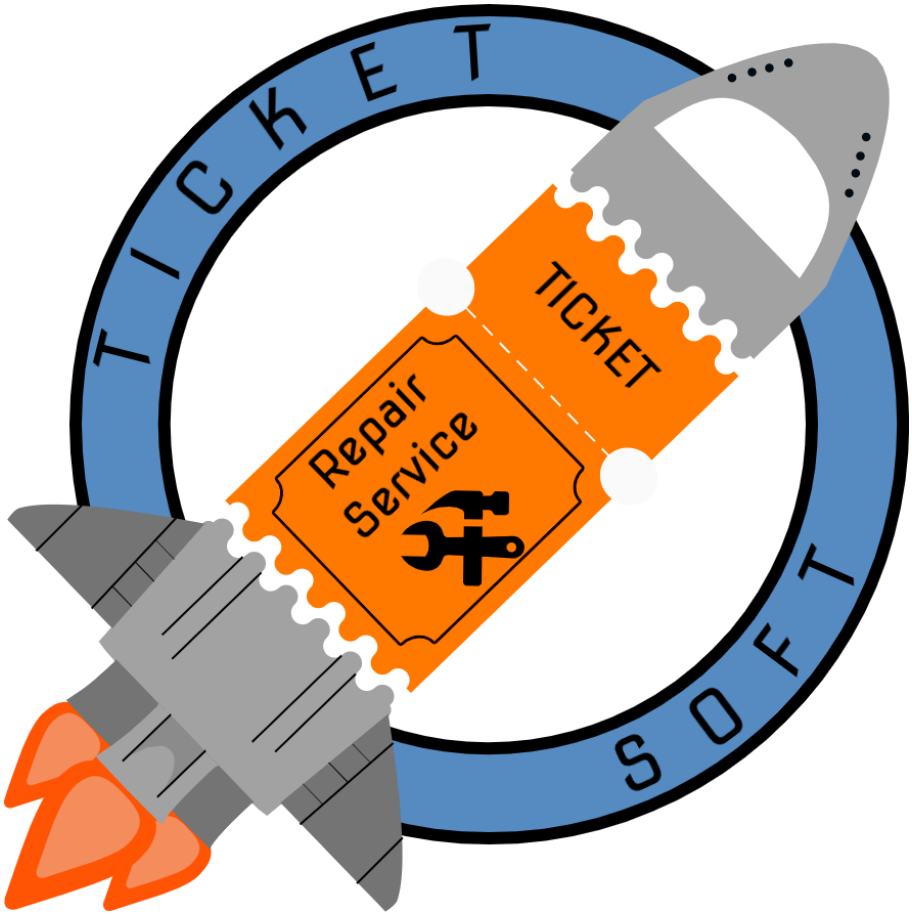
TicketSoft

Plan Assurance Qualité



Projet Génie Logiciel

2019-2020 (durée 6 mois)

Commanditaires

Valerie Guimard (valerie.guimard@u-psud.fr)

Frédéric Voisin (fv@lri.fr)

Rédacteur(s)

MALMASSON Robin, MINET Jordane, PROUST Alexis,
Eurydice, VON ASCHEBERG Thomas

Approuvé par Thomas von Ascheberg

Sommaire

Introduction

Le présent document vise à décrire la gestion de la création du produit développé par la société *IsItWorking? SARL*, ci-dessous désigné comme le "*logiciel*". Celui-ci a été commandé par Frédéric Voisin et Valérie Guimard de la société *PoPS2019-20* désignée comme le "*commanditaire*", afin de répondre aux besoins décrits dans le document de référence « Expression des besoins ». Attention à la nomenclature. Les mots clients et utilisateurs sont ambigus. On définira les termes de la façon suivante :

- **Les commanditaires** = Mme. Guimard et M. Voisin
- **Le client** = L'entreprise qui demande une intervention.
- **Le(La) demandeur(euse)** = la personne qui demande l'intervention pour le compte de l'entreprise (du client)
- **L'opérateur(trice), Le(La) technicien(nne), Le(La) responsable des techniciens** : pour les utilisateurs du logiciel. On préfère définir les utilisateurs du logiciel par leur rôle, cela évite la confusion.

Objet et diffusion

Objet

Ce plan décrit les dispositions mises en place sur le projet Ticket Software afin d'atteindre les résultats attendus, c'est-à-dire la fourniture du projet aux exigences spécifiées lors de la phase de conceptions à nos professeurs référents, Valérie Guimard et Frédéric Voisin, représentants de notre client POPS1920.

Documents de référence

Nous avons à disposition un document de référence fournit par le client. Il s'agit d'un texte dans lequel il exprime ses attentes et ses besoins. De plus nous ferons confirmer le cahier des charges fonctionnel au client.

Diffusion

La diffusion du PAQ se fait principalement en interne. Le PAQ sera envoyé au client lorsque de chaque mise à jour majeure. Il sera révisé si le client en exprime le besoin.

Objectifs et enjeux du projet

Dans le cadre de notre 5e année à Polytech Paris-Sud, dans le cadre du « Projet Génie Logiciel » nous avons été amenés à répondre aux besoins d'un client simulé par deux professeurs (Mme. Valérie Guimard et M. Frédéric Voisin). Ainsi, nous avons formé notre groupe pour répondre à leur besoin d'un logiciel de gestion de tickets en ligne.

Nature du projet

Le projet sera composé de deux parties : Le préprojet (avec toute la mise en place de la gestion du projet) et la réalisation du projet (développement de la solution aux besoins clients). Le but est de nous mettre en conditions réelles afin d'avoir une vision globale du travail qui pourra nous être demandé plus tard.

Le commanditaire est une entreprise réalisant la maintenance d'appareils industriels. Il a donc sa propre clientèle, dont chaque membre est désigné comme le "*client*" dans ce document. Le commanditaire a effectué la demande d'un logiciel de gestion de tickets de maintenance en soulignant l'importance de l'ergonomie, ainsi que de l'implémentation d'un suivi des temps passés sur les interventions et un suivi des tâches. En effet, la solution que le commanditaire a utilisée jusqu'à maintenant a une interface peu efficace et manque de fonctionnalités importantes.

Objectifs

Notre objectif est de répondre aux besoins du commanditaire, dans des conditions proches de celles de notre futur métier d'ingénieur. L'objectif est de fournir un logiciel de niveau professionnel possédant les fonctionnalités demandées. Une des choses essentielles est la gestion de projet afin de rendre en temps et en heure un projet complet aux clients.

Enjeux

L'enjeu de cette simulation est simple : tester nos compétences. Que cela soit les compétences en gestion de projet, développées au cours notre cursus. Ou au niveau de nos compétences de programmation avec un domaine nouveau pour nous : le web.

Périmètre du projet

Le périmètre du projet se limite à la conception et au développement du logiciel de gestion de tickets. Nous devrons sans cesse tenir informé le client de nos avancées, lui demander de valider les fonctionnalités implémentées, et entretenir une bonne communication par des réunions régulières.

Planning

Nous avons décidé de séparer ce projet en deux grandes phases : le préprojet et la réalisation. Le préprojet consiste en la définition du projet et de l'organisation de celui-ci via un ensemble de documents avec le commanditaire. Cela inclut notamment l'organisation de l'équipe et du travail, la conception technique du projet, c'est-à-dire l'organisation du logiciel en lui-même. Pour cette partie, nous avons déterminé le temps nécessaire pour les réaliser à partir de ce qu'on avait déjà fait au démarrage du projet, et de notre organisation hebdomadaire.

Pour la réalisation, il s'agit du développement du logiciel, avec les différentes grandes parties qui vont le composer. Pour déterminer le temps nécessaire à ce niveau, on s'est basés sur nos expériences lors de projets précédents.

Selon le planning établi, nous sommes censés finir le projet en temps et en heure, puisque que la date de fin est le 4 mars 2020 et la date limite est avant le 9 mars 2020.

Enfin, quant à la répartition des tâches, pour la partie préprojet, il a été assez facile de répartir précisément les ressources, comme tous les membres ont l'habitude de ce type de gestion de projet, et que pour ceux dont on n'avait pas encore expérimenté, on s'est basés sur les préférences de chacun.

On a fait de même pour la partie réalisation, cependant, comme nous en sommes encore à la conception, il est difficile de savoir précisément à quel lot de travail chacun sera affecté précisément, mais nous avons déjà une idée de la répartition sur les groupes de travail.

Pour le détail, voir Annexe.

Livrables

Au cours du projet

Durant le projet, de nombreux documents seront édités par l'équipe. Voici la liste des livrables qui seront à fournir pendant le projet :

- Questions/Réponses avec le client
- Compte-rendu des réunions (avec le commanditaire)
- PAQ mis à jour
- Maquettes des IHM
- Planning prévisionnel
- Présentation d'une fonctionnalité (à définir avec la maîtrise d'ouvrage)
- Analyse UML (diagrammes de classes, cas d'utilisation, diagrammes de séquence)
- Démonstration de pré-soutenance
- Cahier de tests
- Organisation des développements

Livrables de fin de projet

En fin de projet, voici la liste des livrables à fournir :

- Cahier des charges mis à jour
- Matrice de respect des exigences du cahier des charges indiquant pour chaque exigence si elle a été respectée, non respectée ou avec des réserves
- Dossiers de tests mis à jour
- Le produit final, avec un rapide retour d'expérience sur les problèmes
- La documentation d'installation, d'exploitation et utilisateur
- Ensemble de nos sources
- Ensemble de nos fichiers de test

Gestion et diffusion des documents du projet

Les documents sont partagés sur un espace de stockage en ligne (Google Drive). Leur diffusion se fait sur cet espace et sur la messagerie instantanée du groupe (Discord). Leur traçabilité sera assurée par un numéro de version accompagné de la date du document. La présence du logo de notre projet sera apposée à chaque document comme preuve de sa provenance.

Pour chaque document rédigé, le rédacteur l'enverra aux membres du sous-groupe dont il fait partie pour approbation, et au chef de projet pour validation. Vous pouvez retrouver la responsabilité de chacun sur la matrice RACI.

Nous travaillerons avec un gestionnaire de version (GitHub) pour partager le code de façon efficace et ainsi réduire les risques de problème de version. À ce GitHub, nous avons ajouté le plugin Zenhub, qui permet l'application des méthodes agiles de management de projet.

Gestion des demandes spécifiques

Les demandes sont traitées selon leur ordre d'importance puis d'arrivée. Elles sont traçables grâce aux mails, puis diffusées au groupe grâce à Discord. Si la demande est importante, une réunion sera organisée afin de décider des mesures à prendre. Un compte rendu de la réunion sera établi et remis au demandeur par mail si c'est le client, diffusé publiquement à l'adresse du demandeur sur Discord s'il est membre du groupe du projet.

Organisation et responsabilités

Les acteurs du projet, leurs rôles et responsabilités

Nous sommes une équipe de 6 personnes, avec pour chef de projet Thomas, qui a été choisi au début de celui-ci. Du fait que nous utilisons les méthodes agiles (SCRUM) nous avons Robin qui est en charge d'être le Scrum-master de l'équipe. De plus, Robin occupe aussi le rôle de responsable communication (il gère les relations avec le commanditaire).

Afin de nous organiser et répartir les différents lots de travail entre les membres de l'équipe, nous avons réalisé une matrice RACI permettant de définir les rôles de chacun dans le projet. Cette distribution a été réalisée sur la base des compétences et des volontés de chacun. Celle-ci n'est pas définitive et est susceptible de changer à l'avenir.

Phase	Catégorie	Lot de travail	Thomas	Robin	Alexis	Eurydice	Jordane	Xiaoxuan	Equipe	Client
Pré-projet	PQP	Définition du périmètre	C	C	C	C	C	I	R	A
		Matrice RACI	A	I	I	I	R	I		
		Matrices des Risques	A	I	I	I	R	I		
		Planning Prévisionnel	A	I	I	R	I	I		I
		Organisation / Rédaction	C	C	C	C	C	I	R	A
	Conception	Cahier des charges	C	R	I	I	C	I		A
		Dossier de tests	I	R	I	I	A	I		
		Choix technologiques	AC	C	C	C	C	I	R	I
	IHM	Maquettes	R	C	C	C	C	I		A
		Storyboard	R	C	C	C	C	I		A
	Base de données	Conception base de données	I	R	I	I	A	I		
	UML	Cas d'utilisations	A	I	R	C	I	I		A
		Classe	C	C	R	A	C	C		
		Séquence	C	I	A	R	I	I		
Réalisation	Back-end	Architecture du logiciel	R	A	I	C	I	I		
		Stockage des données	C	R	I	A	I	I		
		Traitement des données	R	A	I	A	I	I		
		Envoi des données	C	A	I	R	I	I		
	Front-end	Layout pages	I	I	A	C	R	I		
		Envoi des données	I	I	R	A	C	I		
		Lien avec back-end	I	I	C	R	C	I		
		Affichage des données	I	I	R	C	C	I		
	Tests	Tests unitaires	C	C	C	C	A	R		
		Tests d'intégration	C	A	C	C	C	R		
		Tests whitebox	C	C	C	A	C	R		
		Tests blackbox	C	C	A	C	C	R		
	Documentation	Documentation d'utilisation	A	C	C	C	C	C	R	
		Javadoc	A	C	C	C	C	C	R	
	Base de données	Création des tables	I	A	I	I	I	R		
		Lien données vers l'application	I	A	I	I	I	R		
	Livraison	Déploiement	C	C	C	C	C	C	R	A

Figure : Matrice RACI

Organisation du travail

Dès le début du projet, des réunions hebdomadaires avec tous les membres de l'équipe (si présents) ont été instaurées. Leur ordre du jouer est déterminé à l'avance et un compte-rendu est écrit à la fin. Celles-ci permettent de fixer la direction globale du travail, suivre l'avancement du travail de chacun et éventuellement discuter des problématiques qui pourraient surgir. Dans le cas de l'absence de l'un des membres de l'équipe, celui-ci a la responsabilité de "rattraper" cette réunion en lisant le compte-rendu et posant des questions au reste de l'équipe en cas de flou.

C'est au cours d'une de ces réunions qu'il a été décidé d'adopter une méthode de travail *Agile* en adaptant l'implémentation *Scrum* à nos besoins. En effet, Scrum propose un ensemble de pratiques qui sont appropriées pour une équipe développant un produit. Cependant, dans la mesure où la partie développement de ce projet ne démarre que plus tard, nous n'avons sélectionné que quelques-unes de ces pratiques pour le travail effectué en amont du développement.

C'est ainsi que nous en sommes venus à instaurer un système de *sprints* d'une durée de quelques semaines, à la fin desquelles nous proposons une réunion avec le commanditaire du logiciel. Cela nous permet de conserver un mode de travail itératif, entouré de *sprints plannings* et *rétrospectives* qui nous permettent de fixer les objectifs à court terme et d'améliorer le travail au sein de l'équipe. À cela s'ajoutent, si besoin, des *"daily scrums"* (qui n'ont pas lieu tous les jours mais plutôt quand le besoin s'en fait sentir) pour suivre le travail de chacun et être au courant d'éventuelles difficultés au plus tôt.

À ces pratiques Scrum s'ajoute l'utilisation d'un outil de gestion de projet appelé *ZenHub*, similaire à l'outil *Trello*. ZenHub est directement intégré à notre répertoire GitHub et permet de créer des *user stories* (unités de travail assignées à une ou plusieurs personnes), les regrouper en *epics* (un ensemble de tâches représentant une fonctionnalité) et suivre l'avancement du travail du groupe. Robin MALMASSON a pris le rôle du *Scrum Master*, dans le sens où il fait appliquer les différentes pratiques scrum décrites plus haut et répond aux interrogations liées à la méthodologie si besoin.

Moyens : techniques, outils, standards

Pour ce projet nous avons mis différents moyens en œuvre afin de répondre au mieux aux besoins du commanditaire. Dans un premier temps nous allons détailler les technologies choisies pour réaliser le projet. Tout d'abord sur le plan technique :

- **Java** comme langage de programmation du côté applicatif (back-end)
- La combinaison **HTML-CSS-Javascript** pour le côté IHM (front-end). Je précise qu'aucun framework front-end ne sera utilisé pour le front-end.
- Une **base de données relationnelle** basée sur le standard SQL

Puis sur le plan de l'infrastructure :

- Nous allons essayer d'utiliser AWS pour héberger au moins la base de données et peut-être plus (si nous arrivons à rester dans le tiers gratuit)

Finalement sur un plan organisationnel, nous utilisons plusieurs outils :

- **Git** pour la gestion des différentes versions du code. **GitHub** pour l'hébergement en ligne du code
- **ZenHub** pour la gestion agile du projet (découpages en sprint, avancement des sprints, etc.)
- **Google Docs** pour tout ce qui est écriture collaborative des documents (compte-rendus de réunion, documents de gestion de projet, etc.)
- **Discord** (un logiciel de messagerie instantanée) pour communiquer entre nous sur des sujets précis du projet.

Pour les standards, nous utiliserons les standards suivants :

- Nous utiliserons **Java8** comme version de Java. Les versions ultérieures cassant la rétrocompatibilité et nécessitant une licence, nous utiliserons de fait la dernière version libre de Java. Plus précisément, nous utiliserons le compilateur jdk8 d'Oracle.
- **HTML5/CSS3** ce couple étant le format moderne pour la programmation web, nous décidons de l'utiliser
- **"use strict" Javascript**. Afin de garantir une meilleure qualité du code Javascript, nous utiliserons le mode strict de Javascript. Ce dernier impose une syntaxe rigoureuse (à la C++/Java) et limite, notamment, les abus possibles en termes de portée des variables.

Maîtrise du projet

Dispositions de management :

Étant donné le cadre scolaire de ce projet, aucun budget n'a été alloué. Ainsi :

- Maîtrise des dépenses : aucune dépense prévue
- Maîtrise des ressources : tout est détaillé dans le planning. Nous allons essayer de coller au plus proche à ce que nous annonçons grâce aux méthodologies agiles.

Dispositions techniques :

Dans le but de voir le projet se dérouler dans les meilleures conditions possibles, une analyse des risques du projet a été réalisée. Cette analyse révèle de nombreux risques possibles pouvant survenir durant le déroulement du projet, répartis sur ses différentes étapes. Nous en avons relevé sept comme étant critiques et ayant donc la capacité de mettre en échec le projet. Voici leur description :

- Une mauvaise compréhension des besoins, que nous empêchons avec de nombreuses discussions avec le commanditaire.
- Un dépassement des délais prévus dans le planning prévisionnel, que nous gérons par l'utilisation des méthodes agiles (Scrum) et des réunions hebdomadaires. Cela nous permet alors de suivre l'avancement du projet à courts termes et nous adapter en cas de problème.
- Une mauvaise compréhension de la part des membres sur la manière dont fonctionne les différentes parties de l'application et comment elles communiquent entre elles. Cela se résout grâce à la création de différents schémas détaillant le fonctionnement de l'application.
- La rencontre de plusieurs bugs majeurs lors du développement, le retardant fortement alors. La mise en place de revues de code durant la phase de développement nous permet de prévenir de cela, ainsi que la possibilité de donner une forte priorité à la résolution de ce bug pour toute l'équipe.
- Le non fonctionnement des services/serveurs importants qui empêcheraient l'application de fonctionner. Pour empêcher cela, nous allons prioriser cette partie afin de la fonctionner le plus rapidement possible.
- La perte de matériel comme un ordinateur ou serveur. Cela est protégé par l'utilisation de nombreuses plateformes en ligne pour le code (Github) et les documents (Google drive).

- La perte de motivation de la part des membres de l'équipe, que nous limitons par la possibilité à la fin de chaque réunion hebdomadaire de parler de manière ouverte des soucis rencontrés ou de ce qui ne va pas.

Vous pouvez retrouver l'intégralité de cette matrice dans la figure suivante :

Besoins classifiés par lots	Fonctions exprimées	Risques Associés	Estimation			Actions Envisagées	Nouvelle Estimation		
			Probabilité	Gravité	Taux de criticité		Probabilité	Gravité	Taux de criticité
Analyse / Définition du projet	Analyse du besoin du commanditaire	Mauvaise compréhension des besoins	3	5	15	Discussion et questions avec le commanditaire + Validation du cahier des charges fonctionnel par le commanditaire	2	5	10
Organisation du projet	Plannification	Dépassement des délais prévus (tâches)	3	4	12	Réunions hebdomadaires + utilisation des méthodes agiles (Scrum) afin de savoir au mieux l'avancement des tâches et s'adapter en conséquence	2	3	6
	Méthodes agiles	Perte de vision globale du projet	2	3	6	Mesurer l'avancée globale du projet à l'aide du planning + Sprint sur de grandes parties du projet	1	3	3
		Difficulté à être appliquée en parallèle des cours (réunions matinales rapides)	2	4	8	Au moins une grosse réunion hebdomadaire qui permet de maintenir un rythme de progression du projet	2	2	4
Conception	Choix des technologies	Technologie choisie inadaptée	2	5	10	Recherche de différentes alternatives + évaluation des conséquences au changement	1	2	2
	Conception de l'application	Ambigüité sur le fonctionnement de la communication entre front et back	3	4	12	Définition claire du fonctionnement de la communication entre chaque partie au travers de divers schémas et diagrammes	2	4	8
Réalisation	S'accorder sur la manière de développer	Chaque membre code / fonctionne à sa manière	4	2	8	Réalisation d'une charte de développement (forme du code, dépot en ligne, etc)	1	2	2
	Développement	Interface créée ne convient pas au commanditaire	3	2	6	Discussion avec le commanditaire afin de créer une meilleure interface	2	3	6
		Bugs majeurs et difficiles à résoudre	3	4	12	Revue de code + entraide de l'équipe	3	2	6
		Mise en place des services back-end compliquée / non fonctionnels	3	5	15	Partie à prioriser en début de projet, afin d'assurer une marge de sécurité sur le temps et pour la suite du projet	1	5	5
	Tests	Fonctionnalité non testée	2	5	10	Test d'une fonctionnalité à la fin de son développement	1	5	5
Management des ressources (Humaines / Matériels)	Perte de données	Base de données détruite	3	3	9	Réalisation de sauvegardes régulières	1	2	2
		Documents (techniques, fonctionnels, etc) détruits	2	5	10	Sauvegarde en ligne des documents + doublons pour les plus importants	1	2	2
	Perte de matériel	Ordinateur non fonctionnel / cassé	3	5	15	Sauvegarde en ligne des documents + dépot en ligne du code de l'application (Git pour le code, google drive pour les documents, cloud pour base de données)	2	2	4
	Maladie	Retard / incapacité à avancer dans le projet pour un ou plusieurs membres	3	3	9	Discussion avec les autres membres afin d'adapter le planning ou répartir la charge de travail	3	2	6
	Motivation	Perte de motivation des membres	3	4	12	Discussions ouvertes en fin de réunion sur les difficultés morales rencontrées	1	3	3
	Intégration d'un nouveau membre	Difficulté à travailler avec (emploi du temps, adaptation des rôles,...)	5	2	10	Discussion avec le nouveau membre + adaptation du fonctionnement de l'équipe pour intégrer ce membre du mieux possible	5	1	5

Figure : Matrice des risques

Procédures de livraison

Les livrables ne seront pas commandés au cas par cas : comme ce sont des produits numériques, ils seront délivrés via un lien sur un site d'hébergement (ou par un support de stockage standard).

Pour cette raison, la recette se fera chez le client (à Polytech). Le code source, la documentation, les tests, ainsi que le produit final, seront fournis au client et seront disponibles sur GitHub, en libre-service en ligne. Il y aura également une remise des livrables en main propre lors de la présentation finale de notre projet au client.

Suivi des actions correctives

Les actions correctives sur le plan technique seront gérées par notre outil de versionnement en ligne : GitHub. Ainsi, toutes les versions du projet seront gardées sur le Github du projet. Le client aura ainsi accès en permanence aux correctifs du code. Leur suivi sera garanti par GitHub, en datant les correctifs, et en ayant à disposition toutes les versions précédentes.

Les problèmes qui apparaîtront au cours du projet, seront d'abord traités dans les sous-groupes de travail, et s'ils persistent, le groupe complet se réunira afin de trouver une solution.

Gestion et suivi du PAQ

Un suivi et une mise à jour du PAQ sont prévus tout au long du projet, notamment en étant à l'ordre du jour des réunions.

À la fin du projet, et avant la livraison, une révision importante est prévue afin de rendre compte de l'état du projet, des problèmes rencontrés, des solutions mises en place…

Notamment, une mise à jour des documents suivants, si modification : RACI, planning, actions correctives…