National Institut of Applied Sciences and Technology

UNIVERSITY OF CARTHAGE

Graduation Project

Speciality : **GL**

Flouci Developers API

Presented By

Sarra MGHAIETH

University Supervisor : **Mr SELLAOUTI Aymen**
Enterprise Supervisor : **Mr KALLEL Anis**

Presented the : –/–/**2019**

JURY

M. President	FLEN	(President)
Mme. Rapporteur	FLENA	(Rapporteur)

Annee Universitaire : 2018/2019

Dédicaces

« Soyons reconnaissants aux personnes qui nous donnent du bonheur ; elles sont les charmants jardiniers par qui nos âmes sont fleuries »

A LA MÉMOIRE DE MON TRÈS CHER GRAND-PÈRE CHERIF ZOUBAIER

J'aurais tant aimé que tu sois à mes côtés en ce jour si spécial. Que Dieu ait ton âme dans sa sainte miséricorde.

A MA GRAND-MÈRE CHERIF FAOUZIA

Que ce modeste travail soit l'expression des vœux que tu n'as cessé de formuler dans tes prières. Que Dieu te préserve santé et longue vie.

A MON PÈRE MONDHER, A MA MÈRE SONIA

Autant de phrases aussi éloquentes soit-elles ne sauraient exprimer ma gratitude et ma reconnaissance. Vous avez su me transmettre le sens de la responsabilité, de l'optimisme et de la confiance en soi face aux difficultés de la vie. Vos conseils ont toujours guidé mes pas vers la réussite. Votre patience sans fin, votre compréhension et vos encouragements sont pour moi le soutien indispensable que vous avez toujours su m'apporter. Je vous dois ce que je suis aujourd'hui et ce que je serai demain et je ferai toujours de mon mieux pour rester votre fierté et ne jamais vous décevoir. Que Dieu le tout puissant vous préserve, vous accorde santé, bonheur, et vous protège de tout mal.

A MON FRÈRE AZIZ

En témoignage de mon affection fraternelle, de ma profonde tendresse et reconnaissance, je te souhaite une vie pleine de bonheur et de succès et que Dieu, le tout puissant, te protège.

A mon oncle Anis, son épouse Nadine, mon oncle Kamel, ainsi que mes cousins Skander, Ilyas, Rayan et ma cousine Lara milea

Veuillez trouver dans ce travail l'expression de mon respect le plus profond et mon affection la plus sincère

A ma soeur de coeur BENNANI Khedija

En souvenir de notre sincère et profonde amitié et des moments agréables que nous avons passés ensemble, je te souhaite une vie pleine de bonheur et de succès

A tous mes ami(e)s

Merci pour les bons moments qu'on a partagé ensemble et pour votre soutien. Je serai toujours à vos côtés

A toute personne qui m'est chère je dédie ce modeste travail...

Remerciements

Ce travail n'aurait pu aboutir sans le soutien et les encouragements, au quotidien, des personnes qui m'ont accompagné durant le stage.

Tout d'abord, j'adresse mes remerciements à **Mr.Hamdi Meddeb** qui m'a offert l'opportunité de réaliser ce projet de fin d'études qui était en totale adéquation avec mes attentes.

Je tiens aussi à présenter mes remerciements à mon encadrante pédagogique, **Mme.Nadia Chammem**, maître assistant à l'Institut National des Sciences Appliquées et de Technologie au sein du département de biologie, d'avoir accepté de m'encadrer , de m'avoir permis de mener à bien ma mission par ses remarques et ses conseils judicieux et surtout de m'aider à améliorer la présentation et la rédaction de ce document.

Je remercie également l'ensemble des collaborateurs de la **STIAL Délice Danone** qui par leur gentillesse et leur convivialité ont facilité mon insertion au sein de l'entreprise, m'ont permis d'être associée aux projets et d'acquérir de nouvelles connaissances et compétences.

Je remercie plus particulièrement mon tuteur de stage **Mme.Amel Ridène**, occupant la fonction ingénieur Management qualité et sécurité alimentaire, pour sa disponibilité et le temps qu'elle a su me consacrer, pour ses conseils et aides sans lesquels ce travail n'aurait pu être accompli.

Je n'oublierai jamais l'expérience que j'ai acquise au sein de cette équipe, qui fut une expérience enrichissante et passionnante.

Je tiens également à exprimer ma profonde gratitude à l'ensemble des membres du jury pour avoir accepté de participer à l'évaluation et à l'enrichissement de mon travail.

Table of Contents

List of Figures

List of Tables

Summary

Brief summary ...

General Introduction

De nos jours, les consommateurs du secteur agroalimentaire sont devenus de plus en plus exigeants ce qui a créé chez les industriels, un besoin de maîtriser et d'optimiser la qualité des aliments, rendant ainsi, la sécurité des denrées alimentaires et la traçabilité, des atouts stratégiques pour la filière agroalimentaire. En effet, la pression exercée par les clients quant à la qualité du produit de consommation, conduit les entreprises agroalimentaires et les autorités à contrôler non seulement les produits finis mais toute la chaîne de production. Or, + fabriquer un produit sain, salubre et de qualité implique de répondre aux exigences réglementaires relatives à la qualité et à la sécurité des denrées alimentaire

Ainsi, pour maintenir la compétitivité industrielle, toute entreprise du secteur, doit passer par le renforcement de ses capacités à fournir à ses clients des produits conformes aux exigences législatives et réglementaires en vigueur.

Face à une demande de plus en plus importante des clients à la communauté agro-alimentaire pour qu'elle démontre son aptitude à identifier et à maîtriser les dangers liés à la sécurité des denrées alimentaires ;cette dernière a multiplié les initiatives pour établir des règles plus ou moins volontaires. Au sein d'une révolution des référentiels de qualité, la norme FSSC 22000 est le dernier standard de certification pour les fabricants de produits alimentaires. Sa nouvelle version a été publiée le 3 juin 2019. Cette version 5 du référentiel de Food Safety de référence est une version révisée de FSSC 22000 v 4.1, officiellement lancée en juillet 2017.

Consciente des avantages apportés par cette norme en termes de la gestion de la qualité des produits mis à la disposition de ses clients, l'entreprise Société Tunisienne des Industries Alimentaire Laitières (STIAL) Délice Danone s'oriente vers une politique d'amélioration continue et pour renforcer plus son système de management de la sécurité des aliments , la société a introduit la norme FSSC 22000, qui représente l'une des approches les plus exhaustives de ce système et combine les avantages d´un outil de management commercial lié à sécurité des denrées alimentaires et les processus d'affaires avec la capacité de répondre aux exigences de la clientèle mondiale.

Les modifications apportées au référentiel FSSC 22 000 en sa dernière version 5 va rendre obligatoire de mettre à jour le système de sécurité des denrées alimentaire au niveau de la STIAL Délice Danone puisque les audits selon les exigences de la version 4.1 ne sont autorisés que jusqu'au 31 décembre 2019 et tous les certificats FSSC 22000 v 4.1 délivrés deviendront in-

valides après le 29 juin 2021. Les audits de mise à niveau par rapport aux exigences du système FSSC 22000 v 5 seront à effectuer entre le 1er janvier 2020 et le 31 décembre 2020. Dans ce sens, l'objectif assigné à ce stage de fin d'études se résume dans la participation à la migration de ce système selon le protocole de certification FSSC 22 000 version 5 et plus particulièrement à la mise en place des processus de l'entreprise exigée par la nouvelle version de l'ISO 22 000 : 2018 , un des constituants de la FSSC 22000 version 5.

Pour cela, un premier chapitre présente le système de management de la sécurité alimentaire selon le protocole de certification FSSC 22 000 V5 et ses principales exigences , Un deuxième et un troisième chapitre s'intéressent successivement à l'identification, la mise en place des processus de la société et à l'analyses des risques relative à chaque processus ainsi que les actions à mettre en oeuvre face aux risques et opportunités selon la norme ISO 22000 : 2018

Chapitre I

Présentation de l'entreprise

Plan

Introduction

1 Fiche technique de STIAL Délice Danone

2 Produits commercialisés

3 Organigramme de STIAL Délice Danone

4 La politique qualité et sécurité des aliments de STIAL Délice Danone

5 Les réalisations en matière de satisfactions des clients

6 STIAL Délice Danone et l'environnement

7 Diagramme de production

Figure I.1 – Kaoun logo

Chapitre II

Project Development Practices

Plan

Introduction

This chapter is introducing the software development disciplines and rules followed during the achievement of our project. In order to have a clear development structure, a development process must be set in place in the earliest stage of our project life.

Our development process is a combination of different practices like Test-driven development and DevOps.

1 Kanban Tickets Life Cycle

Following the Kanban methodology our project is decomposed to small tickets with limited scopes. The tickets are developed in an incremental way following a priority order, and together they shape our project to it's final version.

1.1 Backlog

All tickets are created in the "Backlog" stage, any work that needs to be done is formed into a ticket.

The ticket needs to have a self-explanatory description, to make it simple for the developer to start working on it without the need for further explanation.

A Tag must be assigned to the ticket as well, tags could be referring to the nature of the work to do (Bug, Fix, Task...), or the scope (Back-end, Front-end, DevOps...).

In the end the ticket priority must be set following a priority system set by the development team. The system could relay on numerical values (0 having the least priority, 1, 2 ...), or having specific tags (LOW, MEDIUM, HIGHT, URGENT) which we will be using in our project.

Tickets could only move from "Backlog" to "Selected For Development".

1.2 Selected For Development

In the "Selected For Development" stage developers have access to the tickets. They have to tackle the ticket following the priority system and the tag matching their expertise (Back-end, Front-end).

Once they choose a ticket they must assign it to their name, move it to the "In Progress" stage and create a git branch with the ticket name and finally start developing.

Tickets could only move from "Selected For Development" to "In Progress".

1.3 In Progress

The "In Progress" stage serve as safety mechanism to avoid having two developers working on the same ticket.

This stage indicates that the tickets is being taking care of, it also shows the person developing the ticket.

If the developer finish his work he should move the ticket to "Code Review" stage and make a pull request of the branch.

In the case of failure ticket should go back to "Selected For Development" stage.

1.4 Code Review

Once a ticket is in the "Code Review" stage, the project manager should open the pull request and check the code committed by the developer.

In case of approval the pull request is merged and deployed, the ticket then is moved to "QA".

In case of disapproval the project manager leaves comments on the pull request and moves the ticket back to "In progress". The developer then needs to check the code and fix the issue.

1.5 QA

In the "QA" stage tickets are tested by fellow developers, the functionality of the ticket should be tested on an environment similar to the production environment, also developer should push the test to the limit and test all edge cases.

If all developers approve that the code is working fine in every possible scenario the ticket is moved to done, otherwise the ticket info should be updated with the issues encountered and the ticket is moved back to the "In Progress" stage.

1.6 Done

All tickets should finally be moved to the "Done" stage, this stage groups all the work done and keeps track of the progress of the development.

After a fixed time tickets get archived to gain space in the Kanban board.

2 Gitflow workflow

Gitflow Workflow is a Git workflow design that was first published and made popular by Vincent Driessen at nvie. The Gitflow Workflow defines a strict branching model designed around the project release. This provides a robust framework for managing larger projects.

Gitflow is really just an abstract idea of a Git workflow. This means it dictates what kind of branches to set up and how to merge them together.

2.1 Feature Branch

The feature branch is created by the developer for every Kanban ticket he tackles. When moving the ticket to the "In Progress" stage the branch should be created with the same name as the ticket.

All the development specific to the scope of the ticket should be done in the same branch, and in the end the developer should make a pull request on the Develop Branch.

2.2 Develop Branch

The Develop branch present the edge version of the product, it contains all new developed features. This branch is the main branch to the internal project team, All newly developed feature are tested by developers on this branch, it maps to the "dev environment".

This branch could present a number of bugs because by nature it's a testing and validation branch of the latest developed code.

This branch is daily updated.

2.3 Master Branch

After tickets validation and bug fixes, the internal team merge a stable version of the product (Develop branch) to the Master branch. This branch maps to the staging environment and serves for testing on the company level.

All company projects depending on our project always use its staging version. The testing and integration with other projects should be verified extensively before merging to the production environment.

2.4 Release Branch

This is the production branch, code on this branch has been tested twice in both the dev and staging environments.

The branch is updated according to the release dates, it's the result of merging a stable master branch.

The branch maps to the prod environment and it's the actual branch used by the end user.

2.5 Hotfix Branch

As it is impossible to have a bug free software, we should take in consideration bugs popping in the prod environment.

Every bug discovered on the prod environment opens an urgent fix ticket, the ticket branch is based on the release branch and its merged directly to the release branch after the fix.

This is a measure to quickly fix issues that have effects on the end user.

3 Test-driven development

Test Driven Development is a development technique that requires the writing of tests even before writing the first line of code.

In theory, the method requires the intervention of at least two different people, one person writes the tests, the other writes the code. This avoids problems related to subjectivity.

In practice things are more complicated, sometimes you develop alone or you write the tests yourself that guarantee the integrity of a new functionality in a collaborative project.

The TDD can be divided into 5 distinct steps :

1. Write a test.
2. Check that it fails.
3. Write the code enough for the test to pass.
4. Check that the test passes.
5. Optimize the code and check that there is no regression.

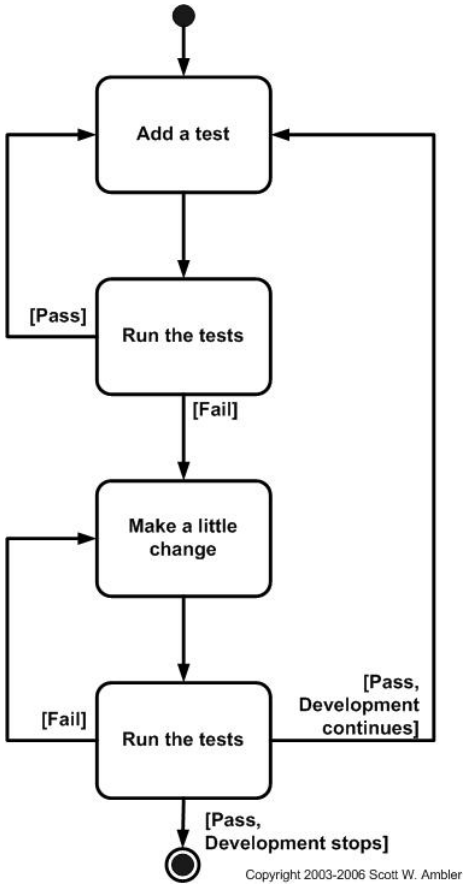

Figure II.1 – TDD Steps

In our project, The development of every ticket starts by writing the test. A pull request can only be approved if it contains and passes the new test.

This discipline enable our project to benefits from the DevOps world, As it's important to have test for the CI-CD process to be defined.

4 DevOps

DevOps is a set of practices that automates the processes between software development and IT teams, in order that they can build, test, and release software faster and more reliably.

In our project we need to have an autonomous testing and deployment process to maintain our three environments :

— **Dev environment :** Represents the Develop branch and used for internal team testing.

— **Staging environment :** Represents the Master branch and used for a company level testing.

— **Prod environment :** Represents the Release branch and used by the end user.

4.1 DevOps Steps

With every branch push our CI-CD server (Jenkins) triggers an automatic script that consists of a set of predefined steps.

The steps behave slightly different according to the branch name.

4.1.1 Preparation

The first step of every build is getting the branch code and preparing a clean environment containing all the dependencies required to run our project.

4.1.2 Testing

In this step all tests are executed for both backend and frontend. A full report is created with the test results. We can only continue the build if all tests are passed gracefully otherwise the job is terminated.

4.1.3 Quality Measurements

In this steps the code is inspected by the open-source platform SonarQube.

The step performs automatic reviews with static analysis of code to detect bugs, code smells, and security vulnerabilities.

4.1.4 Dockerization

If all testing and quality measures are approved we proceed to the Dockerization of our project. This step creates a standard artifact that can be deployed on any infrastructure without the need of specific configurations for different servers.

If the Docker images is created successfully, it will be tagged latest and pushed to the Docker Hub.

4.1.5 Deployment

In the last step of the build, and according to the branch name the new artifact is deployed to the convenient server (dev, staging, prod).

On every target server we have a docker swarm cluster to deploy the new version of the product. The swarm cluster is also configured with the different environment variables as secrets to protect the different credentials (database, cloud services...).

5 Tools

In the section we will present the set of tools that make it possible to follow the development disciplines mentioned above. The tools also help the automation of the processes.

5.1 Bitbucket

Bitbucket is a web-based version control repository hosting service owned by Atlassian, for source code and development projects that use Git revision control systems.

5.2 GitKraken Glo Boards

GitKraken Glo Boards is a software that manage boards and tickets. In our case the Kanban board is created and managed on GitKraken.

5.3 Docker

Docker is a tool designed to make it easier to create, deploy, and run applications by using containers. Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and ship it all out as one package. By doing so, thanks to the container, the developer can rest assured that the application will run on any other machine regardless of any customized settings that machine might have that could differ from the machine used for writing and testing the code.

5.4 Docker Swarm

As a platform, Docker has revolutionized the manner software was packaged. Docker Swarm or simply Swarm is an open-source container orchestration platform and is the native clustering engine for and by Docker. Any software, services, or tools that run with Docker containers run equally well in Swarm. Also, Swarm utilizes the same command line from Docker.

Swarm turns a pool of Docker hosts into a virtual, single host. Swarm is especially useful for people who are trying to get comfortable with an orchestrated environment or who need

to adhere to a simple deployment technique but also have more just one cloud environment or one particular platform to run this on.

5.5 Jenkins

In order to have our CI/CD environment, we used Jenkins which is an open source automation server that helps you to automate the non-human part of the software development process.

Jenkins is a stand-alone open source automation server that can be used to automate all kinds of tasks related to software creation, testing, delivery or deployment.

5.6 SonarQube

SonarQube (formerly Sonar) is an open-source platform developed by SonarSource for continuous inspection of code quality to perform automatic reviews with static analysis of code to detect bugs, code smells, and security vulnerabilities on 20+ programming languages.

SonarQube offers reports on duplicated code, coding standards, unit tests, code coverage, code complexity, comments, bugs, and security vulnerabilities.

Conclusion

In this chapter we set the development disciplines to follow in the making of our project.

We went through the Kanban methodology in practice, explaining the different steps of the incremental code writing.

We also set the rules of the git branching model we will be using in our development process, as well as the test-driven development discipline to follow for every branch.

We tackled our DevOps setup and the different steps of the automation process from code writing to project deployment.

In the end we presented the tools we will be using in our project development life cycle.

Chapitre III

Sprint 1 : Project Launch

Plan

Introduction

Ce chapitre porte sur la partie pratique ainsi que la bibliographie.

1 Outils et langages utilises

L'etude technique peut se trouver dans cette partie, comme elle peut etre faite en parallele avec l'etude theorique (comme le suggere le modele 2TUP). Dans cette partie, il faut essayer de convaincre le lecteur de vos choix en termes de technologie. Un etat de l'art est souhaite ici, avec un comparatif, une synthese et un choix d'outils, meme tres brefs.

2 Presentation de l'application

Il est tout e fait normal que tout le monde attende cette partie pour coller e souhait toutes les images correspondant aux interfaces diverses de l'application si chere e votre coeur, mais abstenez vous ! Il FAUT mettre des imprime ecrans, mais bien choisis, et surtout, il faut les scenariser : Choisissez un scenario d'execution, par exemple la creation d'un nouveau client, et montrer les differentes interfaces necessaires pour le faire, en expliquant brievement le comportement de l'application. Pas trop d'images, ni trop de commentaires : concis, encore et toujours.

evitez ici de coller du code : personne n'a envie de voir le contenu de vos classes. Mais vous pouvez inserer des snippets (bouts de code) pour montrer certaines fonctionnalites [1][2], si vous en avez vraiment besoin. Si vous voulez montrer une partie de votre code, les etapes d'installation ou de configuration, vous pourrez les mettre dans l'annexe.

2.1 Exemple de tableau

Vous pouvez utiliser une description tabulaire d'une eventuelle comparaison entre les travaux existants. Ceci est un exemple de tableau : Tab VI.1.

Tableau III.1 – Tableau comparatif

	Col1	**Col2**	**Col3**	**Col4**
Row1		X		
Row2	X			
Row3	X	X	X	X
Row4	X		X	X
Row5	X		X	X
Row6	X		X	X
Row7	X		X	
Row8	X	X	X	

2.2 Exemple de Code

Voici un exemple de code Java, avec coloration syntaxique VI.1.

Listing III.1 – Helloworld Java

```
  public class HelloWorld {
  //la methode main
    public static void main(String[] args) {
        System.out.println("Hello, World");
    }

}
```

3 Remarques sur la bibliographie

Votre bibliographie doit repondre e certains criteres, sinon, on vous fera encore et toujours la remarque dessus (et parfois, meme si vous pensez avoir tout fait comme il faut, on peut vous faire la remarque quand meme : chacun a une conception tres personnelle de comment une bibliographie devrait etre).

- Une bibliographie dans un bon rapport doit contenir plus de livres et d'articles que de sites web : apres tout c'est une biblio. Privilegiez donc les ouvrages reconnus et publies pour vos definitions, au lieu de sauter directement sur le premier article wikipedia ;
- Les elements d'une bibliographie sont de preference classes par ordre alphabetique, ou par themes (et ordre alphabetique pour chaque theme) ;
- Une entree bibliographique doit etre sous la forme suivante :
  - Elle doit contenir un identifiant unique : represente soit par un numero [1] ou par le nom du premier auteur, suivi de l'annee d'edition [Kuntz, 1987] ;
  - Si c'est un livre : Les noms des auteurs, suivi du titre du livre, de l'editeur, ISBN/ISSN, et la date d'edition ;
  - Si c'est un article : Les noms des auteurs, le titre , le journal ou la conference, et la date de publication ;
  - Si c'est un site web ou un document electronique : Le titre, le lien et la date de consultation ;
  - Si c'est une these : nom et prenom, titre de la these, universite de soutenance, annee de soutenance, nombre de pages ;
  - Exemples :

    [*Bazin*, 1992] BAZIN R., REGNIER B. Les traitements antiviraux et leurs essais therapeutiques. Rev. Prat., 1992, 42, 2, p.148-153.

    [*Anderson*, 1998] ANDERSON P.JF. Checklist of criteria used for evaluation of metasites. [en ligne]. Universite du Michigan, Etats Unis. Site disponible sur : http ://www.lib.umich.edu/megasite/critlist.html.(Page consultee le 11/09/1998).
  - Dans le texte du rapport, on doit obligatoirement citer la reference en faisant appel e son identifiant, juste apres avoir utilise la citation. Si ceci n'est pas fait dans les regles, on peut etre accuse de plagiat.

Conclusion

Voile.

Chapitre IV

Sprint 2 : User and App Managment

Plan

Introduction

Ce chapitre porte sur la partie pratique ainsi que la bibliographie.

1 Outils et langages utilises

L'etude technique peut se trouver dans cette partie, comme elle peut etre faite en parallele avec l'etude theorique (comme le suggere le modele 2TUP). Dans cette partie, il faut essayer de convaincre le lecteur de vos choix en termes de technologie. Un etat de l'art est souhaite ici, avec un comparatif, une synthese et un choix d'outils, meme tres brefs.

2 Presentation de l'application

Il est tout e fait normal que tout le monde attende cette partie pour coller e souhait toutes les images correspondant aux interfaces diverses de l'application si chere e votre coeur, mais abstenez vous ! Il FAUT mettre des imprime ecrans, mais bien choisis, et surtout, il faut les scenariser : Choisissez un scenario d'execution, par exemple la creation d'un nouveau client, et montrer les differentes interfaces necessaires pour le faire, en expliquant brievement le comportement de l'application. Pas trop d'images, ni trop de commentaires : concis, encore et toujours.

evitez ici de coller du code : personne n'a envie de voir le contenu de vos classes. Mais vous pouvez inserer des snippets (bouts de code) pour montrer certaines fonctionnalites [1][2], si vous en avez vraiment besoin. Si vous voulez montrer une partie de votre code, les etapes d'installation ou de configuration, vous pourrez les mettre dans l'annexe.

2.1 Exemple de tableau

Vous pouvez utiliser une description tabulaire d'une eventuelle comparaison entre les travaux existants. Ceci est un exemple de tableau : Tab VI.1.

Tableau IV.1 – Tableau comparatif

	Col1	**Col2**	**Col3**	**Col4**
Row1		X		
Row2	X			
Row3	X	X	X	X
Row4	X		X	X
Row5	X		X	X
Row6	X		X	X
Row7	X		X	
Row8	X	X	X	

2.2 Exemple de Code

Voici un exemple de code Java, avec coloration syntaxique VI.1.

Listing IV.1 – Helloworld Java

```
public class HelloWorld {
//la methode main
  public static void main(String[] args) {
      System.out.println("Hello, World");
  }

}
```

3 Remarques sur la bibliographie

Votre bibliographie doit repondre e certains criteres, sinon, on vous fera encore et toujours la remarque dessus (et parfois, meme si vous pensez avoir tout fait comme il faut, on peut vous faire la remarque quand meme : chacun a une conception tres personnelle de comment une bibliographie devrait etre).

- Une bibliographie dans un bon rapport doit contenir plus de livres et d'articles que de sites web : apres tout c'est une biblio. Privilegiez donc les ouvrages reconnus et publies pour vos definitions, au lieu de sauter directement sur le premier article wikipedia ;
- Les elements d'une bibliographie sont de preference classes par ordre alphabetique, ou par themes (et ordre alphabetique pour chaque theme) ;
- Une entree bibliographique doit etre sous la forme suivante :
  - Elle doit contenir un identifiant unique : represente soit par un numero [1] ou par le nom du premier auteur, suivi de l'annee d'edition [Kuntz, 1987] ;
  - Si c'est un livre : Les noms des auteurs, suivi du titre du livre, de l'editeur, ISBN/ISSN, et la date d'edition ;
  - Si c'est un article : Les noms des auteurs, le titre , le journal ou la conference, et la date de publication ;
  - Si c'est un site web ou un document electronique : Le titre, le lien et la date de consultation ;
  - Si c'est une these : nom et prenom, titre de la these, universite de soutenance, annee de soutenance, nombre de pages ;
  - Exemples :

    [*Bazin*, 1992] BAZIN R., REGNIER B. Les traitements antiviraux et leurs essais therapeutiques. Rev. Prat., 1992, 42, 2, p.148-153.

    [*Anderson*, 1998] ANDERSON P.JF. Checklist of criteria used for evaluation of metasites. [en ligne]. Universite du Michigan, Etats Unis. Site disponible sur : http ://www.lib.umich.edu/megasite/critlist.html.(Page consultee le 11/09/1998).
  - Dans le texte du rapport, on doit obligatoirement citer la reference en faisant appel e son identifiant, juste apres avoir utilise la citation. Si ceci n'est pas fait dans les regles, on peut etre accuse de plagiat.

Conclusion

Voile.

Chapitre V

Sprint 3 : Checkout API

Plan

Introduction

Ce chapitre porte sur la partie pratique ainsi que la bibliographie.

1 Outils et langages utilises

L'etude technique peut se trouver dans cette partie, comme elle peut etre faite en parallele avec l'etude theorique (comme le suggere le modele 2TUP). Dans cette partie, il faut essayer de convaincre le lecteur de vos choix en termes de technologie. Un etat de l'art est souhaite ici, avec un comparatif, une synthese et un choix d'outils, meme tres brefs.

2 Presentation de l'application

Il est tout e fait normal que tout le monde attende cette partie pour coller e souhait toutes les images correspondant aux interfaces diverses de l'application si chere e votre coeur, mais abstenez vous ! Il FAUT mettre des imprime ecrans, mais bien choisis, et surtout, il faut les scenariser : Choisissez un scenario d'execution, par exemple la creation d'un nouveau client, et montrer les differentes interfaces necessaires pour le faire, en expliquant brievement le comportement de l'application. Pas trop d'images, ni trop de commentaires : concis, encore et toujours.

evitez ici de coller du code : personne n'a envie de voir le contenu de vos classes. Mais vous pouvez inserer des snippets (bouts de code) pour montrer certaines fonctionnalites [1][2], si vous en avez vraiment besoin. Si vous voulez montrer une partie de votre code, les etapes d'installation ou de configuration, vous pourrez les mettre dans l'annexe.

2.1 Exemple de tableau

Vous pouvez utiliser une description tabulaire d'une eventuelle comparaison entre les travaux existants. Ceci est un exemple de tableau : Tab VI.1.

Tableau V.1 – Tableau comparatif

	Col1	**Col2**	**Col3**	**Col4**
Row1		X		
Row2	X			
Row3	X	X	X	X
Row4	X		X	X
Row5	X		X	X
Row6	X		X	X
Row7	X		X	
Row8	X	X	X	

2.2 Exemple de Code

Voici un exemple de code Java, avec coloration syntaxique VI.1.

Listing V.1 – Helloworld Java

```
  public class HelloWorld {
  //la methode main
    public static void main(String[] args) {
        System.out.println("Hello, World");
    }

}
```

3 Remarques sur la bibliographie

Votre bibliographie doit repondre e certains criteres, sinon, on vous fera encore et toujours la remarque dessus (et parfois, meme si vous pensez avoir tout fait comme il faut, on peut vous faire la remarque quand meme : chacun a une conception tres personnelle de comment une bibliographie devrait etre).

- Une bibliographie dans un bon rapport doit contenir plus de livres et d'articles que de sites web : apres tout c'est une biblio. Privilegiez donc les ouvrages reconnus et publies pour vos definitions, au lieu de sauter directement sur le premier article wikipedia ;
- Les elements d'une bibliographie sont de preference classes par ordre alphabetique, ou par themes (et ordre alphabetique pour chaque theme) ;
- Une entree bibliographique doit etre sous la forme suivante :
  - Elle doit contenir un identifiant unique : represente soit par un numero [1] ou par le nom du premier auteur, suivi de l'annee d'edition [Kuntz, 1987] ;
  - Si c'est un livre : Les noms des auteurs, suivi du titre du livre, de l'editeur, ISBN/ISSN, et la date d'edition ;
  - Si c'est un article : Les noms des auteurs, le titre , le journal ou la conference, et la date de publication ;
  - Si c'est un site web ou un document electronique : Le titre, le lien et la date de consultation ;
  - Si c'est une these : nom et prenom, titre de la these, universite de soutenance, annee de soutenance, nombre de pages ;
  - Exemples :

    [*Bazin*, 1992] BAZIN R., REGNIER B. Les traitements antiviraux et leurs essais therapeutiques. Rev. Prat., 1992, 42, 2, p.148-153.

    [*Anderson*, 1998] ANDERSON P.JF. Checklist of criteria used for evaluation of metasites. [en ligne]. Universite du Michigan, Etats Unis. Site disponible sur : http ://www.lib.umich.edu/megasite/critlist.html.(Page consultee le 11/09/1998).
  - Dans le texte du rapport, on doit obligatoirement citer la reference en faisant appel e son identifiant, juste apres avoir utilise la citation. Si ceci n'est pas fait dans les regles, on peut etre accuse de plagiat.

Conclusion

Voile.

Chapitre VI

Sprint 4 : Payment Integration

Plan

Introduction

Ce chapitre porte sur la partie pratique ainsi que la bibliographie.

1 Outils et langages utilises

L'etude technique peut se trouver dans cette partie, comme elle peut etre faite en parallele avec l'etude theorique (comme le suggere le modele 2TUP). Dans cette partie, il faut essayer de convaincre le lecteur de vos choix en termes de technologie. Un etat de l'art est souhaite ici, avec un comparatif, une synthese et un choix d'outils, meme tres brefs.

2 Presentation de l'application

Il est tout e fait normal que tout le monde attende cette partie pour coller e souhait toutes les images correspondant aux interfaces diverses de l'application si chere e votre coeur, mais abstenez vous ! Il FAUT mettre des imprime ecrans, mais bien choisis, et surtout, il faut les scenariser : Choisissez un scenario d'execution, par exemple la creation d'un nouveau client, et montrer les differentes interfaces necessaires pour le faire, en expliquant brievement le comportement de l'application. Pas trop d'images, ni trop de commentaires : concis, encore et toujours.

evitez ici de coller du code : personne n'a envie de voir le contenu de vos classes. Mais vous pouvez inserer des snippets (bouts de code) pour montrer certaines fonctionnalites [1][2], si vous en avez vraiment besoin. Si vous voulez montrer une partie de votre code, les etapes d'installation ou de configuration, vous pourrez les mettre dans l'annexe.

2.1 Exemple de tableau

Vous pouvez utiliser une description tabulaire d'une eventuelle comparaison entre les travaux existants. Ceci est un exemple de tableau : Tab VI.1.

Tableau VI.1 – Tableau comparatif

	Col1	**Col2**	**Col3**	**Col4**
Row1		X		
Row2	X			
Row3	X	X	X	X
Row4	X		X	X
Row5	X		X	X
Row6	X		X	X
Row7	X		X	
Row8	X	X	X	

2.2 Exemple de Code

Voici un exemple de code Java, avec coloration syntaxique VI.1.

Listing VI.1 – Helloworld Java

```
public class HelloWorld {
//la methode main
  public static void main(String[] args) {
      System.out.println("Hello, World");
  }

}
```

3 Remarques sur la bibliographie

Votre bibliographie doit repondre e certains criteres, sinon, on vous fera encore et toujours la remarque dessus (et parfois, meme si vous pensez avoir tout fait comme il faut, on peut vous faire la remarque quand meme : chacun a une conception tres personnelle de comment une bibliographie devrait etre).

- Une bibliographie dans un bon rapport doit contenir plus de livres et d'articles que de sites web : apres tout c'est une biblio. Privilegiez donc les ouvrages reconnus et publies pour vos definitions, au lieu de sauter directement sur le premier article wikipedia ;
- Les elements d'une bibliographie sont de preference classes par ordre alphabetique, ou par themes (et ordre alphabetique pour chaque theme) ;
- Une entree bibliographique doit etre sous la forme suivante :
  - Elle doit contenir un identifiant unique : represente soit par un numero [1] ou par le nom du premier auteur, suivi de l'annee d'edition [Kuntz, 1987] ;
  - Si c'est un livre : Les noms des auteurs, suivi du titre du livre, de l'editeur, ISBN/ISSN, et la date d'edition ;
  - Si c'est un article : Les noms des auteurs, le titre , le journal ou la conference, et la date de publication ;
  - Si c'est un site web ou un document electronique : Le titre, le lien et la date de consultation ;
  - Si c'est une these : nom et prenom, titre de la these, universite de soutenance, annee de soutenance, nombre de pages ;
  - Exemples :

    [*Bazin*, 1992] BAZIN R., REGNIER B. Les traitements antiviraux et leurs essais therapeutiques. Rev. Prat., 1992, 42, 2, p.148-153.

    [*Anderson*, 1998] ANDERSON P.JF. Checklist of criteria used for evaluation of metasites. [en ligne]. Universite du Michigan, Etats Unis. Site disponible sur : http ://www.lib.umich.edu/megasite/critlist.html.(Page consultee le 11/09/1998).
  - Dans le texte du rapport, on doit obligatoirement citer la reference en faisant appel e son identifiant, juste apres avoir utilise la citation. Si ceci n'est pas fait dans les regles, on peut etre accuse de plagiat.

Conclusion

Voile.

General Conclusion

C'est l'une des parties les plus importantes et pourtant les plus negligees du rapport. Ce qu'on ne veut pas voir ici, c'est combien ce stage vous a ete benefique, comment il vous a appris e vous integrer, e connaetre le monde du travail, etc.
Franchement, personne n'en a rien e faire, du moins dans cette partie. Pour cela, vous avez les remerciements et les dedicaces, vous pourrez vous y exprimer e souhait.
La conclusion, c'est tres simple : c'est d'abord le resume de ce que vous avez raconte dans le rapport : vous reprenez votre contribution, en y ajoutant ici les outils que vous avez utilise, votre maniere de proceder. Vous pouvez meme mettre les difficultes rencontrees. En deuxieme lieu, on y met les perspectives du travail : ce qu'on pourrait ajouter e votre application, comment on pourrait l'ameliorer.

Bibliographique

[1] Marouane Elkamel. *Flouci developers api*. Math-info Department (2019). 15, 19, 23, 27

[2] Mr. Latex. Debuter avec Latex. www.latex.com, (2008). [En ligne ; consulte le 19-Juillet-2008]. 15, 19, 23, 27

Annexe : Remarques Diverses

— Un rapport doit toujours etre bien numerote ;
— De preference, ne pas utiliser plus que deux couleurs, ni un caractere fantaisiste ;
— Essayer de toujours garder votre rapport sobre et professionnel ;
— Ne jamais utiliser de je ni de on, mais toujours le nous (meme si tu as tout fait tout seul) ;
— Si on n'a pas de paragraphe 1.2, ne pas mettre de 1.1 ;
— TOUJOURS, TOUJOURS faire relire votre rapport e quelqu'un d'autre (de preference qui n'est pas du domaine) pour vous corriger les fautes d'orthographe et de franeais ;
— Toujours valoriser votre travail : votre contribution doit etre bien claire et mise en evidence ;
— Dans chaque chapitre, on doit trouver une introduction et une conclusion ;
— Ayez toujours un fil conducteur dans votre rapport. Il faut que le lecteur suive un raisonnement bien clair, et trouve la relation entre les differentes parties ;
— Il faut toujours que les abreviations soient definies au moins la premiere fois oe elles sont utilisees. Si vous en avez beaucoup, utilisez un glossaire.
— Vous avez tendance, en decrivant l'environnement materiel, e parler de votre ordinateur, sur lequel vous avez developpe : ceci est inutile. Dans cette partie, on ne cite que le materiel qui a une influence sur votre application. Que vous l'ayez developpe sur Windows Vista ou sur Ubuntu n'a aucune importance ;
— Ne jamais mettre de titres en fin de page ;
— Essayer toujours d'utiliser des termes franeais, et eviter l'anglicisme. Si certains termes sont plus connus en anglais, donner leur equivalent en franeais la premiere fois que vous les utilisez, puis utilisez le mot anglais, mais en italique ;
— eviter les phrases trop longues : clair et concis, c'est la regle generale !

Rappelez vous que votre rapport est le visage de votre travail : un mauvais rapport peut eclipser de l'excellent travail. Alors pretez-y l'attention necessaire.