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...

Acknowledgments

This work would not have been possible without the valuable cooperation of a number of people I want to pay tribute to.

I would like to thank all those who have made this internship a rewarding and enjoyable experience, especially :

Mr Nebras JEMAL, co-founder and CEO of Flouci for believing in me from day one, my internship tutor **Mr Anis KALLEL** co-founder and CTO who introduced me to the team and helped me throughout my journey. Thank you for your advice and help throughout the internship.

The **"Flouci"** team, my second family, for their support, advice and all their devotion and passion to what we are doing.

I would particularly like to thank my supervisor **Mr Aymen SELLOAUTI** for he availability, remarks and advice. I also would like to express my respect and my gratitude to him...

Finally, I also express my sincere recognition to the members of the jury : **Mr Flen** and **Mr Flen** for accepting to evaluate my work.

Table of Contents

List of Figures vii

List of Tables viii

Summary ix

General Introduction 1

I Project Scope 2

1 Host company presentation . . . 2

1.1 Presentation of Kaoun . . . 2

1.2 Presentation of Flouci . . . 3

2 Project overview . . . 4

2.1 Project Context . . . 4

2.2 Study of the existing . . . 4

2.3 Project goals . . . 6

3 Methodology . . . 7

3.1 Agile methodology . . . 7

3.2 Kanban methodology . . . 8

3.3 Lean software development . . . 9

II Project Development Practices 10

1 Kanban Tickets Life Cycle . . . 11

1.1 Backlog . . . 11

1.2 Selected For Development . . . 11

1.3 In Progress . . . 12

1.4 Code Review . . . 12

1.5 QA . . . 12

1.6 Done . . . 12

2 Gitflow workflow . . . 13

2.1 Feature Branch . . . 13

2.2 Develop Branch . . . 13

2.3 Master Branch . . . 13

2.4 Release Branch . . . 13

2.5 Hotfix Branch . . . 14
3 Test-driven development . . . 14
4 DevOps . . . 15
4.1 DevOps Steps . . . 16
4.1.1 Preparation . . . 16
4.1.2 Testing . . . 16
4.1.3 Quality Measurements . . . 16
4.1.4 Dockerization . . . 16
4.1.5 Deployment . . . 16
5 Tools . . . 17
5.1 Bitbucket . . . 17
5.2 GitKraken Glo Boards . . . 17
5.3 Docker . . . 17
5.4 Docker Swarm . . . 17
5.5 Jenkins . . . 18
5.6 SonarQube . . . 18

III Sprint 1 : Project Launch **19**
1 Outils et langages utilises . . . 19
2 Presentation de l'application . . . 19
2.1 Exemple de tableau . . . 20
2.2 Exemple de Code . . . 20
3 Remarques sur la bibliographie . . . 21

IV Sprint 2 : User and App Managment **23**
1 Outils et langages utilises . . . 23
2 Presentation de l'application . . . 23
2.1 Exemple de tableau . . . 24
2.2 Exemple de Code . . . 24
3 Remarques sur la bibliographie . . . 25

V Sprint 3 : Checkout API **27**
1 Outils et langages utilises . . . 27
2 Presentation de l'application . . . 27
2.1 Exemple de tableau . . . 28
2.2 Exemple de Code . . . 28

3 Remarques sur la bibliographie 29

VI Sprint 4 : Payment Integration **31**

1 Outils et langages utilises 31

2 Presentation de l'application 31

2.1 Exemple de tableau 32

2.2 Exemple de Code 32

3 Remarques sur la bibliographie 33

General Conclusion **35**

Bibliographique **36**

Annexe : Remarques Diverses **37**

List of Figures

1 APIs 1

I.1 Kaoun logo 3
I.2 Flouci logo 4
I.3 Stripe Api 5
I.4 Stripe Api 6
I.5 Twint Payment Method 6

II.1 TDD Steps 15

List of Tables

III.1 Tableau comparatif 20

IV.1 Tableau comparatif 24

V.1 Tableau comparatif 28

VI.1 Tableau comparatif 32

Summary

Brief summary ...

General Introduction

In the modern software world APIs are becoming a must for every tech company to allow products to be integrated by developers in multiple Apps.

APIs do all this by "exposing" some of a product's internal functions to the outside world in a limited fashion. That makes it possible for applications to share data and take actions on one another's behalf without requiring developers to share all of their software's code.

Figure 1 – APIs

That's the reason why KAOUN, a Tunisian FinTech start-up, decided to create its own developers API for its main product FLOUCI which is a mobile payment solution. This project will expand FLOUCI to the developers world and unleash the full potential of the product, also it could take FLOUCI into new markets and open doors for unlimited integrations.

The following report is a synthesis of the efforts done to build FLOUCI developers API.

To detail the process of our work, we have divided this report into XX chapters representing the different aspects of our project.

In the first chapter, entitled Project scope, we started with a presentation of the host company. Afterwards, we gave an overview of our project and we detailed the followed methodology for its realization.

The second chapter is about the development disciplines and rules we set during our project development life cycle. It's a detailed explanation of the development practices needed to start our project development in the most efficient way possible.

We close our work with a general conclusion in which we try to evaluate our contribution as well as we develop our vision for the project's potential improvements.

Chapitre I

Project Scope

Plan

1 Host company presentation 2

1.1 Presentation of Kaoun 2

1.2 Presentation of Flouci 3

2 Project overview 4

2.1 Project Context 4

2.2 Study of the existing 4

2.3 Project goals 6

3 Methodology 7

3.1 Agile methodology 7

3.2 Kanban methodology 8

3.3 Lean software development 9

Introduction

This chapter is dedicated to the presentation of our project's general scope. This will include an introduction of the host company Kaoun and it's main product Flouci. Also we will give an overview of the developer API project. After that, we will describe the chosen methodology that we followed during the realization of our project.

1 Host company presentation

In The first section of the report, we will introduce Kaoun, The host company that made the project possible.

1.1 Presentation of Kaoun

Kaoun is a new FinTech company that builds reliable infrastructure for payments and credits in Tunisia, and whose mission is to enable all individuals and businesses to access

financial services using any phone, anywhere, anytime.

Kaoun's first product, Flouci, is a mobile and web payment application built on top of a unique decentralized inter and intrabank infrastructure that allows instant transactions for peer to peer transfers and merchant payments. Kaoun plans to work with governments, traditional banks, mobile operators, and microfinance institutions to fix the lag between technology adoption and financial inclusion by reducing the barriers to entry for the unbanked and the underbanked.

Figure I.1 – Kaoun logo

1.2 Presentation of Flouci

Flouci is the first wallet designed to innovate mobile payment in Tunisia. It serves as a quick, easy, and convenient way to open a bank account, send and receive money, and pay different merchants in-person or online, all from within the app.

— **Open an account :**
In order to create a Flouci account, you either link your Flouci wallet to an existing bank account or follow the step-by-step KYC (Know Your Customer) guide to create an account with one of our partner financial institutions. Once you have your wallet and your QR code, you can start sending and receiving money and paying merchants using your phone.

— **Send and receive money :**
Once you create a Flouci account, you can send and receive money to and from anybody that has a flouci account/wallet. It's easy, cheap, and secure. And the best part is that it's practically instant. All you have to do is enter their number or scan their personal QR code to access their profile. You then enter the amount and confirm. You both then receive a confirmation SMS and you're done.

— **Pay merchants :**
Through Flouci, you have access to a wide range of partner merchants across Tunisia.

You can pay through the app by just scanning the QR code shown on the counter of the merchant. No waiting lines, no more looking for change or realizing you forgot your wallet at home.

Figure I.2 – Flouci logo

2 Project overview

In this section, we will present the developer API project context, we will analyze some of the existing mobile payments API's. And finally, we will indicate our project goals.

2.1 Project Context

Flouci in it's first version made it possible for users to pay merchants in simple steps and without the need of cash. However the market is rapidly shifting toward online e-commerce sites with companies like Jumia, Tayara and many other introducing their solutions in Tunisia.

In it's current implementation Flouci is not able to integrate into any form of online payments due to the lack of it's implementation.

Facing this problem and a fast moving e-commerce market the company had to move toward implementing a solution for developers.

The API should make it possible for developers to link their Flouci wallets to the their e-commerce sites and add flouci as an easy and instant payment method.

2.2 Study of the existing

In The world of e-commerce a lot of payments methods exists. Most of the methods are credit card related and offer the possibility to pay using the card number and the 3 digits code.

In our case Flouci offer payments through QR codes scans. Although the payment steps on the user side is different the developer API should offer similar functionalities.

Below is a list of world leader online payment API's :

— **Stripe :**
The Stripe API is organized around REST. The API has predictable resource-oriented URLs, accepts form-encoded request bodies, returns JSON-encoded responses, and uses standard HTTP response codes, authentication, and verbs.
You can use the Stripe API in test mode, which does not affect your live data or interact with the banking networks. The API key you use to authenticate the request determines whether the request is live mode or test mode.

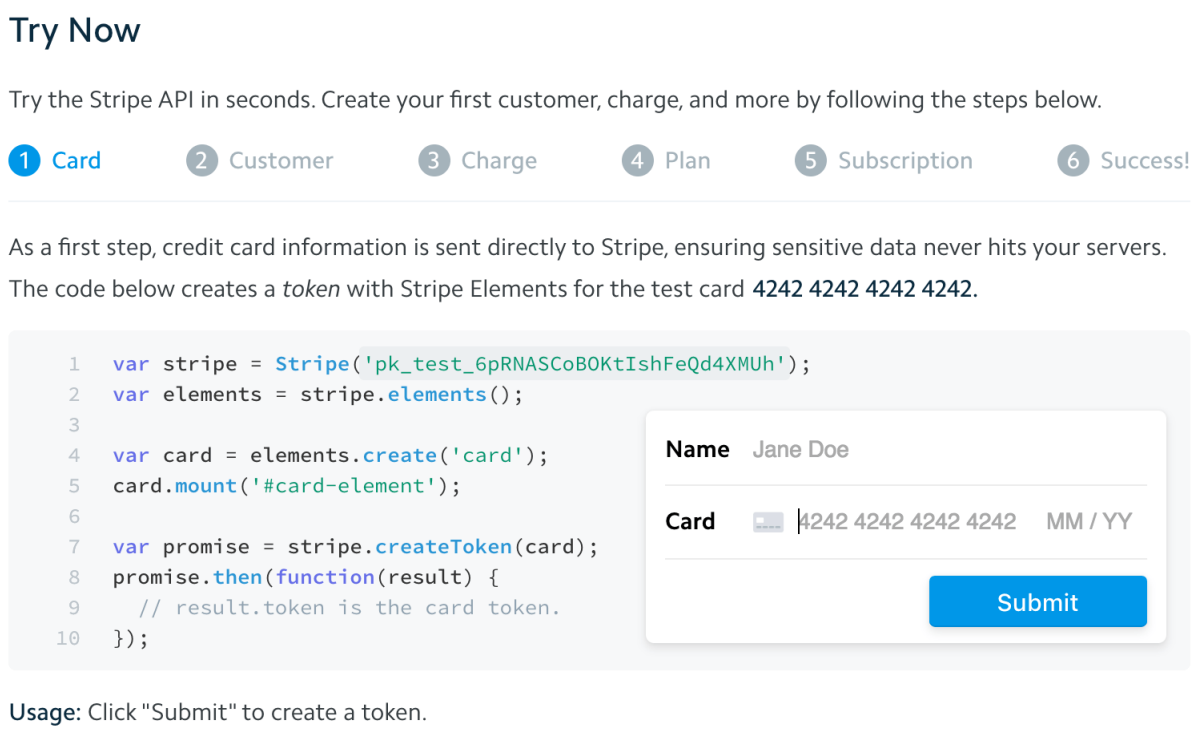

Figure I.3 – Stripe Api

— **Paypal :**
The PayPal APIs are HTTP-based RESTful APIs that use OAuth 2.0 for authorization. API request and response bodies are formatted in JSON.

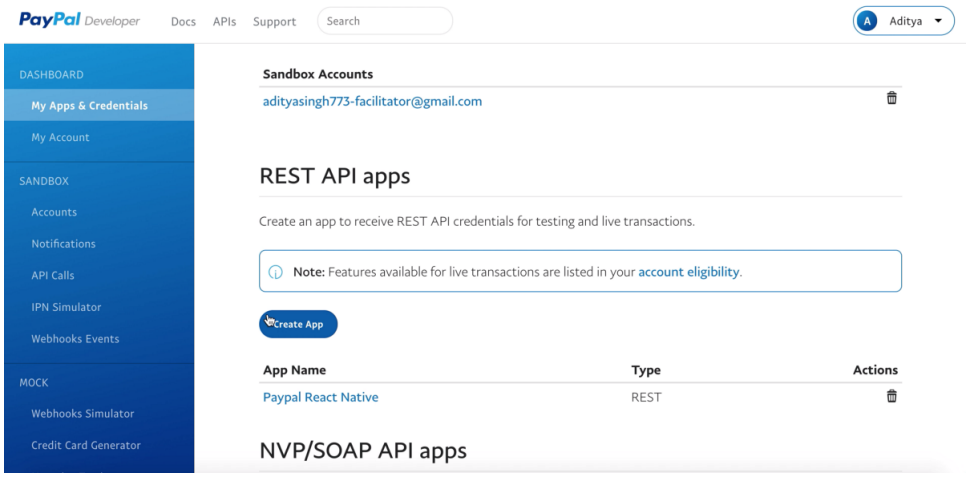

Figure I.4 – Stripe Api

— **Twint :**
Twint is the closest implementation to Flouci, it offers payments through QR code scans. The plugin allows QR code generation on the web page, it also creates a code for each transaction, It serves to confirm payments.

Figure I.5 – Twint Payment Method

2.3 Project goals

To bring developers to the Flouci world and allow e-commerce owners to introduce a mobile payment solution Kaoun has decided to build it's own developer API from scratch and provide an easy way to accept payments in few simple steps. This presented the opportunity for us to turn this problem into the main objective of our graduation project.

By the end of our project, we need to achieve these goals :

— **Create Account :**
Any developer should be able to create a Flouci developer account from the web platform, basic informations are needed to open an account.
Also it should be possible to use an existing Flouci account and switch it to developer mode.

— **Create App and link Flouci wallet :**
The web interface should offer the possibility to create an App and link an existing wallet to the specific configuration. A two steps verification system should be put in place to verify ownership of the wallet. To verify transaction developer could choose between two modes :
  — **Active mode :** Activate an endpoint to verify transactions by id.
  — **Passive mode :** Configure a webhook to receive transactions info once validated.
A unique token is generated for each app to allow client integration.

— **Integrate Flouci client :**
Once the App is configured the developer can easily add Flouci as a payment method with the token provided.

— **Check App analytics :**
Every App should enable a dashboard for the developer to monitor sales and check a set of KPI's.

3 Methodology

In this section, we will go through the importance of having a fixed methodology in a software development project, as well as our choice for this project and the reasons behind it.

3.1 Agile methodology

In every professional IT project, it is essential to adopt a methodology of work in order to guarantee a good organization of tasks and to define the different phases through which the project passes. Since their appearance, agile development methods have considerably improved the quality of the software and have reduced their production time. Thanks to its nature, incremental and collaborative, such a methodology allows to take into account the evolution of the customer's needs. Agile methodologies are based on four main values :

— They promote interaction between the various parties involved in the project by in relation to processes and tools.

— They replace exhaustive documentation with concrete functionalities.
— The client participates in the project throughout its implementation instead of defining contracts that formalize the relationship with the client.
— It is necessary to accept the likely changes during the implementation process.

We chose the Kanban methodology for our project for three reasons :

— Visually see work in progress.
— Empower teams to self-manage visual processes and workflows.
— Kanban boards can easily be managed on different free tools, like Trello, MeisterTask or GitKraken which we will be using in our project.

3.2 Kanban methodology

In general, Kanban is a scheduling system for lean and other JIT processes. In a Kanban process, there are physical (or virtual) "cards" called Kanban that move through the process from start to finish. The aim is to keep a constant flow of Kanban so that as inventory is required at the end of the process, just that much is created at the start.

When used for software development, Kanban uses the stages in the software development lifecycle (SDLC) to represent the different stages in the manufacturing process. The aim is to control and manage the flow of features (represented by Kanban cards) so that the number of features entering the process matches those being completed.

Kanban is an agile methodology that is not necessarily iterative. Processes like Scrum have short iterations which mimic a project lifecycle on a small scale, having a distinct beginning and end for each iteration. Kanban allows the software be developed in one large development cycle. Despite this, Kanban is an example of an agile methodology because it fulfils all twelve of the principles behind the Agile manifesto, because whilst it is not iterative, it is incremental.

The principle behind Kanban that allows it to be incremental and Agile, is limited throughput. With no iterations a Kanban project has no defined start or end points for individual work items ; each can start and end independently from one another, and work items have no pre-determined duration for that matter. Instead, each phase of the lifecycle is recognized as having a limited capacity for work at any one time. A small work item is created from the prioritized and unstarted requirements list and then begins the development process, usually with some requirements elaboration. A work item is not allowed to move on to the next phase until some capacity opens up ahead. By controlling the number of tasks active at any one time, developers still approach the overall project incrementally which gives the opportunity for Agile principles to be applied.

Kanban projects have Work In Progress (WIP) limits which are the measure of capacity

that keeps the development team focused on only a small amount of work at one time. It is only as tasks are completed that new tasks are pulled into the cycle. WIP limits should be fine-tuned based on comparisons of expected versus actual effort for tasks that complete.

Kanban does not impose any role definition as say, Scrum does and along with the absence of formal iterations, role flexibility makes Kanban attractive to those who have been using waterfall-style development models and want to change but are afraid of the initial upheaval something like Scrum can cause while being adopted by a development team.

3.3 Lean software development

Lean Software Development is a set of principles to deliver software according to the principles of lean manufacturing. In a lean environment, activities or processes that result in the expenditure of effort and/or resources towards goals that are not producing value for the customer should be eliminated. Essentially, lean is centered on preserving value with less work. Lean approaches are often called six-sigma or Just-In Time (JIT).

Conclusion

This first chapter allowed us to define the general boundaries of our project.

We gave an introduction of our host company and the motivations behind the project. We studied the project context and the existing similar projects, we took a look into the three biggest implementations.

In The end we set the basis of the project by fixing a methodology to follow in the development and realization of the project.

Chapitre II

Project Development Practices

Plan

1 Kanban Tickets Life Cycle 11

1.1 Backlog 11

1.2 Selected For Development 11

1.3 In Progress 12

1.4 Code Review 12

1.5 QA 12

1.6 Done 12

2 Gitflow workflow 13

2.1 Feature Branch 13

2.2 Develop Branch 13

2.3 Master Branch 13

2.4 Release Branch 13

2.5 Hotfix Branch 14

3 Test-driven development 14

4 DevOps 15

4.1 DevOps Steps 16

5 Tools 17

5.1 Bitbucket 17

5.2 GitKraken Glo Boards 17

5.3 Docker 17

5.4 Docker Swarm 17

5.5 Jenkins 18

5.6 SonarQube 18

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.

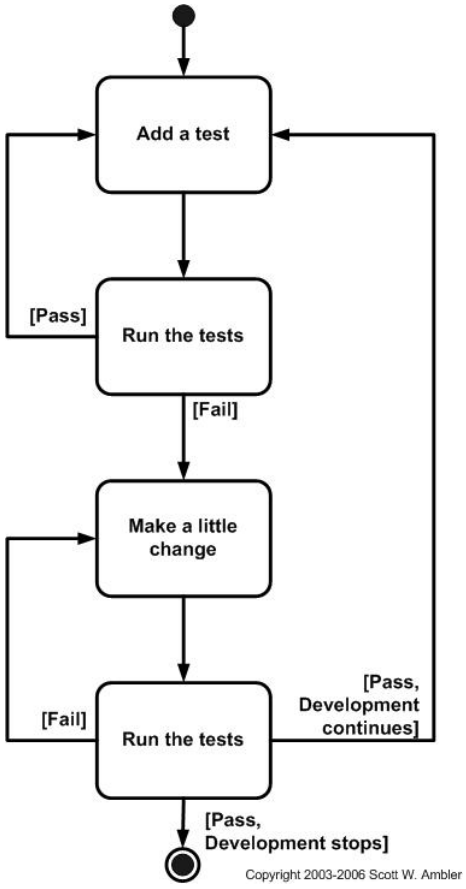

Figure II.1 – TDD Steps

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

1 Outils et langages utilises 19

2 Presentation de l'application 19

2.1 Exemple de tableau 20

2.2 Exemple de Code 20

3 Remarques sur la bibliographie 21

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

1 Outils et langages utilises 23

2 Presentation de l'application 23

2.1 Exemple de tableau 24

2.2 Exemple de Code 24

3 Remarques sur la bibliographie 25

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

1 Outils et langages utilises 27

2 Presentation de l'application 27

2.1 Exemple de tableau 28

2.2 Exemple de Code 28

3 Remarques sur la bibliographie 29

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

1 Outils et langages utilises 31

2 Presentation de l'application 31

2.1 Exemple de tableau 32

2.2 Exemple de Code 32

3 Remarques sur la bibliographie 33

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). 20, 24, 28, 32

[2] Mr. Latex. Debuter avec Latex. www.latex.com, (2008). [En ligne ; consulte le 19-Juillet-2008]. 20, 24, 28, 32

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.