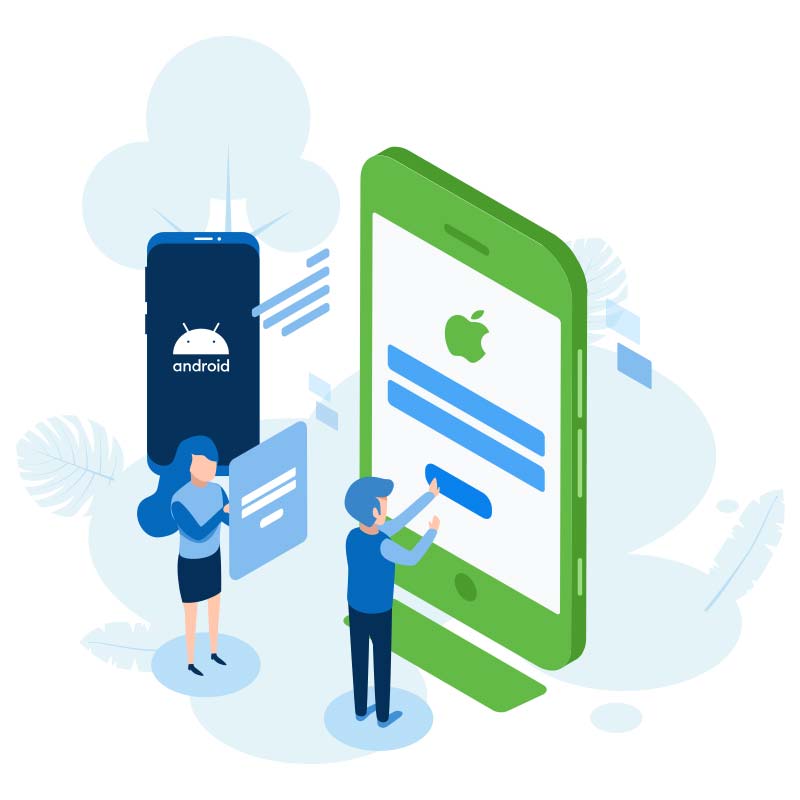
|  |
| --- |
| Développement d’une application mobile multiplateforme |



Koetschet Thierry

Chemin du Perrey 22

1670 Ursy

thierry.koetschet.1998@gmail.com



**SI-CA2a**

08.02.2023

Table des matières

[1 Analyse préliminaire 5](#_Toc130897337)

[1.1 Introduction 5](#_Toc130897338)

[1.2 Organisation 6](#_Toc130897339)

[1.3 Objectifs 6](#_Toc130897340)

[1.4 Planification initiale 7](#_Toc130897341)

[1.5 Structure du dossier 8](#_Toc130897342)

[2 Analyse / Conception 8](#_Toc130897343)

[2.1 Description des frameworks 8](#_Toc130897344)

[2.1.1 Apache Cordova 8](#_Toc130897345)

[2.1.2 React Native 9](#_Toc130897346)

[2.1.3 NativeScript 9](#_Toc130897347)

[2.1.4 Flutter 9](#_Toc130897348)

[2.2 Analyse des frameworks 9](#_Toc130897349)

[2.3 Choix du framework 10](#_Toc130897350)

[2.4 Concept 11](#_Toc130897351)

[2.5 Stratégie de test 11](#_Toc130897352)

[2.6 Risques techniques 12](#_Toc130897353)

[2.7 Planification 12](#_Toc130897354)

[2.8 Dossier de conception 13](#_Toc130897355)

[3 Réalisation 14](#_Toc130897356)

[3.1 Dossier de réalisation 14](#_Toc130897357)

[3.2 Installation du framework 15](#_Toc130897358)

[3.3 Installation de l’IDE 15](#_Toc130897359)

[3.4 Description des tests effectués 15](#_Toc130897360)

[3.5 Erreurs restantes 15](#_Toc130897361)

[3.6 Liste des documents fournis 15](#_Toc130897362)

[4 Conclusions 16](#_Toc130897363)

[5 Annexes 17](#_Toc130897364)

[5.1 Cahier des charges 17](#_Toc130897365)

[5.2 Test effectués 20](#_Toc130897373)

[5.3 Résumé du travail 20](#_Toc130897374)

[5.4 Sources – Bibliographie 20](#_Toc130897375)

[5.5 Journal de travail 20](#_Toc130897376)

[5.6 Glossaire 21](#_Toc130897377)

[5.7 Table des illustrations 21](#_Toc130897378)

[5.8 Manuel d'installation 21](#_Toc130897379)

[5.9 Manuel d'utilisation 21](#_Toc130897380)

[5.10 Archives du projet 21](#_Toc130897381)

# Analyse préliminaire

## Introduction

Le but ce travail consiste tout d’abord à rechercher différents frameworks permettant de faire du développement d’application mobile, puis de les analyser et enfin de les tester afin de choisir l’outil le plus pertinent selon mes critères. Ce choix est important car le framework que je choisirai sera utilisé pour réaliser mon TPI.

Dans un deuxième temps, je programmerai une petite application avec le framework choisi afin de me préparer pour l’application que je développerai pour le TPI. L’idée de cette application est de permettre aux habitants de Sainte-Croix de commander des repas dans les différents restaurants de la région et de se les faire livrer à domicile. Cette application s’inspirera des géants du domaine tels que Uber eats, Just Eat ou encore Deliveroo.

La raison de mon choix de développer une application mobile s’explique par le fait que je suis très intéressé par la programmation mais j’avais également envie de faire quelque chose de nouveau, que l’on n’a pas vu en cours. J’ai alors réalisé que de créer une application mobile remplissait les deux conditions et j’ai donc rempli ma fiche signalétique en conséquence.

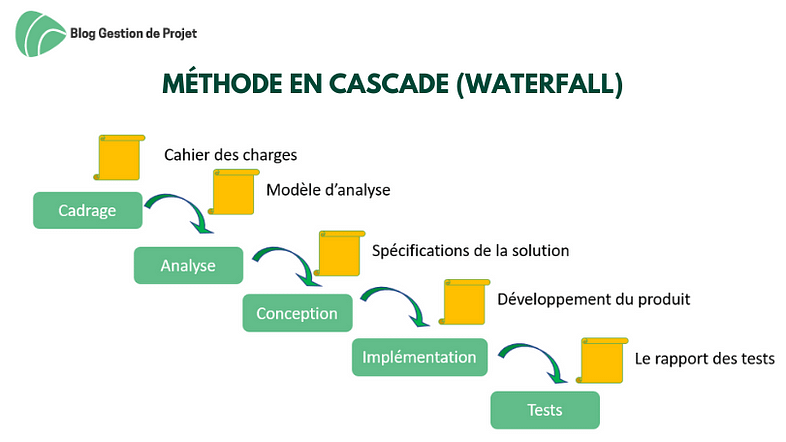
*Ce chapitre décrit brièvement le projet, les raisons de ce choix et ce qu'il peut apporter à l'élève ou à l'école. Il n'est pas nécessaire de rentrer dans les détails (ceux-ci seront abordés plus loin) mais cela doit être aussi clair et complet que possible (idées de solutions). Ce chapitre contient également l'inventaire et la description des travaux qui auraient déjà été effectués pour ce projet.*

## Organisation

L’organisation de mon projet c’est fait de la manière suivante : J’ai tout d’abord partager mon travail en use cases pour définir les quatre majeures parties. La première partie concerne tout ce qui s’apparente à la l’analyse, les tests et le choix entre les frameworks pour continuer la suite du projet. Le deuxième use case englobe toutes les tâches relevant de la modélisation de la base de données. Le troisième concerne la partie pratique du projet, c’est-à-dire la création et le développement de l’application mobile. Enfin le dernier use case va composer du travail de rédaction de la documentation, de mon organisation personnelle et de la partie présentation du projet.

## Méthode de gestion de projet

J’ai choisi la méthode en cascade pour la gestion de mon projet. Je trouve que pour un projet en solo, cette méthode répond parfaitement à mes besoins en terme de gestion de projet.



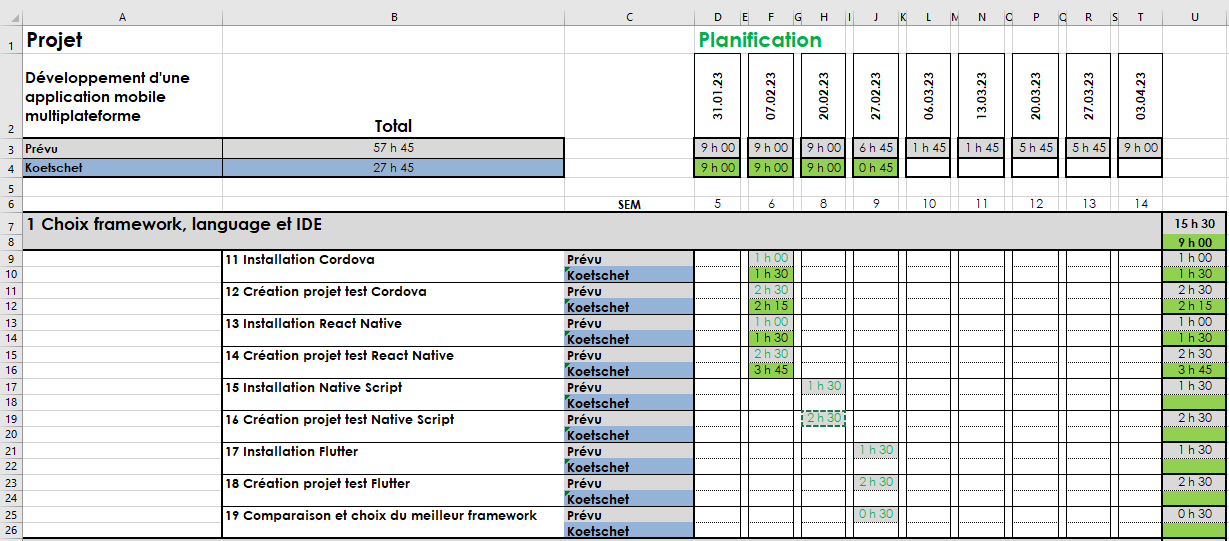
Les cinq étapes de la méthode en cascade correspondent parfaitement à la vision que j’ai pour la réalisation d’un tel projet. En effet, le cadrage en début de projet est primordial afin de structurer notre travail et de ne pas trop s’éparpiller. La partie analyse correspond à ce que j’ai fait en me documentant sur les différents frameworks. Après la partie analyse, j’ai réalisé la conception et l’implémentation de mon application et terminé mon projet par la rédaction de la méthode de tests et réalisation des différents tests par rapport à l’installation du framework et le fonctionnement de la l’application.

## Objectifs

Il est important pour moi d’atteindre certains objectifs lors du pré-TPI : j’aimerais tout d’abord choisir un framework avec lequel je suis assez familier pour pouvoir développer une application mobile d’une certaine envergure. Afin de me familiariser avec le framework en question, j’aimerais programmer une plus petite application me permettant de mieux comprendre le langage de programmation, le fonctionnement du framework et de l’IDE et enfin de pouvoir tester le produit fini.

J’aimerais également acquérir certaines capacités par rapport à la planification et l’organisation d’un tel travail mais aussi concernant la rédaction d’une documentation conforme aux attentes de l’école et du monde professionnel plus généralement.

## Planification initiale



Le cadre bleu représente les use cases de mon projet. Le cadre rouge montre les tâches à réaliser. Les cadres jaunes indiquent le temps planifier pour chaque tâche et les valeurs temporelles surlignées en vert représente le temps réel utilisé pour chaque tâche. Le reste du tableau contient des formules afin de représenter de manière plus visuelle les différentes données saisies.

Le document complet de planification sera remis en annexe à cette documentation.

## Structure du dossier

Ce dossier est composé de 4 parties principales.

La première partie représente l’analyse préliminaire de ce travail et se compose d’une introduction, d’une explication de l’organisation du projet, une description mes objectifs personnels et puis un bref aperçu de ma planification initiale.

La deuxième se compose de toute la partie analyse et conception du projet avec l’élaboration du concept, de la stratégie de test, un compte rendu des risques techniques, une révision de la planification initiale du projet et enfin la partie conception du projet.

La troisième partie représente toute la réalisation pratique du projet commençant par lister tous les fichiers du dossier de réalisation, puis une description des tests effectués et des erreurs restantes et finalement une énumération des documents fournis à la remise du projet.

La dernière décrit les conclusions auxquelles je suis arrivé à la fin de ce projet telles que les objectifs atteints ou non, mon ressenti au fil du projet, les difficultés rencontrées et les améliorations que je pourrais apporter si je devais refaire un tel projet.

# Analyse / Conception

## Description des frameworks

Un framework est un ensemble de composants logiciels servant à gérer et créer la structure et le corps d’une application. Les différents framework analysés dans ce travail sont spécifiquement utiles à la création d’applications mobiles. Cette analyse, en plus des différents tests effectués sur les frameworks, va permettre faire un choix qui semble le plus adéquat pour la réalisation d’une petite application mobile.

### Apache Cordova

Cordova est un framework gratuit et open-source permettant la création d’application mobile multiplateforme avec un seul code de base composé de HTML, CSS et JavaScript. Les premières version d’Apache Cordova en 2011 sous le nom de Fondation Apache, puis Apache PhoneGap.

### React Native

ReactNative est tout comme Cordova un framework open-source créer par Meta Platforms (anciennement Facebook). Il permet le développement de logiciel pour plateforme mobile pour des systèmes d’exploitation tels qu’Android ou iOS.

### NativeScript

Le framework NativeScript a été créé et mis à disposition par Progress gratuitement. Il sert à coder des applications mobiles avec le langage de programmation JavaScript ou des langages dérivés tels que TypeScript. La particularité de ce framework est qu’il contient une multitude d’API natifs, c’est-à-dire propre à NativeScript

### Flutter

Le dernier framework que j’ai testé est Flutter. Il s’agit d’un SDK open-source créé et mis à disposition par Google pour le développement d’applications Android, iOS, Linux, MacOS, Windows, etc. Contrairement aux autres frameworks, Flutter possède son propre langage de programmation : Dart.

## Analyse des frameworks

Au début de ce projet, j’ai décidé de commencer par Apache Cordova. J’ai réalisé l’installation en suivant le guide d’installation mis à disposition sur le site officiel d’Apache Cordova[[1]](#footnote-1). Les premiers problèmes sont apparus lors de l’installation de l’environnement de développement Android Studio. En effet, j’ai rencontré à ce moment des soucis pour créer un AVD nécessaire pour l’affichage de l’application. L’interface de gestion des appareils virtuels me refusait catégoriquement de créer un émulateur. Je me suis rendu compte plus tard dans mon travail en installant d’autres frameworks que le guide d’installation d’Apache Cordova manquait de détails au niveau de la modification des variables d’environnement et que cela en était certainement la cause. De plus, le build de l’application a tout simplement échoué sans pour autant que je ne comprenne la raison, dû certainement à mon manque d’expérience dans le domaine.

J’ai donc décidé de tester l’installation d’un autre framework, React Native. J’ai également suivi la documentation d’installation de l’environnement disponible sur le site de React Native[[2]](#footnote-2). J’ai alors rencontré les mêmes problèmes qu’avec Cordova. J’ai donc essayé de suivre un tutoriel sur Youtube[[3]](#footnote-3). Grâce à la vidéo, je me suis rendu compte de deux problèmes principaux dans mon setup : le premier était que je n’avais pas ajouter de variables d’environnement ce qui empêchait le framework de trouver tous les chemins nécessaires pour fonctionner correctement. Et en configurant ces variables d’environnement, j’ai rencontré un deuxième problème. En effet, j’ai remarqué que le dossier Sdk, donc le dossier contenant le kit de développement de software Android ne situait pas dans le même dossier que l’IDE Android Studio ce qui rendait la configuration des variables d’environnement relativement compliquée.

La décision qui me semblait alors la plus logique était de refaire une installation de zéro en prenant soin de mettre tous les dossiers concernés dans un même dossier appelé Android à la racine de mon C:\. Pour ce faire, j’ai donc désinstallé Android Studio et supprimé tous les dossiers restants contenant le Android dans leur nom. Puis j’ai refait toute l’installation en faisant attention à ce que tous les dossiers aient la même racine. Après toute cette procédure, j’ai refait des tests avec React Native. Cette fois-ci, j’ai réussi à créer un AVD et à construire une application vide avec React Native.

J’ai décidé de poursuivre mon analyse et mes tests sur le framework NativeScript[[4]](#footnote-4). Son installation est similaire aux autres frameworks avec le téléchargement de la librairie, le setup de l’IDE et du SDK qui a déjà été fait précédemment. Après avoir téléchargé la librairie NativeScript avec la commande **npm install –g nativescript,** une autre commande très utile montre un diagnostic très complet des éléments que NativeScript arrive à trouver et les erreurs qu’il rencontre. Cette commande est **ns doctor android** et elle est une des raisons qui a influencé mon choix. Une fois que le diagnostic est totalement fonctionnel, j’ai pu créer, construire et lancer mon application test.

Le dernier framework que j’ai décidé de tester est Flutter[[5]](#footnote-5). Tout comme NativeScript, son installation a été relativement simplifiée par toutes les étapes d’installation de l’IDE et du SDK fait au début du projet. J’ai pu créer une application test et la lancer sur l’émulateur Android et elle fonctionnait bien malgré quelques latences au niveau de la machine hôte probablement dues au grand nombre de ressources demandé par l’AVD. La seule raison qui a poussé mon choix sur NativeScript est que le langage de programmation avec Flutter est complètement nouveau pour moi et j’ai préféré me diriger sur quelque chose que je connais déjà avec du JavaScript.

## Choix du framework

Comme je l’ai déjà un peu abordé au chapitre précédent, j’ai choisi NativeScript comme framework pour programmer mon application.

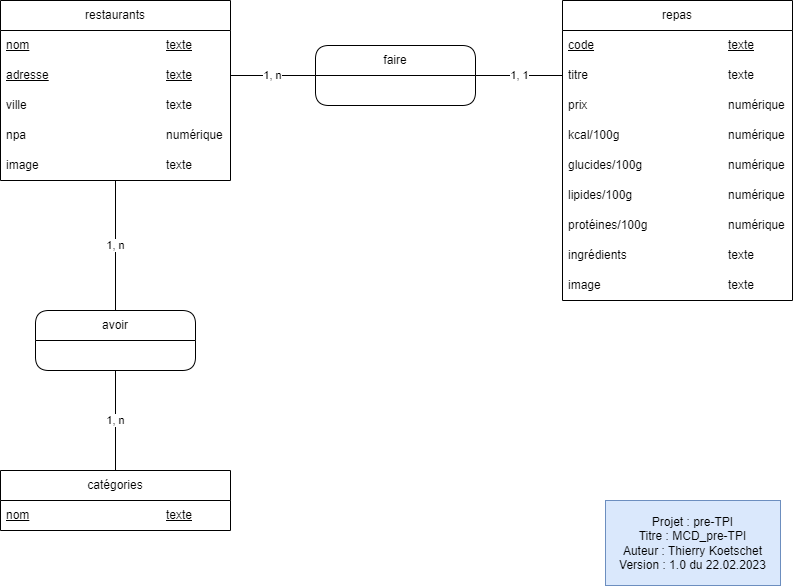
La première raison s’explique par le fait qu’il s’agit du framework que j’ai eu le moins de problèmes à installer et à configurer. En effet, sa documentation était particulièrement détaillée par rapport aux autres et comme je l’ai expliqué, la commande **ns doctor android** a rendu le debug de l’installation beaucoup plus aisé.

Deuxièmement, JavaScript est un langage de programmation que je connais assez bien grâce au cours ICT-120 ou encore ICT-306 reçus pendant ma première année de formation.

Finalement, NativeScript a répondu à toutes les attentes que j’avais pour la réalisation de mon programme : créer une application multiplateforme, visualiser des tutoriels compréhensibles et disposer d’une documentation d’installation complète et fonctionnelle.

## Concept

### Modèle conceptuel de données



Le MCD a été réalisé avec l’application draw.io

### Modèle logique de données

Le MLD a été réalisé avec MySQL Workbench 8.0 CE

Celui-ci a permis de générer le script de création de la base de données que j’ai exécuté sur HeidiSQL.

## Stratégie de test

Dans un premier temps, les tests effectués vont me permettre vérifier si l’installation des frameworks est fonctionnelle.

Ma stratégie de test va d’abord se composer de test unitaires. Ceux-ci vont me permettre de vérifier l’état des composants nécessaires et au fonctionnement du framework :

* Le Software Developpement Kit
* L’environnement de développement
* Les variables d’environnement
* L’émulateur

Ces composants sont communs aux quatre framework mais d’autres composants sont propres à chaque framework, et nécessaire pour leur bon fonctionnement, qu’il faudra aussi pouvoir tester.

Si chaque test unitaire est concluant, je pourrai passer au tests d’intégration. Ceux-ci permettront de contrôler si tous les modules testés individuellement fonctionnent ensemble.

Enfin, je terminerai la réalisation de mes tests par les tests fonctionnels. Ils me permettront de déterminer si les frameworks installés et mon application mobile fonctionnent correctement.

## Risques techniques

Le principal risque technique que j’ai rencontré est le manque de temps. Plus le projet avançait, plus je réalisais que l’installation de framework était chronophage et que j’allais vraisemblablement manqué de temps à la fin du projet. Malheureusement, entre la partie analyse, la partie pratique et la rédaction de la documentation, j’ai décidé de coupé court à la partie pratique afin d’avoir assez de temps pour rendre un document convenable.

Le deuxième risque technique auquel je me suis heurté est mon manque de compétence dans le domaine du développement d’applications mobiles. En effet, l’ajout d’un framework à un projet de développement apporte une toute nouvelle dimension que je n’ai jamais eu l’occasion d’apprendre dans ma formation. C’est ce nouvel aspect qui m’attirait au début du projet mais je me suis vite rendu compte que j’ai été un peu trop ambitieux peut-être que j’aurais sûrement dû me contenter des connaissances acquises en cours afin de garder un cadre connu autour de mon projet.

## Planification

*Révision de la planification initiale du projet :*

* *planning indiquant les dates de début et de fin du projet ainsi que le découpage connu des diverses phases.*
* *partage des tâches en cas de travail à plusieurs.*

*Il s’agit en principe de la planification* ***définitive du projet****. Elle peut être ensuite affinée (découpage des tâches). Si les délais doivent être ensuite modifiés, le responsable de projet doit être avisé, et les raisons doivent être expliquées dans l’historique.*

## Dossier de conception

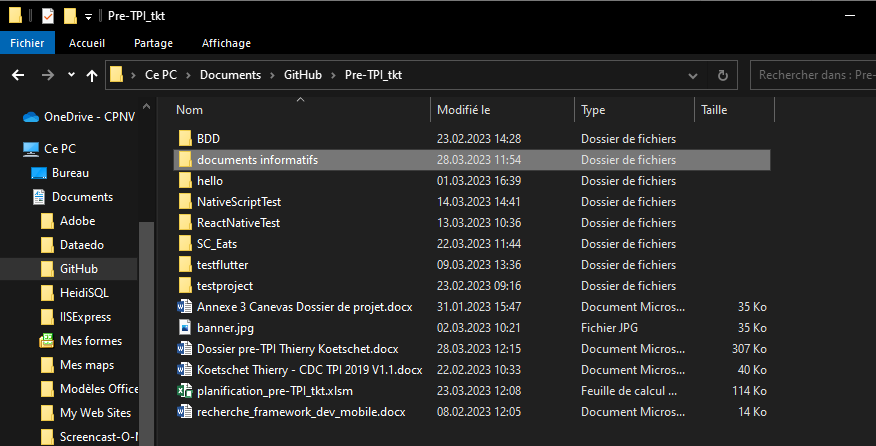
*Fournir tous les document de conception:*

* *le choix du matériel HW*
* *le choix des systèmes d'exploitation pour la réalisation et l'utilisation*
* *le choix des outils logiciels pour la réalisation et l'utilisation*
* *site web: réaliser les maquettes avec un logiciel, décrire toutes les animations sur papier, définir les mots-clés, choisir une formule d'hébergement, définir la méthode de mise à jour, …*
* *bases de données: décrire le modèle relationnel, le contenu détaillé des tables (caractéristiques de chaque champs) et les requêtes.*
* *programmation et scripts: organigramme, architecture du programme, découpage modulaire, entrées-sorties des modules, pseudo-code / structogramme…*

***Le dossier de conception devrait permettre de sous-traiter la réalisation du projet !***

# Réalisation

## Dossier de réalisation



La totalité du projet se trouve sur un répertoire GitHub ouvert au public (<https://github.com/ThierryKoetschet/Pre-TPI_tkt>). L’organisation du répertoire a été fait de la manière suivante :

* Tout ce qui concerne la base de données (MCD, MLD, scripts, backups, …) se trouve dans le dossier « BDD »
* Le dossier « documents informatifs » contient des documents concernant l’évaluation du projet par le responsable de projet et les experts
* Le dossier « hello » contient une tentative de création d’application avec Cordova
* Le dossier « testproject » contient également une tentative de création de projet avec React Native
* Le fichier « planification\_pre-TPI\_tkt.xlsm » sert à l’organisation et la planification des tâches et contient également le journal de travail.
* Le fichier « Koetschet Thierry – CDC TPI 2019 V1.1.docx » est la première version du cahier des charges transmis par M. Saison pour la réalisation du pré-TPI.

*Décrire la réalisation "physique" de votre projet*

* *les répertoires où le logiciel est installé*
* *la liste de tous les fichiers et une rapide description de leur contenu (des noms qui parlent !)*
* *les versions des systèmes d'exploitation et des outils logiciels*
* *la description exacte du matériel*
* *le numéro de version de votre produit !*
* *programmation et scripts: librairies externes, dictionnaire des données, reconstruction du logiciel - cible à partir des sources.*

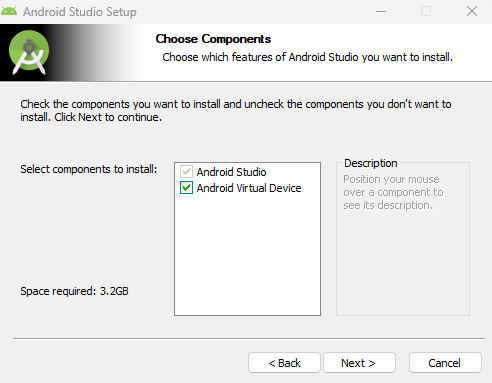
*NOTE : Evitez d’inclure les listings des sources, à moins que vous ne désiriez en expliquer une partie vous paraissant importante. Dans ce cas n’incluez que cette partie…*

## Installation de NativeScript

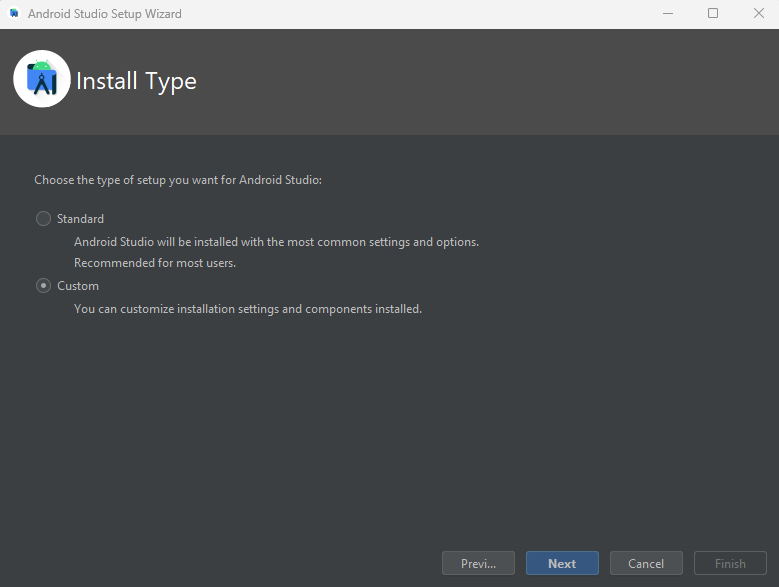
Cette marche à suivre ne traite que de l’installation et de la configuration de NativeScript pour Windows 10.

Tout d’abord, il faut télécharger Chocolatey afin de pouvoir installer les dépendances Nodejs et Java Developpement Kit. Pour ce faire nous devons télécharger la licence sur le site de Chocolatey[[6]](#footnote-6). Nous pouvons alors saisir la commande **choco install -y nodejs.install temurin11** pour installer les dépendances citées.

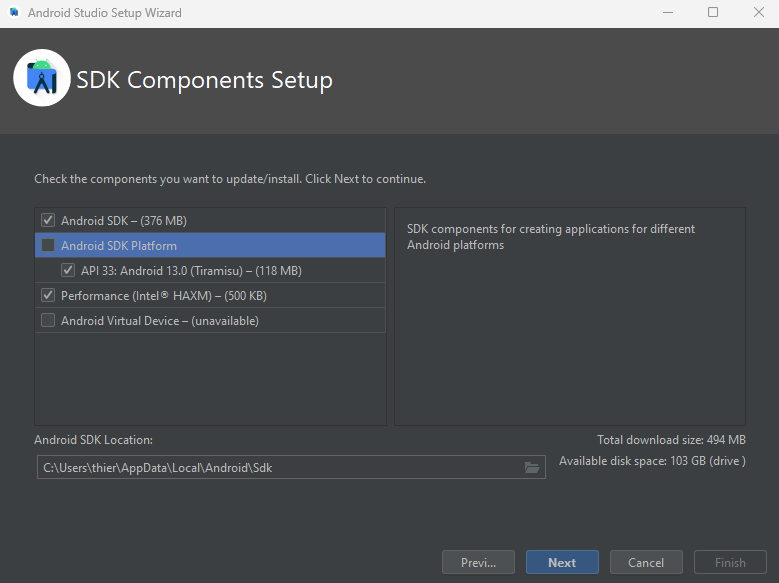
L’étape suivante consiste à installer Android Studio et tous ses composants complémentaires tels que Android SDK, Android SDK Platform, Android Virtual Device et Performance (Intel ® HAXM). Ces composants complémentaires sont absolument nécessaires pour le fonctionnement de la future application. Nous pouvons commencer par installer Android Studio en téléchargeant l’exécutable sur le site officiel de Android Studio[[7]](#footnote-7). Exécutez le fichier téléchargé, puis continuez l’installation.



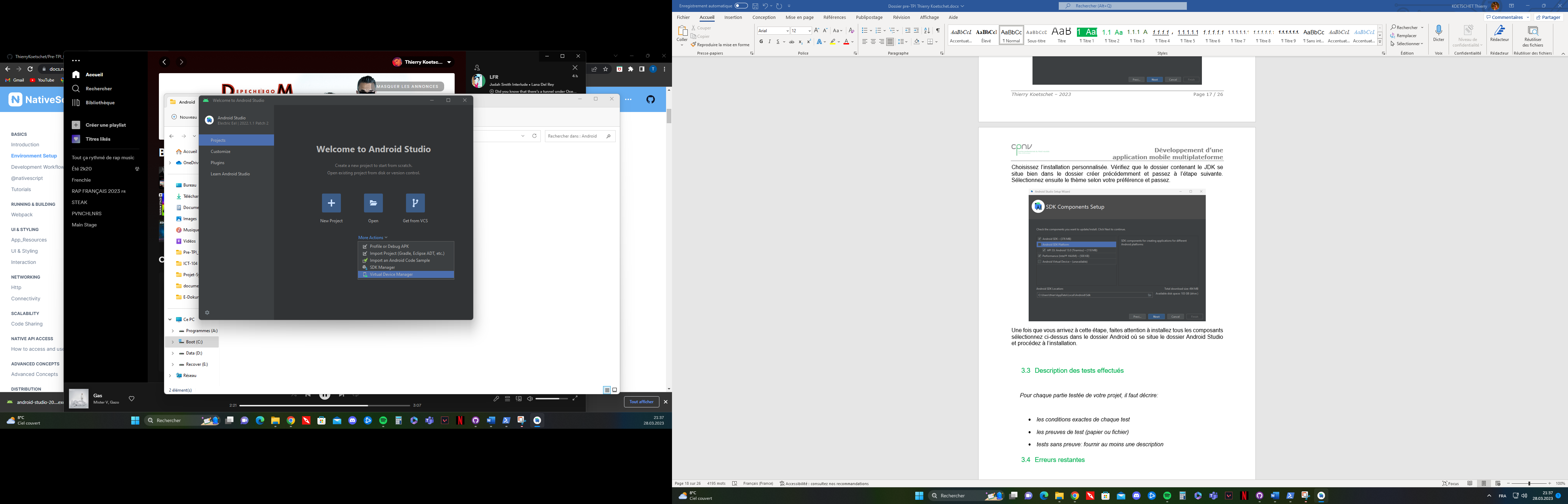
La checkbox Android Virtual Device doit être cochée pour continuer l’installation. Choisissez un dossier d’installation et gardez ce dossier dans un coin de votre mémoire car tous les composants complémentaires devront être installés dans le même dossier afin de nous faciliter la tâche avec les variables d’environnement. Enfin, procédez à l’installation d’Android Studio et lancez l’application. Il faut ensuite passer les prochaines étapes jusqu’à ce que l’on doive choisir le type de configuration que l’on veut faire pour Android Studio.



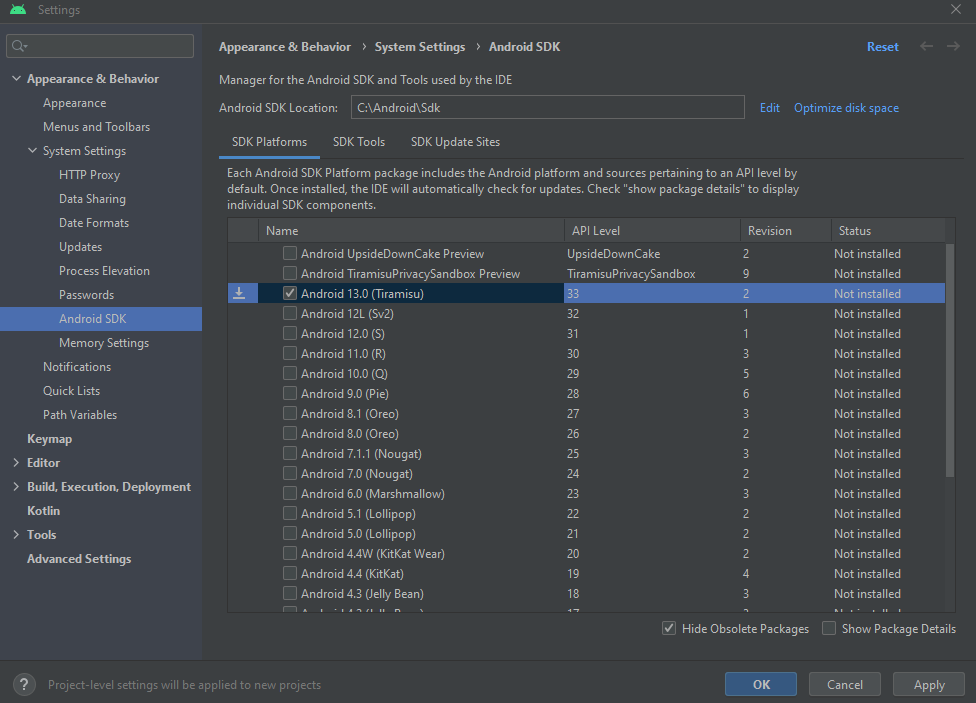
Choisissez l’installation personnalisée. Vérifiez que le dossier contenant le JDK se situe bien dans le dossier créer précédemment et passez à l’étape suivante. Sélectionnez ensuite le thème selon votre préférence et passez.



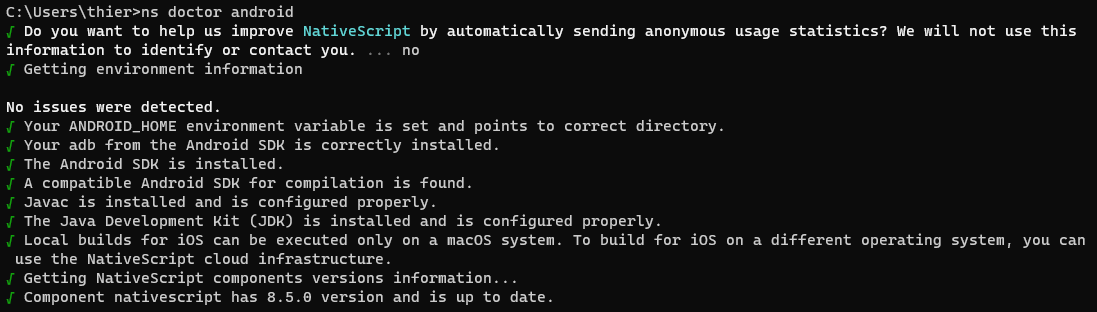
Une fois que vous arrivez à cette étape, faites attention à installez tous les composants sélectionnez ci-dessus dans le dossier Android où se situe le dossier Android Studio et procédez à l’installation.



Ensuite, accédez à SDK Manager et installez Android 13.0 (Tiramisu) de l’onglet SDK Platforms s’il n’est pas déjà installé. Il servira d’API pour l’émulateur.

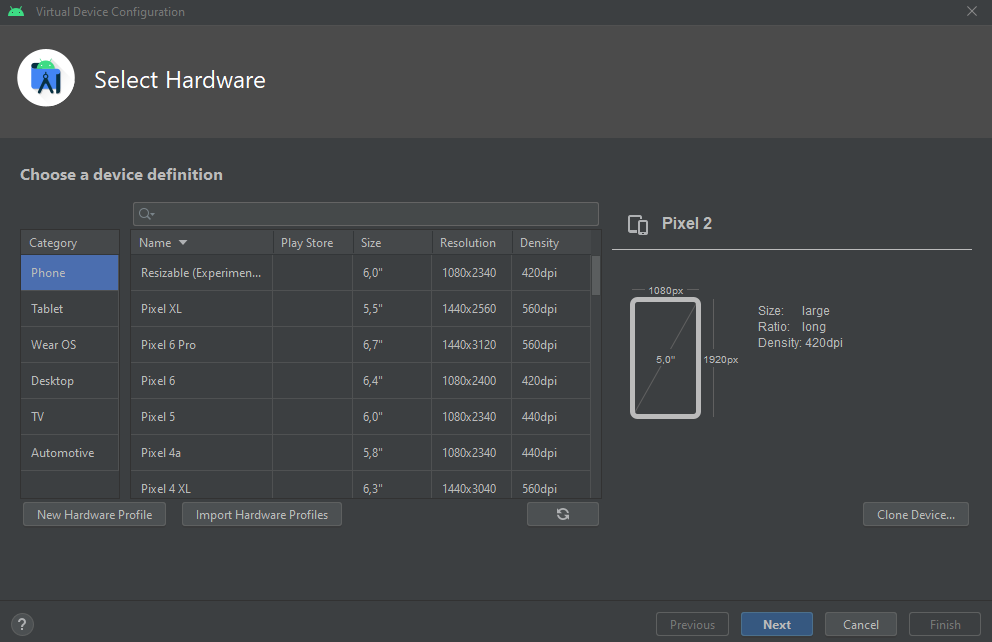


Nous allons maintenant configurer les variables d’environnement pour permettre au framework NativeScript de trouver Android SDK. Pour ce faire, éditez les variables d’environnement système dans la barre recherche Windows, cliquez sur « Variables d’environnement … » et, sous variables utilisateur, créez une nouvelle variable nommée **ANDROID\_HOME** avec comme valeur le dossier Sdk situé dans le dossier Android que nous avons créer au début de la marche à suivre. Modifiez ensuite la variable « Path » et ajoutez une nouvelle variable **[Dossier Android]\Sdk\platform-tools**.

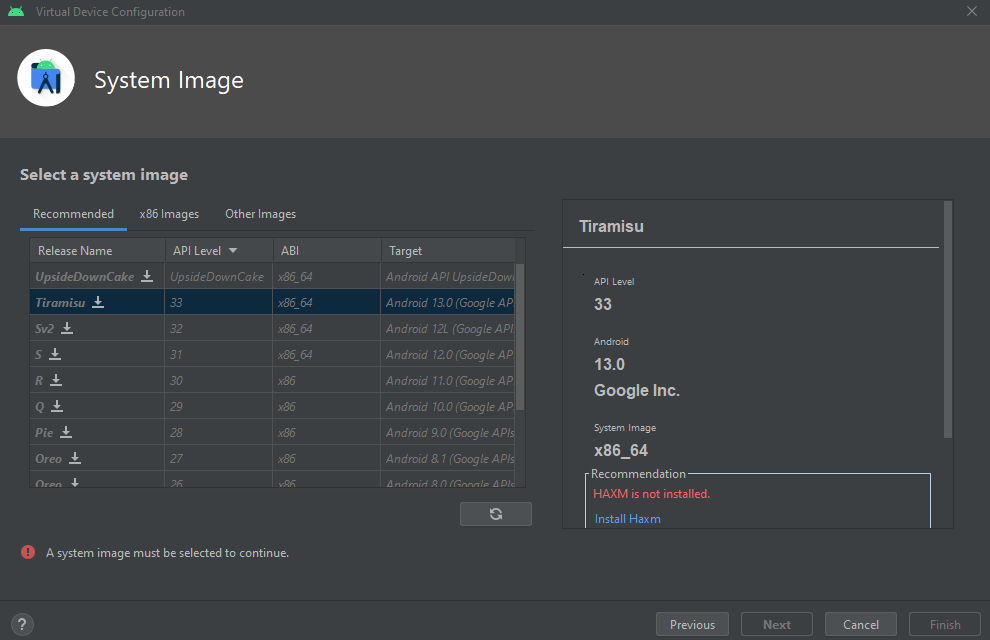
Nous pouvons maintenant installer NativeScript CLI en saisissant la commande **npm install -g nativescript** dans un invite de commande. La commande ns doctor android permet de vérifier si tous les composants nécessaires au fonctionnement de NativeScript sont installés ou configurés.

Si vous avez respecté la marche à suivre jusqu’ici, toutes les vérifications de la commande **ns doctor android** devraient être validées comme sur la capture ci-dessus.

L’ultime étape consiste à créer un AVD (Android Virtual Device) pour pouvoir visualiser l’application mobile en taille réelle. Pour ce faire, accédez au Virtual Device Manager de Android Studio et cliquez sur « Create device ».



Sélectionnez ensuite le téléphone que vous souhaitez émuler. J’ai personnellement choisi le Pixel 6.



Sélectionnez alors l’image système Tiramisu pour correspondre au choix de la plateforme SDK et créez l’appareil virtuel.

L’installation et la configuration de l’environnement de développement sont terminées et l’application devrait tourner pouvoir tourner sur l’émulateur en exécutant la commande **ns run android** dans le dossier contenant l’application mobile.

## Description des tests effectués

*Pour chaque partie testée de votre projet, il faut décrire:*

* *les conditions exactes de chaque test*
* *les preuves de test (papier ou fichier)*
* *tests sans preuve: fournir au moins une description*

## Erreurs restantes

*S'il reste encore des erreurs:*

* *Description détaillée*
* *Conséquences sur l'utilisation du produit*
* *Actions envisagées ou possibles*

## Liste des documents fournis

*Lister les documents fournis au client avec votre produit, en indiquant les numéros de versions*

* *le rapport de projet*
* *le manuel d'Installation (en annexe)*
* *le manuel d'Utilisation avec des exemples graphiques (en annexe)*
* *autres…*

# Conclusions

Pour conclure ce travail, je pense que les objectifs que je m’étais fixés en début de projet sont loin d’avoir été atteints. Certes, j’ai réussi à choisir un framework qui correspond à mes attentes pour la réalisation d’une application mobile multiplateforme, cependant j’ai aussi réalisé que le développement mobile nécessite une formation spécifique et qu’il ne suffit pas d’apprendre les bases en autodidacte.

A propos de mon application, je suis vraiment déçu du produit fini. J’ai tellement perdu de temps en tentant d’installer les différents frameworks et je ne me rendais pas encore compte au début du semestre le temps que cela me prendrait et toutes les difficultés que j’allais rencontrées. Effectivement, je n’ai pas réalisé qu’installer un environnement de développement fonctionnel, avec un framework particulier à chaque fois et en plus de faire tourner l’application sur un émulateur pouvait être aussi laborieux. Régulièrement, lorsque j’essayais de visualiser le rendu de mon application sur l’AVD, il refusait de fonctionner et le lendemain il fonctionnait à nouveau correctement.

Je pense que de manière générale, ma frustration due à la répétition de problèmes rencontrés au cours du travail a eu un impact sur ma motivation et fait que je rends un projet dont je suis loin d’être fier et cela n’est pas facile moralement. De plus, j’ai réalisé que mes connaissances étaient loin d’être suffisantes pour ce qui me sera demandé au TPI. C’est pour ces raisons que j’ai décidé de demander mon chef de projet, M. Saison, de changer mon cahier des charges pour le TPI et de partir sur la réalisation d’un site web en PHP avec un CRUD et une architecture en MVC. Après discussion avec M. Saison, il a accepté ma demande et m’a confirmé qu’il adapterait le cahier des charges en conséquences.

Pour terminer sur une note positive, je pense que ce pré-TPI m’a tout de même permis de rendre compte de tout ce qu’il fallait ou ne pas faire en terme d’organisation, de documentation et de réalisation d’un projet de telle envergure. Cela reste une expérience qui à mon avis portera ces fruits et la différence sera notable entre mon pré-TPI et mon TPI.

# Annexes

## Cahier des charges

1. TITRE

Développement d’une application mobile multiplateformes

1. MATÉRIEL ET LOGICIEL À DISPOSITION

1 ordinateur type CPNV

Environnement de développement Windows

Outil de modélisation de base de données

Logiciels de la suite Microsoft Office pour la rédaction du rapport et la présentation

1. PRÉREQUIS

* Programmation C
* Développement Web (HTML5, CSS, PHP, JavaScript)
* Modélisation et gestion de base de données (MySQL)

1. DESCRIPTIF DU PROJET

Le but de ce projet consiste en la réalisation d’une application mobile multiplateforme : iOS, Android.

Dans un premier temps, le candidat devra analyser les différentes solutions existantes :

* Applications natives
  + iOS : Xcode (VM MacOS sur Windows), avec programmation en Objective-C vs Swift
  + Android : Android Studio ou équivalent, avec programmation en Java.
  + (Windows : Visual Studio, avec programmation en C#)
* Applications hybrides
  + Flutter (Google), avec programmation en Dart
  + React Native (Facebook) : React.js
  + Native Script : Angular.js, Vue.js
  + Autres ?
* Web views :
  + PhoneGap, Cordova, Capacitor, Ionic, …

L’application produite est laissée libre au candidat, mais devra respecter certaines contraintes :

* Design : responsive, pages multiples et menu de navigation
* Exploitation de données, typiquement xml (web service, flux rss, …)
* Gestion de la mémoire (*session* vs *local storage*)
* Si le temps le permet : Système de notifications, écran de chargement (*splash screen*), …

1. LIVRABLES

Le candidat est responsable de livrer à son chef de projet et aux deux experts :

* Une planification initiale sous forme électronique au format PDF le 8 février 2023.
* Un rapport de projet sous forme électronique au format PDF 1x par semaine, le vendredi soir.
* Un journal de travail sous forme électronique au format PDF 1x par semaine, le vendredi soir.
* A la fin du Pré-TPI,
  + le rapport de projet et son journal de travail imprimés,
  + le rapport de projet final et son journal de travail sous forme électronique au format PDF,
  + un fichier « archive » contenant :
    - Un script de création de la base de données SQL ;
    - Un dossier contenant l’application complète ;
    - Une procédure d’installation et de mise en service de l’application.
* Un livrable de l’application web chaque vendredi déposé sur un dépôt distant de type Github ou Bitbucket.

1. POINTS TECHNIQUES ÉVALUÉS SPÉCIFIQUES AU PROJET
2. La grille d’évaluation définit les critères généraux selon lesquels le travail du candidat sera évalué (documentation, journal de travail, respect des normes, qualité, …).
3. En plus de cela, le travail sera évalué sur les 7 points spécifiques suivants (Point A14 à A20) :
4. Qualité de la modélisation de l’application
5. L’analyse comparative des solutions existantes
6. La pertinence des choix effectués
7. Description et qualité des tests effectués
8. La compatibilité de l’application sur les divers OS / versions
9. La transmissibilité des codes produits
10. Qualité de la documentation
11. HORAIRE DE TRAVAIL

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Lundi | Mardi | Mercredi | Jeudi | Vendredi |
| 08:00 – 08:45 |  |  |  |  |  |
| 08:50 – 09:35 |  |  |  |  |  |
| 09:50 – 10:35 |  |  |  |  |  |
| 10:40 – 11:25 |  |  |  |  |  |
| 11:30 – 12:15 |  |  |  |  |  |
| Pause midi | | | | | |
| 13:30 – 14:15 |  |  |  |  |  |
| 14:20 – 15:05 |  |  |  |  |  |
| 15:20 – 16:05 |  |  |  |  |  |
| 16:10 – 16:55 |  |  |  |  |  |

## Test effectués

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Test | Résultat | Validation |
| Tests unitaires | | | |
|  | Android SDK est installé |  |  |
|  | Android Studio est installé |  |  |
|  | L’émulateur Android est installé |  |  |
|  | Les variables d’environnement sont configurées |  |  |
| Tests d’intégration | | | |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
| Tests fonctionnels | | | |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Résumé du travail

Le principal but de ce travail était d’analyser plusieurs frameworks utiles au développement d’application mobile. Cette analyse a permis de choisir un seul de ces frameworks par rapport à mes besoins personnels et aux avantages qu’il propose. Celui-ci sera utilisé pour la réalisation du projet de TPI. Le cahier des charges fourni par le chef de projet demandait également de réaliser une application mobile répondant à certaines caractéristiques.

## Sources – Bibliographie

<https://www.atlassian.com/fr/continuous-delivery/software-testing/types-of-software-testing>

*Liste des livres utilisés (Titre, auteur, date), des sites Internet (URL) consultés, des articles (Revue, date, titre, auteur)… Et de toutes les aides externes (noms)*

## Journal de travail

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Durée** | **Activité** | **Remarques** |
| 31.01.2023 |  |  |  |
|  |  |  |  |

## 

## Glossaire

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

## Table des illustrations

## Manuel d'installation

Pour l’installation du framework, veuillez suivre le point 3.3 de cette documentation.

## Manuel d'utilisation

## Archives du projet

*Media, … dans une fourre en plastique*

1. Site officiel d’Apache Cordova : <https://cordova.apache.org/docs/en/latest/guide/cli/index.html> [↑](#footnote-ref-1)
2. Documentation d’installation de React Native : <https://reactnative.dev/docs/environment-setup> [↑](#footnote-ref-2)
3. Lien vidéo d’installation React Native : [https://www.youtube.com/watch?v=oZFCt69Bccc&ab](https://www.youtube.com/watch?v=oZFCt69Bccc&ab_channel=%EF%BC%A3%EF%BC%AF%EF%BC%A4%EF%BC%A5%EF%BC%B8) [↑](#footnote-ref-3)
4. Documentation d’installation de NativeScript : <https://docs.nativescript.org/environment-setup.html> [↑](#footnote-ref-4)
5. Documentation d’installation de Flutter : <https://docs.flutter.dev/get-started/install> [↑](#footnote-ref-5)
6. Site officiel de Chocolatey : <https://chocolatey.org/install#individual> [↑](#footnote-ref-6)
7. Site officiel de Android Studio : <https://developer.android.com/studio> [↑](#footnote-ref-7)