() {  
        return null;  
    }  
}
```

```

@Then("La fusion {string} se fait on a la somme du pouvoir {string} et " +
    "{string} qui ont {string} et {string} a puissance {string}")
public void virusMechantOuPas(String string, String string1, String string2,
    String string3, String string4, String string5) {
    FusionAvenger fusionAvenger = new FusionAvenger(string,new Avenger(string1,Integer.parseInt(string3)),
        new Avenger(string3,Integer.parseInt(string4)));

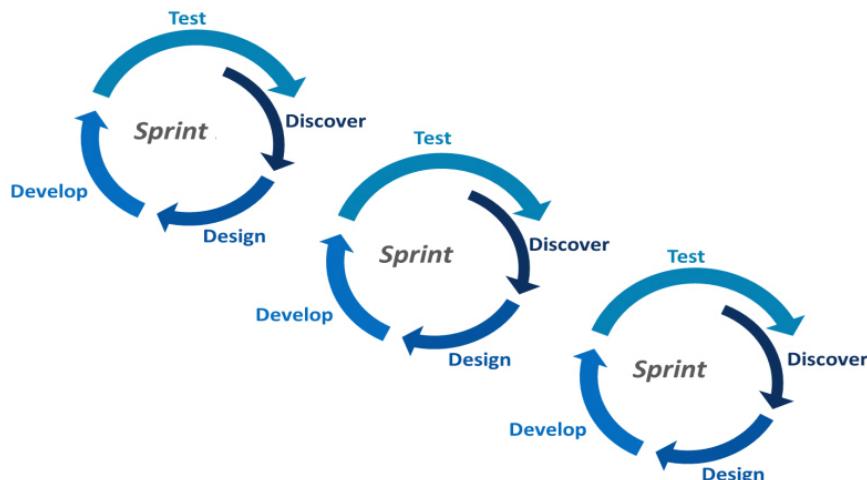
    assertEquals(fusionAvenger.getPower(),Integer.parseInt(string3)+Integer.parseInt(string4));
}

}

```

Outils & méthodes utilisés:

Sprint:

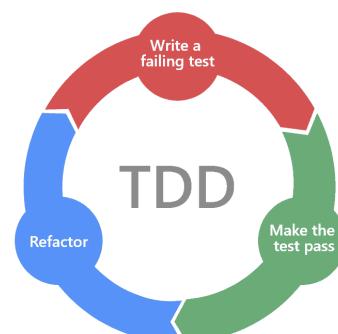


Nous avons au cours de ce projet, principalement usé de 3 Sprint, dans lesquelles, nous avons discuté des spécifications et du développement dérivé par les tests.

Test-Driven Development:

Au vu que nous avons utilisé les pratiques de l'Extreme Programming, nous avons donc fait usage des **TDD**, alors qu'est ce que la **TDD**?

Le **Test-Driven Development (TDD)** est une méthode de développement de logiciel, qui consiste à concevoir une solution informatique de façon itérative et incrémentale, en écrivant chaque test avant d'écrire le code source et en remaniant le code continuellement.



Même si le développement basé sur des tests peut, au premier abord, sembler contre-intuitif, il est logique et conduit à de meilleurs résultats. Tandis que les procédures conventionnelles de test en aval

utilisent un modèle en cascade ou en v, les processus du TDD sont cycliques. Cela signifie que vous devez d'abord déterminer les cas tests qui échouent fréquemment. Cela est fait exprès, car dans la deuxième étape, vous n'écrivez que la quantité de code nécessaire pour réussir le test. Les composants sont ensuite remaniés : tout en conservant ses fonctions, vous étendez le code source ou le restructurez si nécessaire.

SonarLint:

SonarLint est un plugin permettant de réaliser les analyses de code SonarQube au fil des développements, directement dans l'IDE. Un bon coup de pouce pour identifier au plus tôt, c'est-à-dire avant même le commit, les points à corriger. De ce fait, le coût de la qualité du code est extrêmement réduit, voire invisible, à l'inverse des lourdes campagnes correctives qui interviennent sur du code déjà mergé depuis longtemps.

Git:

Git est un système de contrôle de version décentralisé, gratuit et open source conçu pour la gestion des grands comme les petits projets.



`master` Updated 6 minutes ago by NadirOmega

`User_story_3_L'unione_fait_la_force` Updated 6 minutes ago by NadirOmega

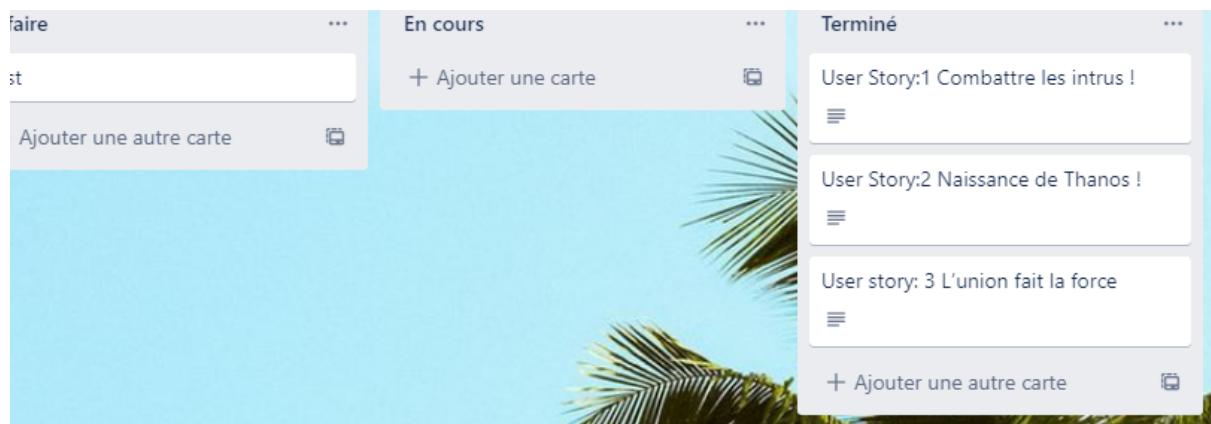
`User_Story2Naissance_de_Thanos` Updated 2 hours ago by spanow

`User_story1_Combattre_les_intrus` Updated 2 days ago by NadirOmega

Trello:

Trello est un outil de gestion de projet en ligne inspiré par la méthode Kanban de Toyota. Il repose sur une organisation des projets en planches listant des cartes, chacune représentant des tâches.





💻 User Story:2 Naissance de Thanos !

Dans la liste Terminé

☰ Description [Modifier](#)

Scenario: Un seul Thanos

Given La creation virus "sars-cov-19"

When Le virus "Thanos" se transforme en supermechant

Then Le supermechant "Thanos" est unique

Priorité: Haute

Estimation: 1 semaine

🕒 Activité

[Afficher les détails](#)



Écrivez un commentaire...

Conclusion:

Nous avons tout au long de ce projet non seulement appris à user des techniques de l'extreme programming dans un environnement agile en utilisant du refactoring, des TDD et autres, mais nous avons surtout appris à travailler en équipe, en gérant les différents sprint!