ENSSAT
LANNION

Projet UML

Projet de l'entreprise BLIT « Best Transfort For Yourself »

me [at] colinleverger [dot] fr

Colin Leverger - ENSSAT Informatique, Multimédia et Réseaux
Promotion 2017

Client : Entreprise BLIT, « Better Live In Town »
Destinataire final : Jean-Baptiste Caignaert

5 novembre 2015

Introduction

Dans le cadre de la formation Informatique, Multimédia et Réseaux dispensé à l'E.N.S.S.A.T. de Lannion, nous avons suivi en deuxième année un cours sur le formalisme UML [1]. Cette manière de décrire un système, largement utilisée dans le monde du développement logiciel, permet de concevoir des projets dans un langage compréhensible par les humains et par les machines. Il s'agit de décrire de manière très visuelle les interactions entre les différentes composantes d'un système, afin de spécifier le travail de développement attenant et de fixer des objectifs clairs.

L'objectif de ce module est de nous sensibiliser à la conception de logiciels et de nous apprendre à procéder. Jusqu'à présent, nous avions l'habitude de fonctionner par instincts dans le développement de logiciels, sans avoir une méthode fixée pour la conception. Il s'avère que de réfléchir à la conception d'un logiciel n'est pas iné, et le langage UML permet de cadrer cette phase de réflexion.

Dans ce contexte, nous avons réalisé un projet qui va s'appuyer sur la méthodologie UP [2]. Celle-ci permet de travailler de manière itérative à la conception d'un logiciel. Il s'agit de décomposer le système en plusieurs fonctions importantes, puis de décomposer cesdites fonctions en sous fonctions, décomposables elles même en suivant le même protocole. Il faut ensuite expliquer chaque fonction, calculer les risques et expliciter l'utilité fonctionnelle de ces dernières.

Ce rapport va s'articuler en trois mouvements. Premièrement, une description du projet et des objectifs à destination du professeur sera donnée. Le rapport client sera présent en deuxième partie ; il est destiné au client virtuel de ce projet. La conclusion sera quant à elle disponible en troisième partie.

1. UML : Unified Modeling Language
2. Introduction sur la méthodologie UP : Unified Process, developpez.com

Table des matières

Partie 1

Partie professeur

1.1 Le sujet et les modalités

Le sujet choisi pour cette étude sera la conception du logiciel **BTFY**, « Best Transport For Yourself ».

Le sujet explicite l'idée du client, de manière très conscice et peu détaillée. Il faut évidemment que le concepteur imagine le concept, détaille, agrémente et conçoive les fonctionnalités de manière très précise.

Le client souhaite réaliser une application permettant de rendre les transports d'une ville « intelligents ». Il s'agit de prévoir le trafic afin de proposer aux utilisateurs des recommandations personnalisées, qui leur permettront de suivre le chemin le plus adapté pour circuler entre un point A et un point B dans une ville.

Une approche critique de l'idée est attendue, ainsi qu'une proposition sous forme de rapport. Le protocole UP est imposé pour la conception.

1.2 Estimation des coûts

Avant de partir dans la conception, nous devions estimer le temps que nous allions passer à travailler sur ce projet. Ne connaissant pas la méthodologie UP, j'ai estimé le travail comme suis :

— Entre 32h30 et 35h00 de travail en tout sur le projet (incluant choix du sujet, conception des diagrammes UP et des diagrammes divers / uses cases, rédaction du rapport).
— Entre 4 et 5 niveaux de UP.

Un retour sur cette estimation sera fait dans la partie conclusion.

Partie 2

Partie client

Votre entreprise **BLIT** est spécialisée dans la gestion de ville intelligente. Avec l'essor du « Big Data » et l'avènement des nouvelles technologies (notamment Smartphone, Internet, …), vous avez eu une idée de produit. Ce produit, que vous avez nommé **BTFY**, « Best Transport For Yourself », permettra aux utilisateurs de profiter de temps de trajet optimisés et donc de passer moins de temps dans les transports.

Je vous fournis la conception de votre produit clef en main, en tant que concepteur de services et d'application. Je réponds avec ce présent document à l'appel d'offres que vous avez envoyé le 16 octobre 2015.

2.1 Capture des besoins

2.1.1 Contexte et rappels sur votre appel d'offres

Circuler en véhicule dans les grandes villes du monde n'est pas toujours une tâche aisée. Les embouteillages, les ralentissements, les accidents et les évènements temporaires ne rendent pas toujours la vie des conducteurs aisée. Il est facile de perdre du temps dans les transports, et le monde est face à un problème grandissant : comment éviter tous ces problèmes et améliorer la qualité des transports dans nos villes ?

C'est dans ce contexte que vous voulez mettre en place une application innovante pour *rationaliser* les moyens de transport. Votre application permettra de modifier les moyens de transport d'une ville en fonction des jours précédents. Il s'agit d'analyser le comportement des conducteurs

et du trafic des journées / mois précédent, d'analyser le contexte (départs en vacances, évènements divers et variés) et d'émettre des recommandations aux utilisateurs concernant leur trajet. Quelle route doivent-ils emprunter pour perdre le moins de temps possible ?

L'application ne concernera qu'une et une seule ville, les utilisateurs ne pourront donc améliorer leur déplacement qu'au sein de celle-ci (et pourquoi pas de l'agglomération attenante). Il sera impossible pour l'utilisateur de *créer un trajet entre deux villes différentes.*

L'application concernera tous les modes de transports existants au sein d'une ville : bateaux-mouches, avions, voitures, bus…

L'accès a votre application sera bien évidemment payant, et les utilisateurs pourront éventuellement avoir une réduction s'ils partagent leurs futurs trajets. Cela permettra évidemment de créer de la valeur ajoutée à votre système, en proposant des trajets basés sur des données *passées* et *futures.*

Les utilisateurs auront accès à de multiples fonctionnalités au travers de leurs *comptes utilisateur.* Ils pourront partager leurs trajets, chercher des nouveaux trajets et mettre à jour leurs informations personnelles (liste non exhaustive).

L'application sera contrôlée par un administrateur, qui pourra ajouter des « évènements » ; ces évènements seront en fait des manifestations qui pourront avoir une influence sur les transports dans la ville pendant un laps de temps donné. L'administrateur ne va évidemment pas *organiser* les évènements, mais plutôt les créer et les mettre à jour dans l'application.

2.1.2 Objectifs

- Faire du bénéfice en cannibalisant le marché (ce type d'application n'existe actuellement pas dans le marché). Créer un standard international.
- Analyser les déplacements des gens à grande échelle.
- Calculer des trajets pertinents et arrangeants pour les utilisateurs.
- Avoir une interface Web ergonomique et facile à utiliser pour que tous les clients s'y retrouvent.
- Vendre l'application dans 50 villes de France d'ici janvier 2017. [1]
- Utiliser l'argument écologie pour asseoir l'image de votre marque **BLIT**.

Le but de cette application pourra être résumé en une phrase : « Gestion collaborative, intelligente et rationnelle des différents moyens de transport dans une ville »

1. Objectif viable, pourquoi pas…

2.1.3 Slogans

Slogans proposés pour l'application :

— « BTFY rend le transport easy ! »
— « BTFY, pour des transports meilleurs ! »
— « BTFY, même pour les Astèques ! » [2]

2.1.4 Besoins fonctionnels

Définitions

Les transports

Les transports d'une ville pourront être définis comme sur la figure 2.1 en page 8. Ils seront divisés en cinq catégories : les transports urbains sur rails, les transports interurbains sur rails, les transports aériens, les transports maritimes et les transports en commun de type bus.

Un utilisateur

Un utilisateur sera une personne abonnée au service payant que vous proposez.

Le système

Le système représentera votre service à proprement parler : **BTFY**.

Liste des besoins

L'application que vous cherchez à développer va « rationaliser les moyens de transport dans une ville ». Le niveau UP associé (N0) sera visible en figure 2.2 en page 8.

Les besoins fonctionnels relevés seront les suivants :

2. Slogan axé sur la prononciation française du nom du produit, BTF**Y**

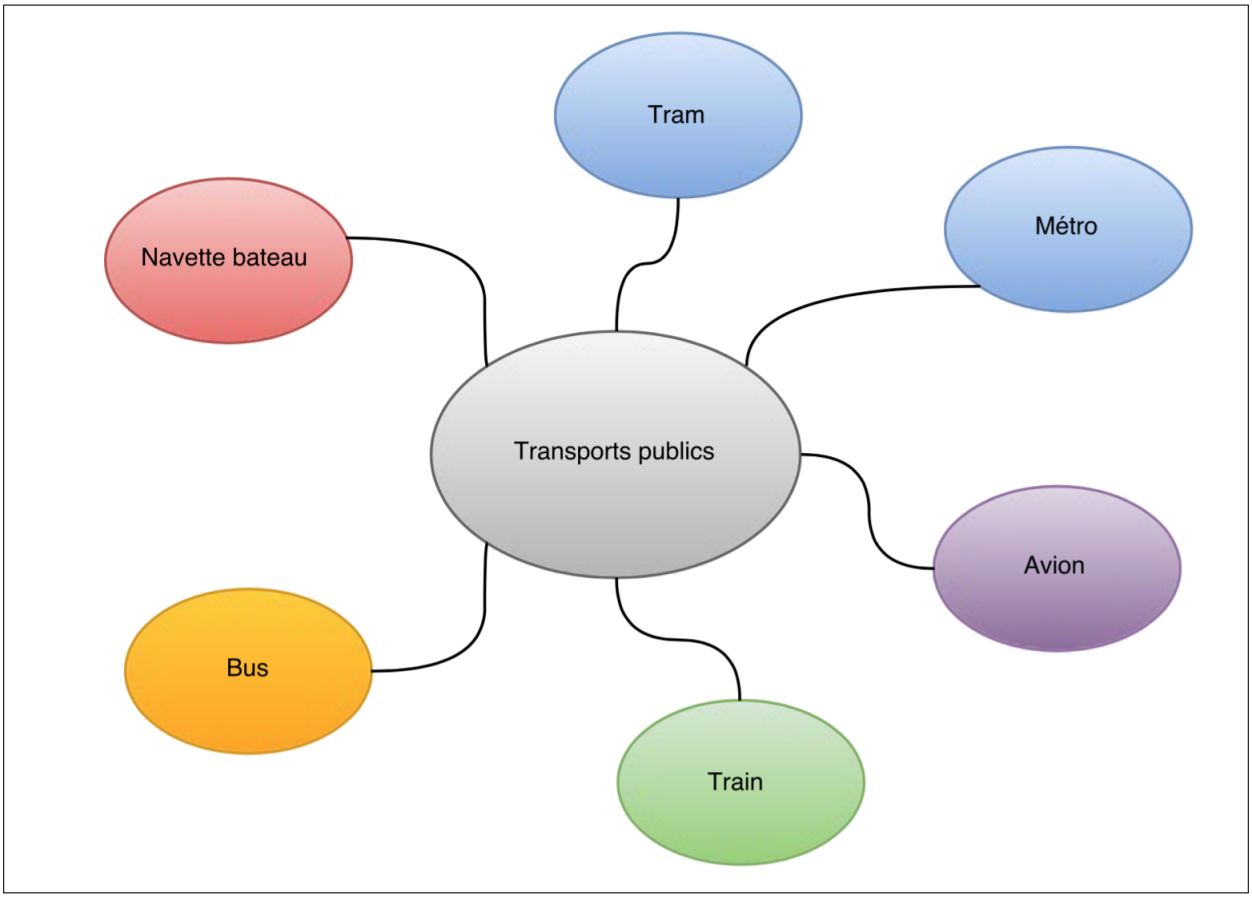

FIGURE 2.1 – Les transports

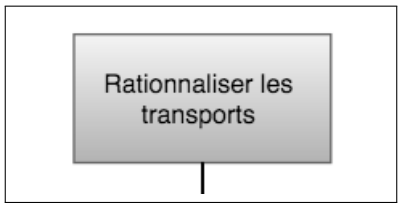

FIGURE 2.2 – UP : le niveau 0

1. Administrer le système
2. Administrer les comptes utilisateurs
3. Analyser les transports
4. Adapter les transports
5. Utiliser le système
6. Monetiser

2.1.5 Besoin non fonctionnel

— Nous allons travailler dans le cadre du « Big Data », ce qui implique une infrastructure moderne. Nous conseillons pour les débuts de l'application (à revoir si l'application prend

de l'ampleur rapidement) 4 serveurs (proc i3 — 4 gigas de ram — 500 gigas de mémoire data) pour traiter la masse considérable de données de manière distribuée.
— Toujours dans le cadre du « Big Data », l'utilisation d'une base de données MongoDB semble la plus appropriée. Cette base est en effet la plus simple à mettre en œuvre concernant des projets de cette ampleur.
— Disposer d'une application mobile et d'une application Web pour satisfaire le plus de clients possible.

2.2 Analyse

Nous allons maintenant analyser les risques attenants aux besoins fonctionnels décrits ci-dessus.

La fonction « monétiser » (fonction numéro 6 de la liste des besoins) est vitale ; sans elle, il ne serait pas possible de développer le système. Il faut donc avoir une approche réfléchie et définir avec soin le business modèle et les offres ; le business modèle doit être évolutif et à l'écoute du marché / des utilisateurs. En effet, une offre inadaptée ne permettra pas au système de s'adapter. Nous conseillons de faire une étude de marché complète, basée sur des sondages utilisateurs, des bêta-tests. Ceci étant, si le business modèle est adapté, il y a peu de chance que cette fonction pose un réel problème.

Les utilisateurs de votre système évolueront dans « l'utilisation du système » et « l'administration de leurs comptes » (respectivement fonctions fonction numéro 5 et 2 de la liste). Ils sont eux aussi vitaux dans votre application : sans utilisateurs payant, pas de revenus ! (et sans revenus, pas de produit.)

Analyser les transports et adapter les transports (fonction numéro 3 et 4) sera le cœur de l'application. Ce service sera fourni par le backend et les serveurs de votre infrastructure.

2.3 Élaboration

Les différents niveaux de UP seront explicités dans cette partie.

Le choix a été fait de détailler les niveaux en largeur et en profondeur sur le diagramme UP. Je souhaite en effet vous livrer une solution des plus précises ; la conception doit être schématiquement quasi exhaustive. Concernant les explications textuelles et les approches plus ciblées, je n'ai pas assez de budgets pour préciser toutes les fonctions. Je vais donc me focaliser sur les plus importantes à mon sens.

2.3.1 Conventions

Légende pour les diagrammes UP :

- En gris : N0
- En violet : N1
- En bleu : N2
- En vert : N3
- En jaune : N4

Formalisation : pour formaliser les niveaux textuellement, je vais préfixer les explications d'un petit (Nx) en parlant des fonctionnalités décrites, pour indiquer le niveau de UP (x) qui est concerné.

Note : Le diagramme est d'une taille assez conséquente. De ce fait, je vous invite à aller observer la version en ligne sur le site http://colinleverger.fr/assets/UMLGLOBAL.html.

2.3.2 Administrer son compte utilisateur

Définitions

Le trajet

Un trajet pourra être défini comme sur la figure 2.3 en page 11. Il sera composé d'un point de départ, d'un point d'arrivée, d'une heure de départ, d'une heure d'arrivée et d'un moyen de transport, qui sera choisi par l'utilisateur.

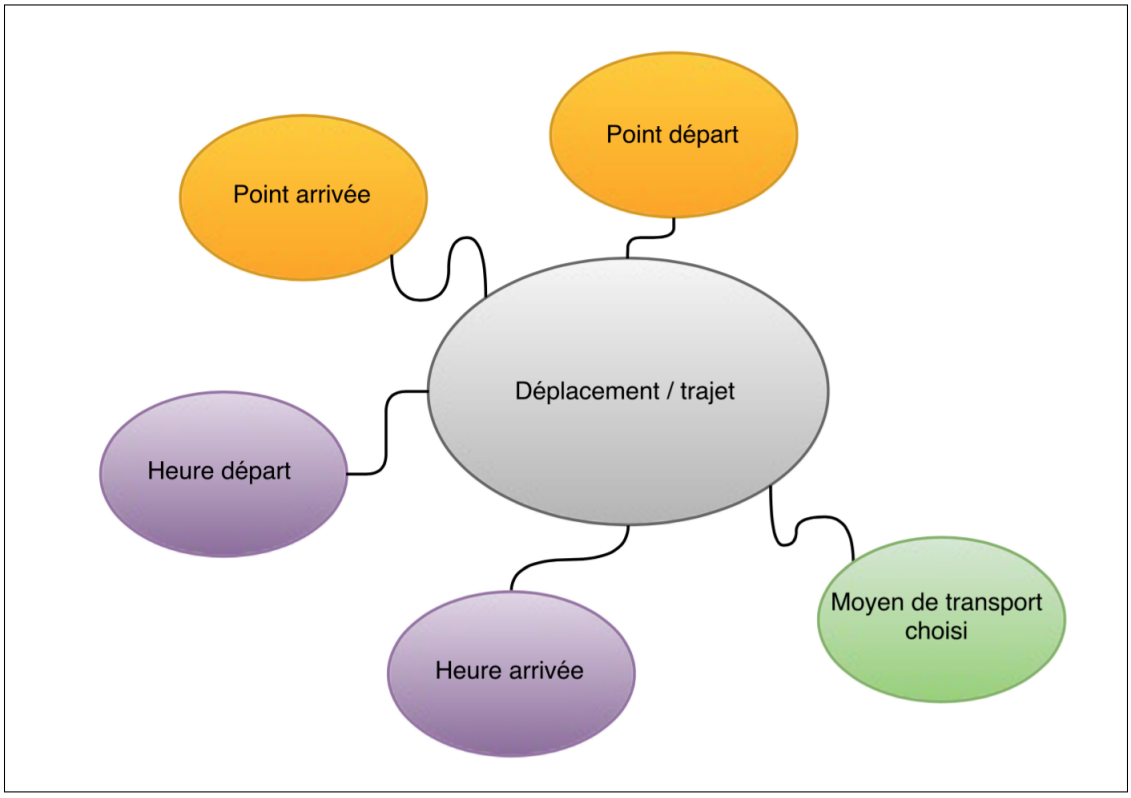

FIGURE 2.3 – Le trajet

Les besoins fonctionnels

Votre application ne pourrait pas exister sans ses utilisateurs. Les besoins fonctionnels s'axeront donc sur l'analyse de ce que peut faire un utilisateur avec le service. (À noter : un utilisateur peut aussi être administrateur.)

Dans la solution étudiée ici, il y a une distinction entre utilisateur « personnel » et utilisateur « publique ». En effet, la première catégorie représente les personnes comme vous et moi, ayant une utilisation des transports publics ou personnels ; les « utilisateurs privés » seraient les conducteurs

de bus / moyens de transport en commun. Typiquement, le choix a été fait ici de proposer aux conducteurs de bus une optimisation **automatique** de leurs trajets. Les trajets seraient anticipés et communiqués aux conducteurs (via Smartphones / GPS / à étudier) de manière automatisée en fonction de leurs moyens de transport et des trajets qui leur seraient assignés par leur employeur (la ville…).

Les besoins fonctionnels seront les suivants (sous forme de liste).

Dans le cadre d'un utilisateur simple :

- Créer un compte utilisateur.
- Demander le calcul d'un trajet entre un point de départ et un point d'arrivée.
- Modifier les variables et les préférences du compte utilisateur.
- Gérer ses moyens de transport favoris.
- Gérer ses trajets favoris.
- Partager ses futurs trajets.
- Intéragir avec la communauté / ses amis / réseaux sociaux.
- Gérer ses moyens de transport et sa formule (avec ou sans partage ?)
- Gérer ses moyens de paiement.
- Supprimer son compte.

Cas d'utilisation

Un simple visiteur du site / service ne pourra rien faire avec l'application. Tant qu'une personne n'est pas enregistrée dans le système, son champ d'action est totalement réduit (voir inexistant).

Fonctionnellement, il est important de proposer aux simples visiteurs un moyen de s'enregistrer comme utilisateurs du système, via un formulaire ou une page Web.

Une fois enregistré, un utilisateur pourra faire les actions décrites dans la figure 2.4 en page 13.

Les utilisateurs vont pouvoir enregistrer leurs moyens de transport préférés et leurs trajets préférés. En effet, si les utilisateurs font toujours les mêmes trajets avec les mêmes moyens de transport, il semble pratique de pouvoir leur proposer de manière automatique le trajet (gain de temps).

La fonction gérer son compte permettra à l'utilisateur de modifier les informations suivantes :

- Password

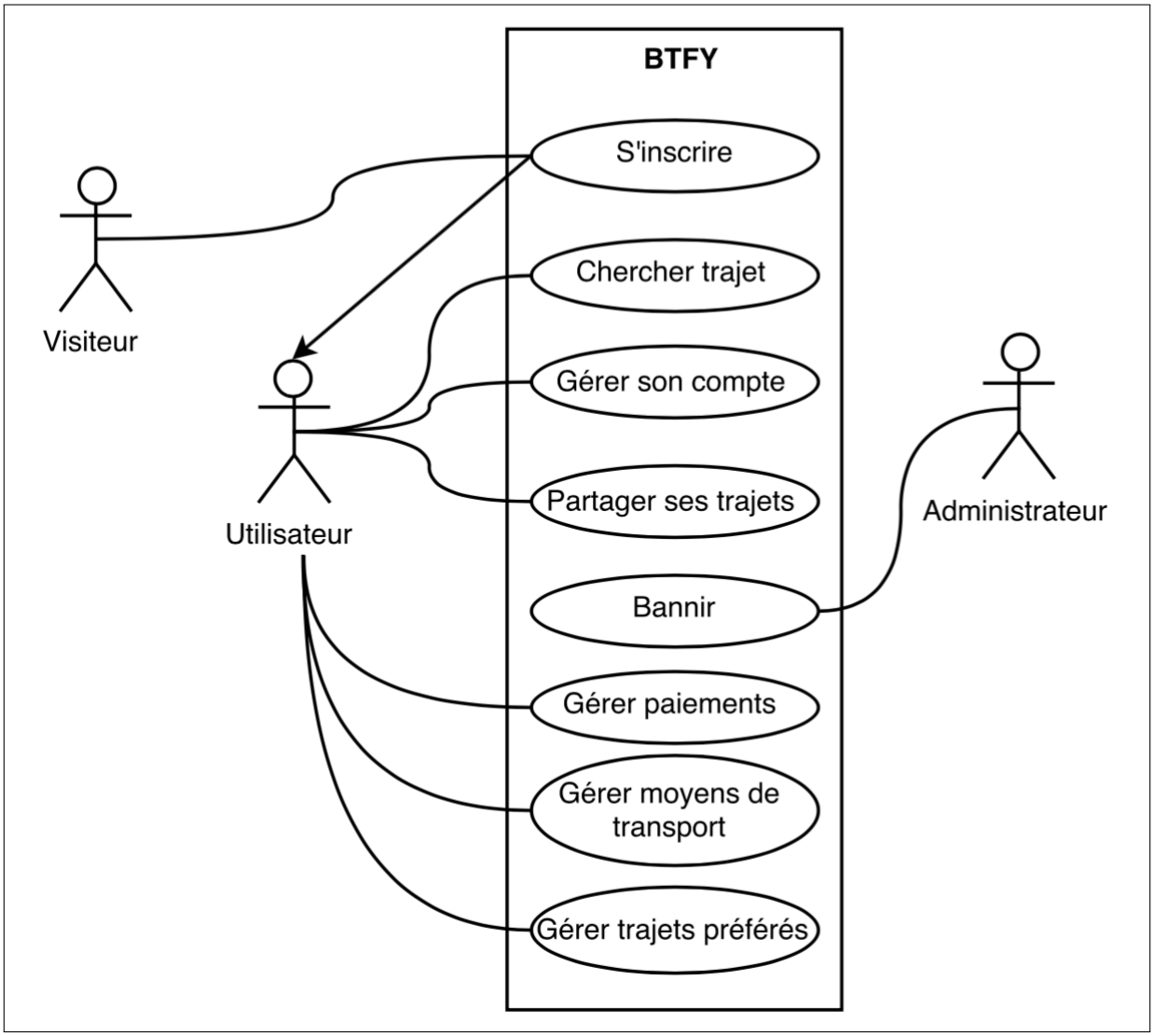

FIGURE 2.4 – Cas d'utilisation : l'utilisateur enregistré

— Nom / prénom / adresse
— Connexions sociales
— Photo / avatar

Un cas d'utilisation classique pourrait être celui de **Pierre**. Visiteur du site, il décide de s'inscrire pour tester l'offre. Il enregistre son moyen de paiement [3] et décide de gérer son compte. Il va mettre à jour sa photo de profil, son mot de passe et différents détails (adresse,...). Il enregistre ensuite ses moyens de transport préférés : voiture et bus. Il teste enfin le système en cherchant un trajet. Le premier trajet proposé par le système ne lui convient pas, il choisira le second. [4] Après ce trajet, il décide d'ajouter un troisième moyen de transport : le métro.

Pierre serait donc un utilisateur typique de l'application.

3. Il faut au moins un moyen de paiement valide pour pouvoir s'inscrire, sans ça l'inscription est abandonnée.
4. Le système va calculer des trajets alternatifs tant que l'utilisateur n'est pas satisfait.

Diagramme UP

Ce niveau (N1) est le plus important du système (en terme de niveau de UP horizhontal). Voir figure 2.5 en page 15.

Les niveaux détaillés ici concernent les actions pouvant être effectuées par un utilisateur.

Un niveau intéressant à détailler serait « gérer sa communauté » (N2). Il s'agit pour notre utilisateur de pouvoir ajouter / lier ses comptes sociaux (Google +, Facebook, ...) et de pouvoir gérer ses interactions avec ces derniers. Typiquement, il s'agit de pouvoir paramétrer le partage automatique de ses trajets avec ses connexions, de pouvoir voir les trajets de ses amis, etc...

Le choix a été fait de séparer les niveaux « changer mot de passe » (N2), « modifier mail » (N2) et « modifier données personnelles » (N2), car les deux premières sont plus critiques que la dernière. En effet, changer une photo n'a pas d'incidence sur le fonctionnement du compte utilisateur ; modifier son mail a plus de conséquences.

Le niveau « gérer la vie privée » (N2) se rapporte principalement au choix de l'utilisateur concernant la formule. Il s'agit de définir s'il veut bénéficier ou non de la réduction s'il partage ses trajets.

Risques

Cas du visiteur non enregistré :

Il semble crucial que le simple visiteur ne puisse pas utiliser le système avant de s'être inscrit. En effet, votre business modèle repose sur le fait que les utilisateurs paient pour utiliser le service. Sans cette source de revenus, votre application ne serait pas viable.

Le risque qu'un utilisateur non enregistré ait accès à l'application est très faible (quasiment risque 0). On considèrera donc que ce risque est un risque mineur.

Solution : Sécuriser l'accès à l'application et faire des *tests de pénétration du système* ; utiliser un prestataire de service pour faire ces tests.

Cas de l'utilisateur enregistré

Premièrement, il est important que chaque utilisateur ait un moyen de paiement **vérifié** avant de pouvoir utiliser l'application. Il ne faut pas que l'application accepte / valide une inscription sans

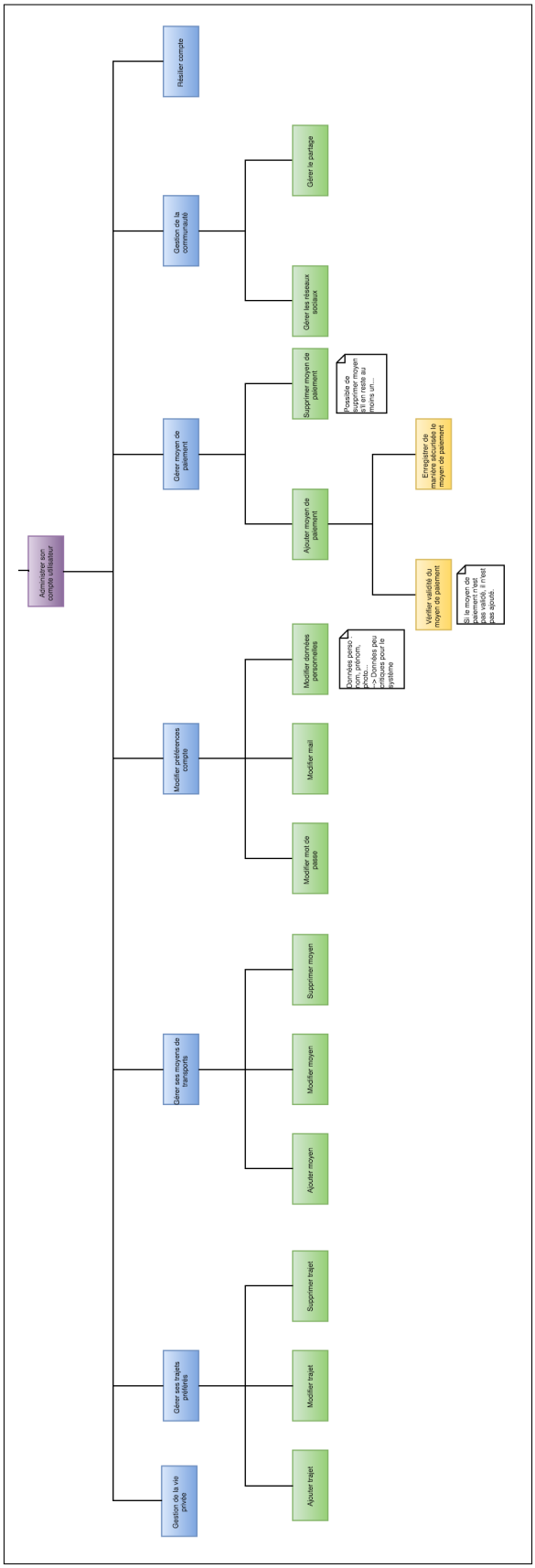

FIGURE 2.5 – UP : N1, administrer son compte utilisateur

moyen de paiement vérifié. Le risque est potentiellement haut : en effet, sans moyen de paiement, un utilisateur ne doit pas pouvoir utiliser le service (business modèle oblige !)

Deuxièmement, il faut que l'interface fournie aux utilisateurs soit assez simple d'utilisation pour que beaucoup d'utilisateurs s'y retrouvent. Risque plutôt moyen, mais à ne pas négliger, une mauvaise interface pourrait porter préjudice au développement de l'application

Troisièmement, il est important pour l'utilisateur de pouvoir modifier son profil simplement ; il s'agit là encore de maintenir l'expérience client au plus haut. Risque modéré, mais existant, un site qui ne fonctionne pas attire rarement les foules.

Solutions :

- Vérifier les moyens de paiement pour finaliser l'inscription (pas d'utilisation possible du système avant la vérification du moyen de paiement)
- Proposer une interface disponible sur plusieurs supports (PC, Tablette, Smartphone Android / Apple)
- Tester l'interface de manière complète (outils Selenium, Robot Framework…)

2.3.3 Administrer l'application

Définitions

La ville

Une ville pourra être définie comme sur la figure 2.6 en page 17. Les trois composantes principales seront les différents moyens de transport disponibles dans cette ville, sa taille et son nombre d'habitants.

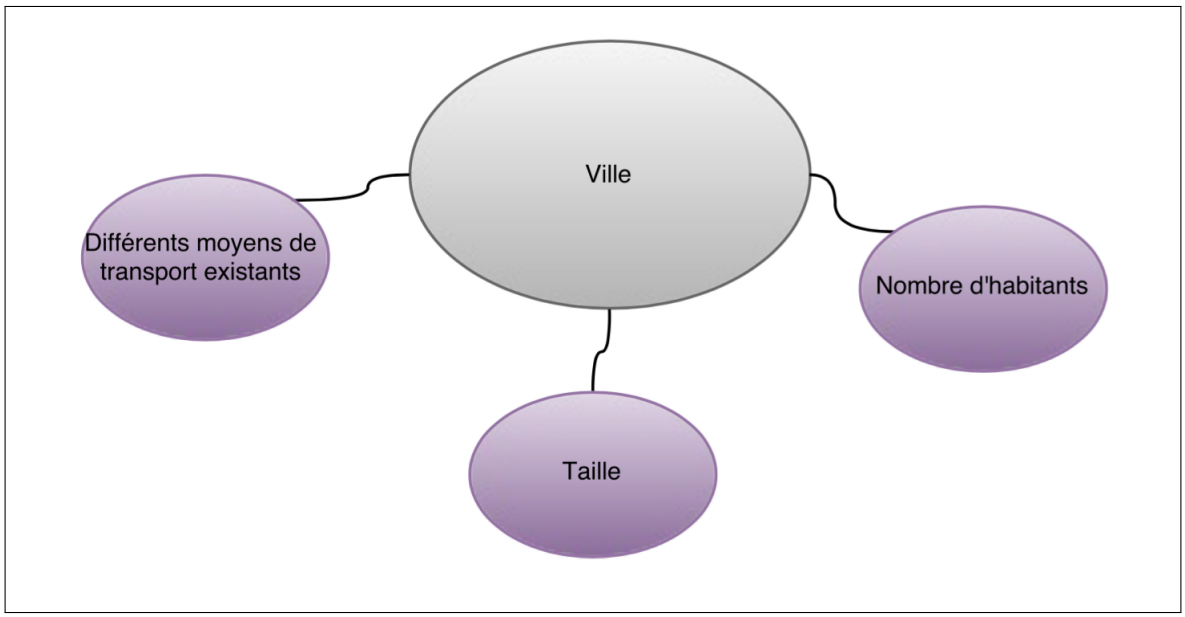

FIGURE 2.6 – La ville

Un évènement

Un évènement pourra être défini par une date de début, une date de fin, un nombre de personnes attendues. Typiquement, les évènements vont modifier le comportement des transports dans une ville pendant leurs occurrences. S'il y a beaucoup de personnes, cela va en effet rendre les transports plus compliqués.

Besoins fonctionnels

Dans le cadre d'un utilisateur administrateur :

— Gérer les évènements.
— Gérer la ville.

— Gérer les données.
— Gérer les moyens de transport public.
— Gérer l'application en elle même.
— Gérer les utilisateurs

Cas d'utilisation

L'administrateur, qui pourra aussi être un utilisateur classique du site en parallèle, pourra avoir les interactions vues en figure 2.7 en page 18 avec le système.

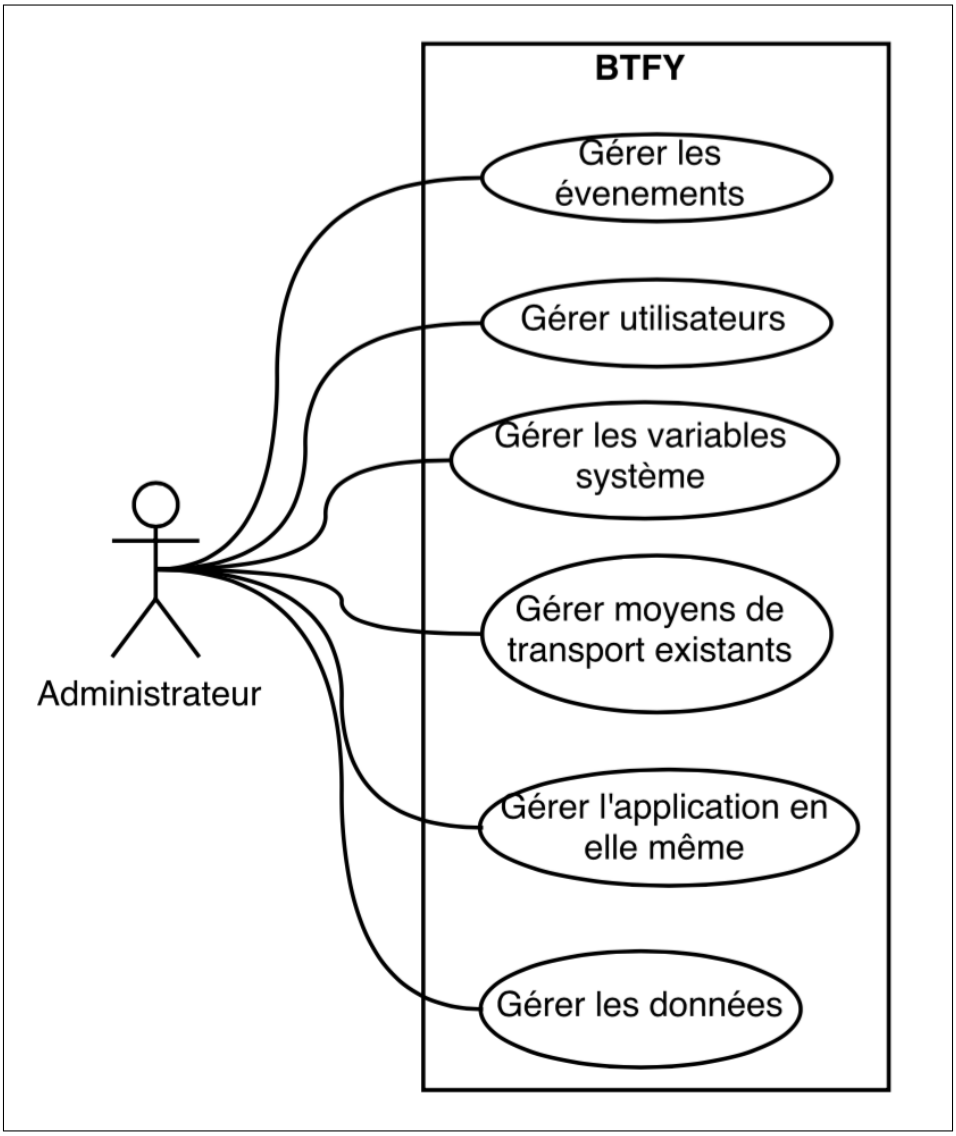

FIGURE 2.7 – Cas d'utilisation : l'administrateur

Les variables système géré par l'administrateur sont les suivantes :

— Prix / offres

— Historisation et délais de suppression des données
— Données attenantes à la ville (nombre de personnes, taille de la ville…)

Il semblait important que l'administrateur puisse gérer les utilisateurs du site afin d'avoir un aspect modération. Si **Pierre** décide de hacker le système ou d'effectuer des actions dangereuses et non désirées, il faut pouvoir le stopper et supprimer son compte.

L'administrateur peut gérer l'application en elle même et éventuellement la supprimer. Il gèrera aussi les données et l'historisation. Par exemple, les données pourront être gardées un an, deux ans, et tout ceci est paramétrable par l'administrateur.

L'administrateur pourra aussi effectuer des modifications sur des véhicules / moyens de transport existants. Il pourra aussi créer des nouveaux moyens de transport. En effet, la solution que vous livrez doit être **adaptable à toute situation** pour viser un large marché.

Enfin, l'administrateur va pouvoir ajouter des évènements (concerts, festival…) au sein de sa ville et indiquer quel est l'impact de cet évènement pour que le système le prenne en compte.

René est administrateur du site. Après son café le matin, il décide d'aller faire un tour sur le panel d'administration. Il observe des comportements anormaux sur deux utilisateurs qu'il s'occupera de bloquer du système. Son patron lui indique que les bus vont disposer d'une dizaine de places supplémentaires. **René** s'occupe de faire les modifications, puis ajoute dans le système l'évènement « fête de la galette » ; 10 000 personnes y seront attendues pendant 2 jours et l'impact sur les transports sera potentiellement important. Il décide pour terminer sa matinée bien chargée de changer l'historisation des données en la passant à trois ans au lieu de une année.

Diagramme UP

La première fonction de (N1) décrite ici sera la fonction administrer le système en figure 2.8 en page 20.

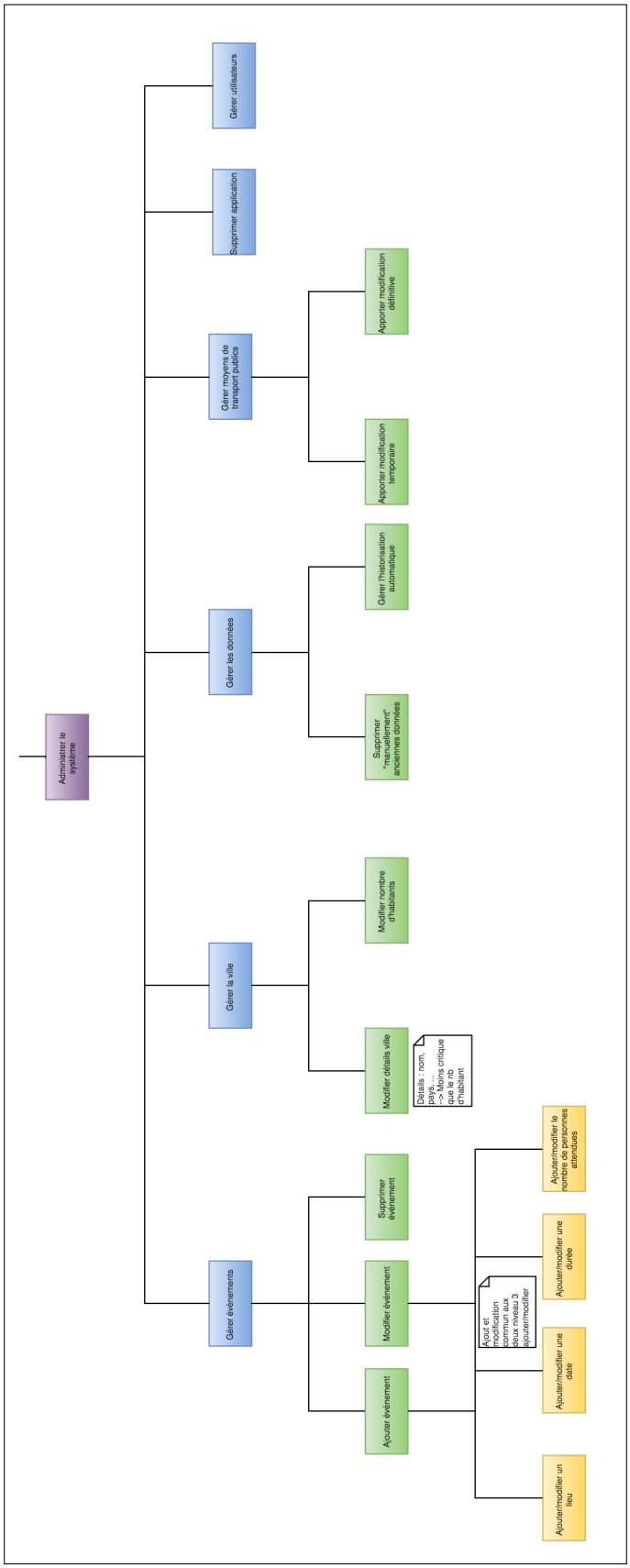

FIGURE 2.8 – UP : N1, administrer le système

Comme spécifié plus haut, il s'agit principalement de la décomposer en plusieurs sous fonctions de (N2).

« Gérer la ville » (N2) pourra se décomposer en deux niveaux ; il s'agit de pouvoir « modifier la taille de la ville » (N3) et le « nombre d'habitants » (N3) qui la composent.

On considère qu'un administrateur peut avoir des actions sur les transports publics d'une ville ; en effet, il peut modifier le nombre de places d'un bus, d'un tram...Mais pas d'un véhicule personnel. Cela n'aurait en effet pas de sens, c'est *à l'utilisateur* de pouvoir avoir une action sur son véhicule.

Un administrateur peut « gérer les évènements » (N2). Typiquement, il peut « ajouter un évènement » (N3), le « modifier » (N3) et le « supprimer » (N3). Le niveau « ajouter » (N3) et « modifier » (N3) sont intimement liés et auront les mêmes fonctionnalités en (N4) (à savoir, « ajouter un lieu », « ajouter une date », ...).

Sur la « gestion d'un moyen de transport public » (N2), il est possible de faire une « modification temporaire » (N3), ou une « modification définitive » (N3). Les modifications temporaires pourront être du type « diminution pour trois jours du nombre de places disponibles dans les bus... »

Risques

Il faut s'assurer que l'administrateur puisse gérer l'intégralité du système ; le risque est élevé, car sans administrateur, l'application pourrait être à la dérive et avoir beaucoup de problèmes. Il faut aussi sécuriser les interactions entre le système et l'administrateur. Le système doit être sécurisé et il ne faut pas qu'un hackeur mette en péril les données.

Sur la fonction « gérer les données » (N2), le niveau de risque est élevé. En effet, si l'administrateur se trompe dans ses actions il faut pouvoir revenir en arrière (si on perd 1 an de données à cause d'un souci de couche 8, cela pourra causer de gros problèmes pour votre application). Le même problème pourra être trouvé pour la fonctionnalité permettant de « supprimer application » (N2). Ce second risque est élevé, car « Errare humanum est » [5] et il est rapide de se tromper dans les manipulations, même pour un administrateur...

Solution : nous proposons (solution non fonctionnelle) de faire des sauvegardes massives de chaque serveur une fois par jour et de stocker les sauvegardes dans un stockage de type « glacier » (Amazon Web Services) [6] Afin de limiter les problèmes. Fonctionnellement parlant, nous proposons d'ajouter une sécurité / validation par un autre administrateur pour chaque action sur les données et sur le système.

5. **Saint Augustin** ; **164**.
6. Ref : https://aws.amazon.com/fr/glacier/

2.3.4 Analyser les transports

Définitions

Le contexte

Ici, le contexte représentera les évènements en cours dans la ville. S'il y a la fête de la galette et le Salon de l'auto en même dans dans la ville A, le contexte sera : deux évènements.

Trafic

Le trafic représente l'affluence de personnes sur les routes.

Besoins fonctionnels

Les besoins pourront être décomposés comme suis :

- Analyser les jours précédents.
- Analyser le contexte actuel.
- Analyser le trafic actuel.
- Anticiper les trajets futurs des usagers.

La nécessité de faire un diagramme d'utilisation ne se ressent pas dans ce cas. En effet, nous analysons actuellement une fonctionnalité centrée *backend*, c'est-à-dire que ces actions seront réalisées par un *serveur*.

Cas d'utilisation

Le diagramme d'activité pour cette fonction analysée pourrait être décrit en figure 2.9 en page 23.

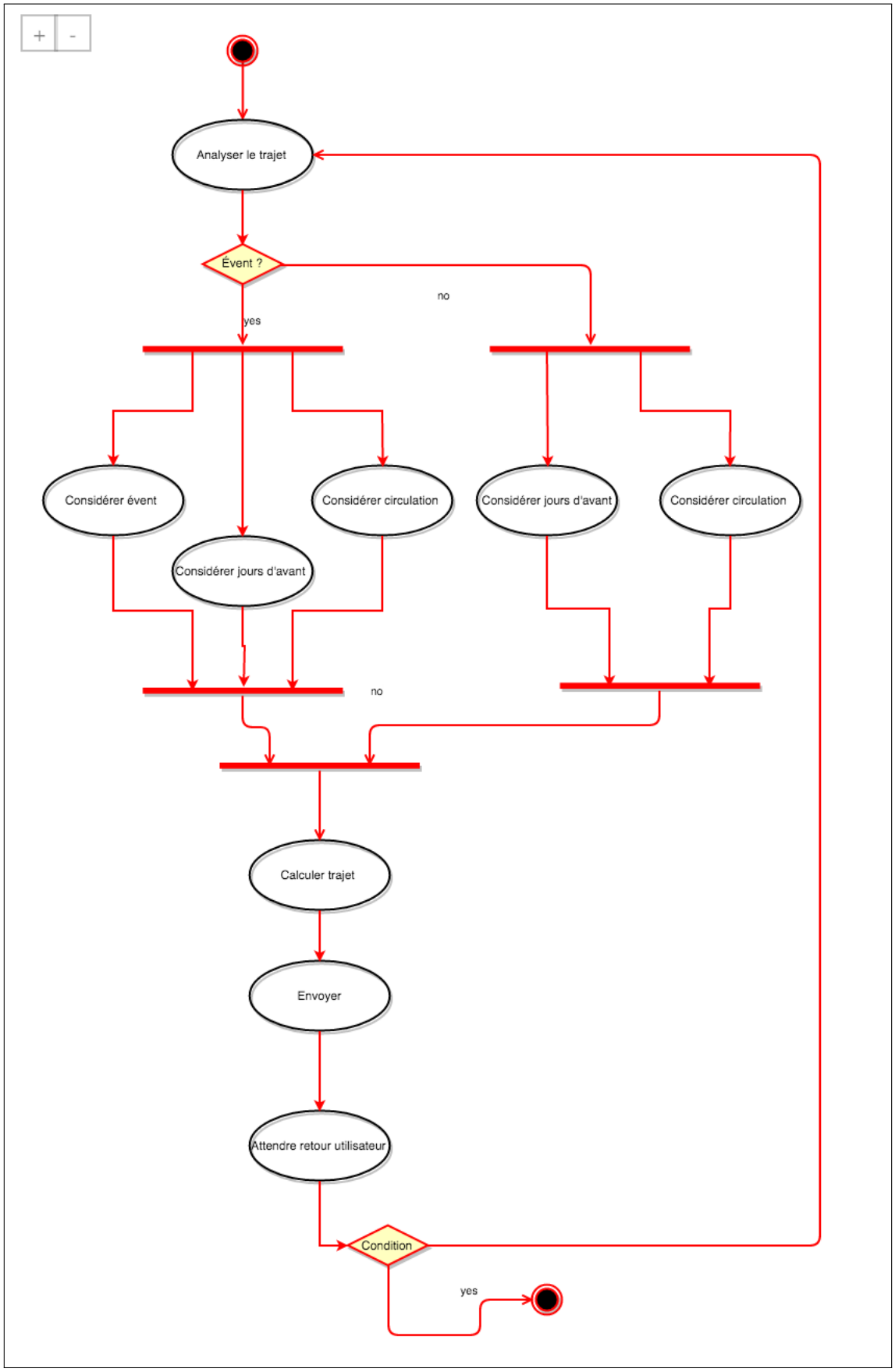

FIGURE 2.9 – Diagramme d'activité, analyser les transports

L'attente du retour utilisateur est importante. En effet, l'utilisateur va envoyer un retour à l'application : s'il est content du trajet reçu, il va l'accepter et l'application va pouvoir s'arrêter. Dans le cas contraire, l'application va devoir calculer un trajet alternatif, jusqu'à ce que l'utilisateur accepte le trajet proposé. Quand il sera accepté, le trajet sera automatiquement historisé et pris en compte par le système.

Diagramme UP

Le diagramme UP symbolisant l'analyse des transports pourra être vu en figure 2.10 en page 24.

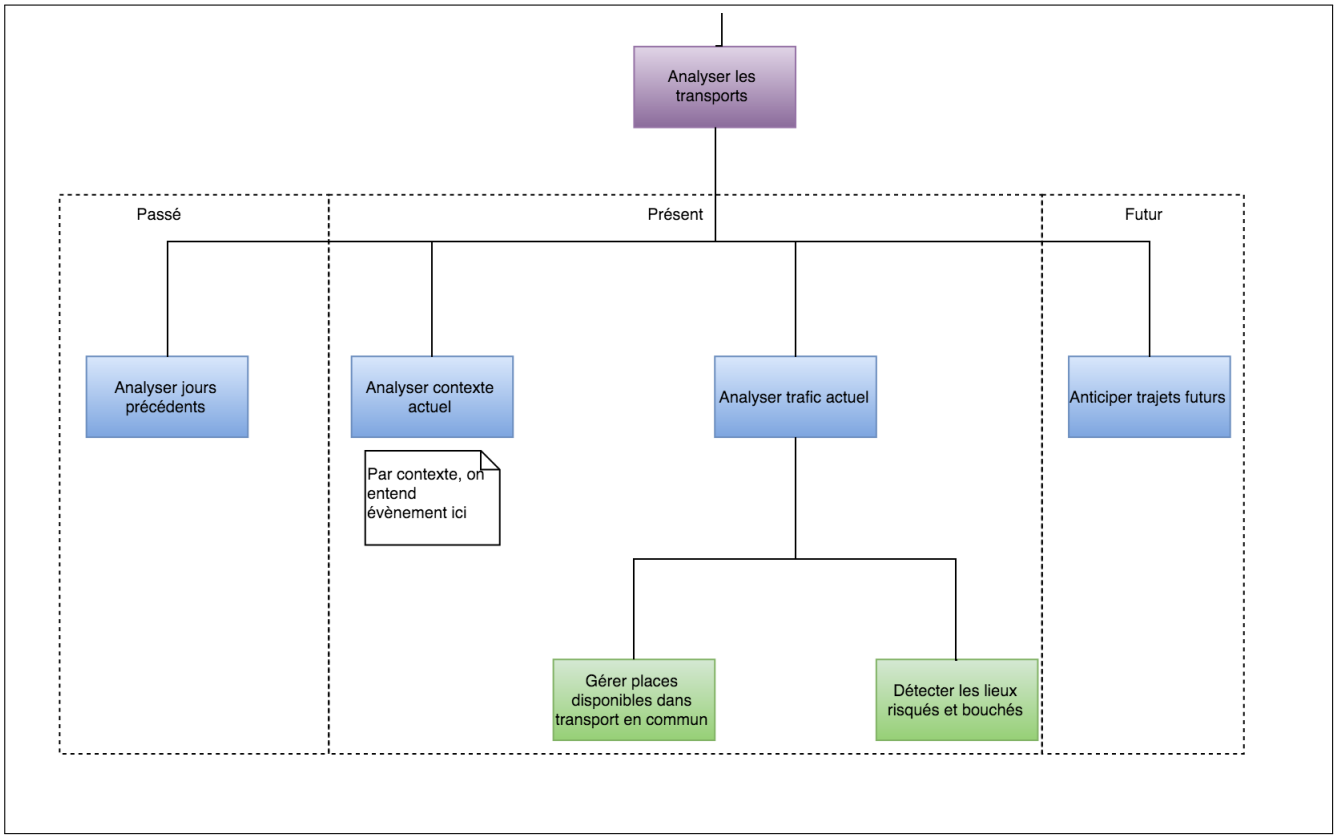

FIGURE 2.10 – UP : N1, analyser les transports

On observe que les niveaux (N2) sont divisés en 3 parties : Passé, Présent et Futur. En effet, le niveau (N1) « Analyser les transports » s'appuie sur des faits passés, présents et futurs. Il s'agit de « mixer » intelligemment les données disponibles pour générer des recommandations de transports. Les serveurs vont donc se reposer sur les données passées, le trafic actuel et les trajets futurs des utilisateurs ayant choisi de partager avec l'application pour générer les suggestions.

Dans le bloc « analyser trafic actuel » (N2), il s'agit typiquement de mettre à jour le nombre de

places disponibles dans les transports en commun [7] et de détecter les « endroits à risque » (à cause de bouchons, accidents, etc…).

À noter, les blocs (N2) en bleus pourront être effectués de manière *simultanée* par le système.

Risques

Concernant le (N2), analyser les transports

Si votre système s'arrête d'analyser les transports, cela risque d'être problématique. Cette fonction est critique, car sans analyse, pas de prédiction possible, et sans prédictions, votre système ne sert à rien à l'utilisateur ! Il faut donc minimiser les chances que cela se produise !

D'un premier abord, si l'application est robuste, ce problème ne peut se produire que du côté serveur. Si les normes de sécurités sont respectées et si le système est bien conçu, cela a peu de chance se produire.

Concernant le (N3), anticiper trajets futurs

Les utilisateurs ne vont pas tous partager leurs futurs trajets. Il faut d'attendre à manquer d'information sur ce plan-là (et cela risque d'arriver globalement assez souvent). Il faut donc considérer de facto que les données futures ne sont pas aussi importantes que les données déjà disponibles dans l'algorithme de prédiction.

7. Ce nombre est connu de l'application et est utile pour analyser les transports !

2.3.5 Adapter les moyens de transport

Définitions

Requêtes utilisateur

Lorsqu'un utilisateur a besoin d'un trajet, il va effectuer une recherche via l'interface et utiliser votre API pour rechercher un trajet. La demande de l'utilisateur sera la requête.

Besoins fonctionnels

Pour « adapter les moyens de transport » (N1), le système devra fonctionnellement :

- Recevoir les requêtes des utilisateurs.
- Préparer les trajets.
- Envoyer les trajets préalablement calculés (après l'étape de préparation)
- Sauvegarder les trajets envoyés (après l'envoi)

Cas d'utilisation

Le diagramme de séquence présent en 2.11 en page 27 décrit l'utilisation de cette fonction.

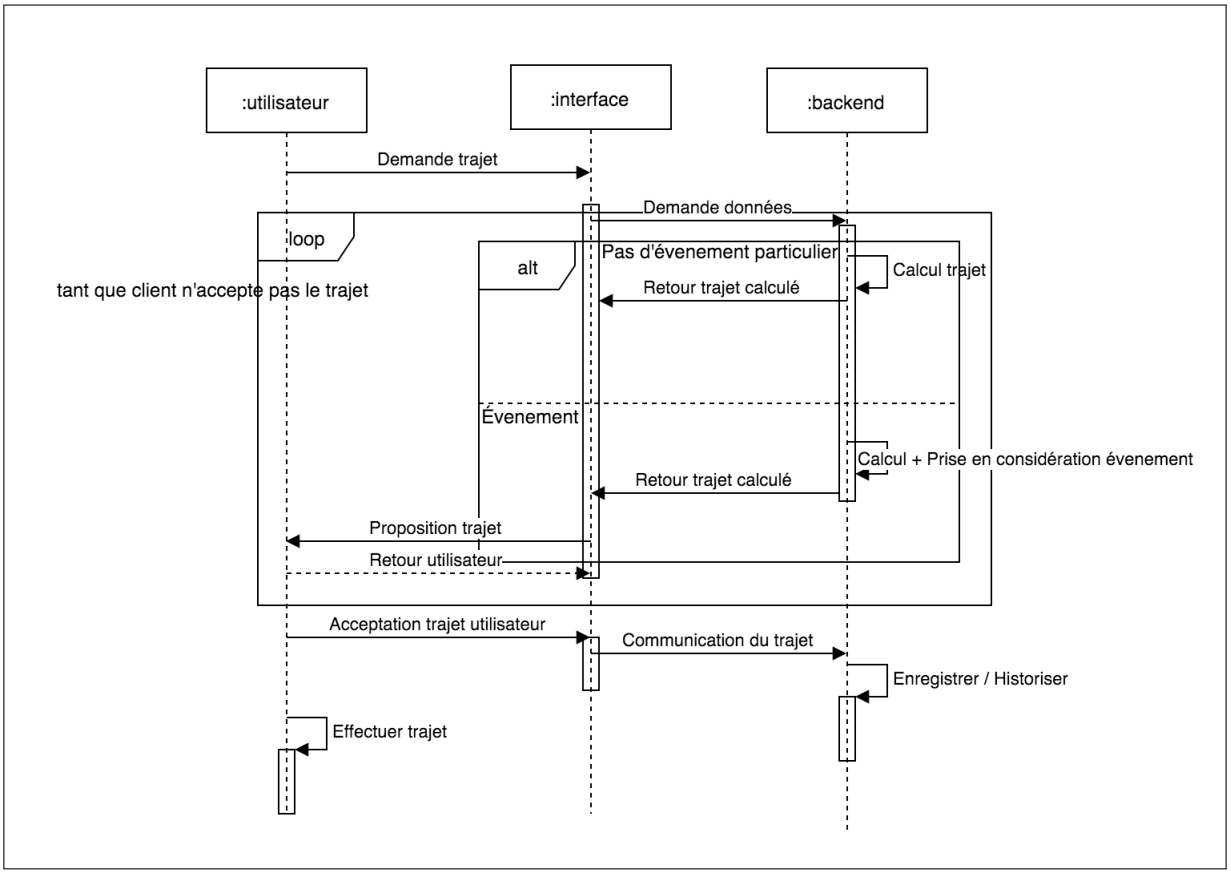

FIGURE 2.11 – Diagramme de Séquence, adapter les moyens de transport

Le « backend » est en fait représenté les différents serveurs qui feront les calculs.

Diagramme UP

Le diagramme UP symbolisant l'adaptation des moyens de transport pourra être vu en figure 2.12 en page 28.

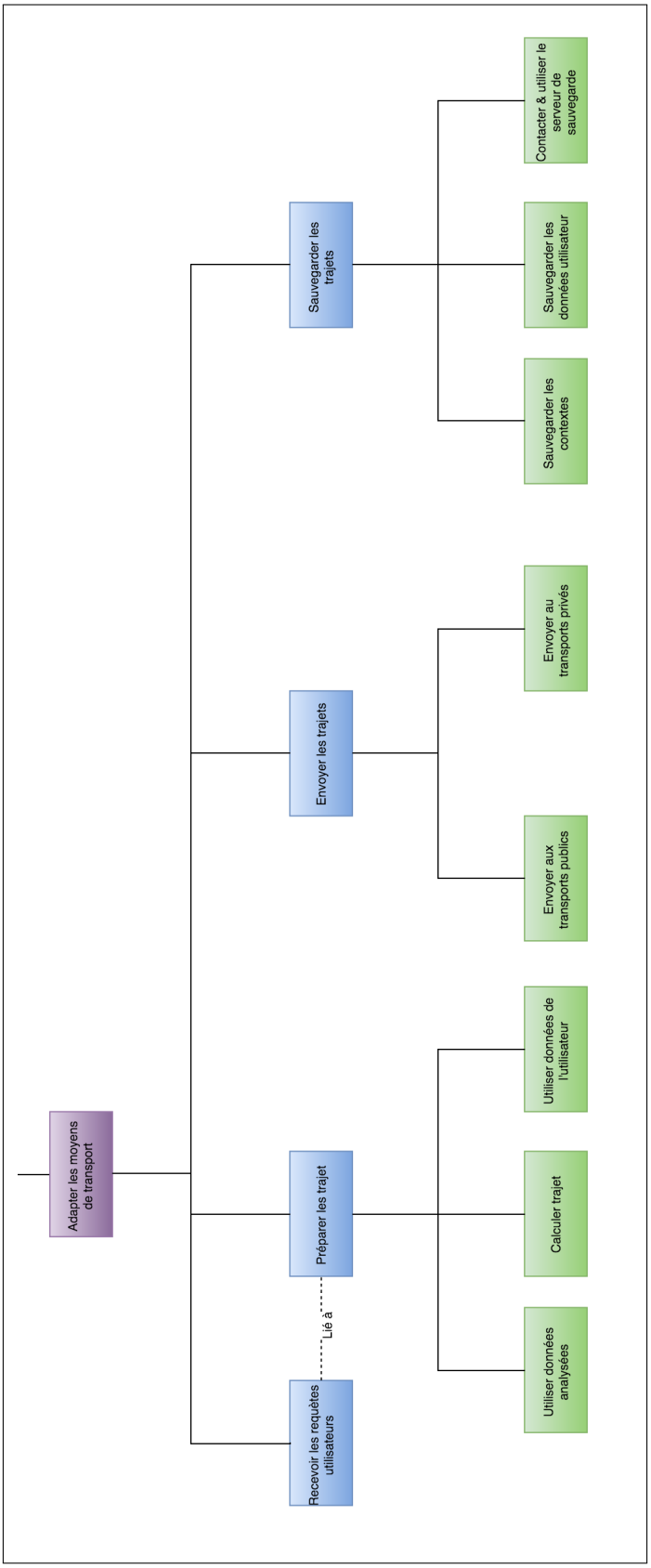

FIGURE 2.12 – UP : N1, adapter les moyens de transport

Les fonctions (N2) sont organisées séquentiellement, en allant de gauche à droite. En effet, le trajet ne peut être « calculé » (N3) et « analysé » (N3) qu'après avoir été reçu, émis par l'utilisateur ; il ne peut être « envoyé » et « sauvegardé » qu'après avoir été « préparé ». Cette hiérarchie est importante pour la compréhension du système.

« Sauvegarder les trajets » consiste en fait à historiser et garder en mémoire les trajets qui ont été acceptés (et donc effectués) par les utilisateurs.

Risques

« Adapter les moyens de transport » ne doit pas tomber en panne. En effet, c'est un des trois piliers de l'application (avec les niveaux « utiliser le système » et « analyser les transports ». C'est typiquement ce qui donne la raison d'être au système.

Un second risque à ne pas négliger est la sécurité des données. Il serait sage d'insister dans le marketing sur le fait que vous ne revendrez pas les données utilisateurs ; le business modèle exposé dans cette réponse à votre appel d'offres n'est en effet pas basé sur ce type de revenu, et cela pourrait être mal perçu par les utilisateurs, qui veulent contrôler leurs données personnelles. Il faut par ailleurs sécuriser un maximum les données. Ce risque peut être considéré comme moyen dans l'application ; un problème pourrait ternir l'image de votre marque.

2.3.6 Utiliser le système

Besoins fonctionnels

Dans ce niveau UP, il faut fonctionnellement que le système puisse :

- Permettre de chercher un trajet.
- Permettre à l'utilisateur d'accepter un trajet que le système lui a proposé.
- Permettre à l'utilisateur de partager ses trajets.
- Permettre à l'utilisateur d'interagir avec sa communauté.

Ce bloc est fortement dépendant du premier, « administrer son compte utilisateur » (voir section 2.3.2 page 11). En effet, on considère qu'une fois que l'ensemble du compte utilisateur est configuré correctement, on peut utiliser l'application et donc demander des calculs de trajets.

Diagramme UP

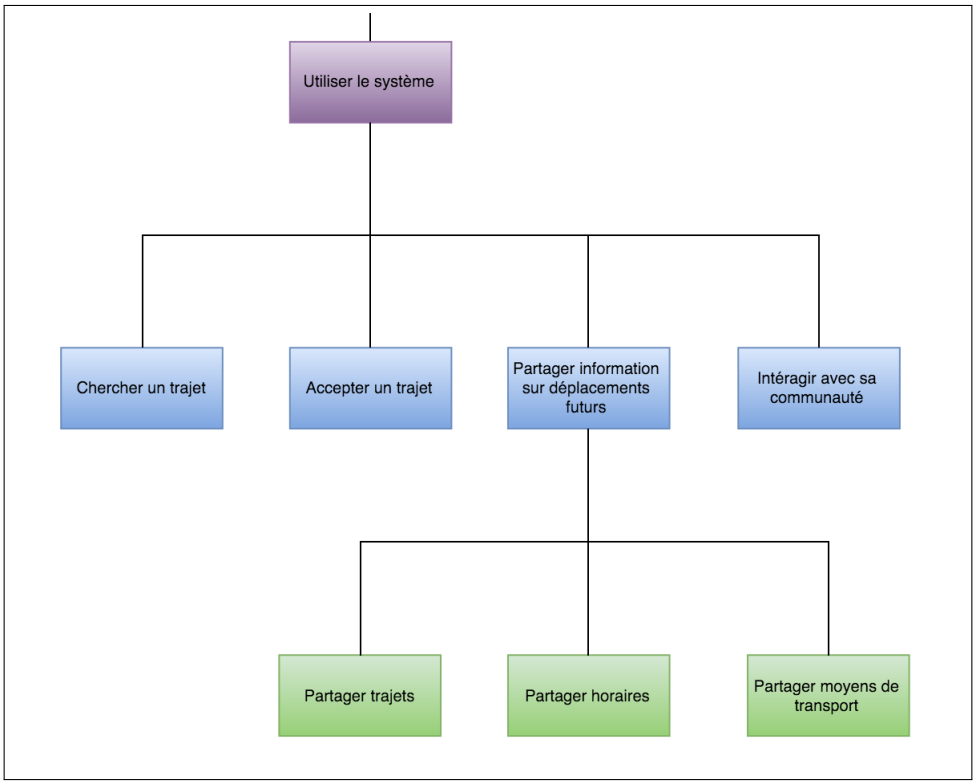

FIGURE 2.13 – UP : N1, utiliser le système

Risques

L'étude des risques n'est ici pas adaptée : en effet, le système en tant que tel n'encourt aucun risque…À ce que personne ne l'utilise !

2.3.7 Monétiser

Définitions

Business modèle

Les offres commerciales que vous proposerez. Abonnement ? Paiement en une foi de l'application ?

Données bancaires

Essentiellement numéro bancaire des clients, comptes PayPal,...

Aspects législatifs

Tout ce qui a un rapport de près ou de loin avec la législation (Française pour le moment).

Besoins fonctionnels

Pour « monétiser » (N1), le système devra fonctionnellement :

- Gérer le développement du système.
- Adapter les prix en fonction de l'offre choisie par les utilisateurs : partage ou non des données.
- Modifier business modèle.
- Gérer les données bancaires.
- Gérer les paiements.
- Gérer les aspects législatifs.

Diagramme UP

Diagramme UP disponible en figure 2.14 en page 33.

Le niveau « gérer le développement du système » (N2) est essentiellement basé sur l'évolution de votre produit. Quelles seront, à l'avenir, les nouvelles fonctionnalités proposées ? Il s'agit de

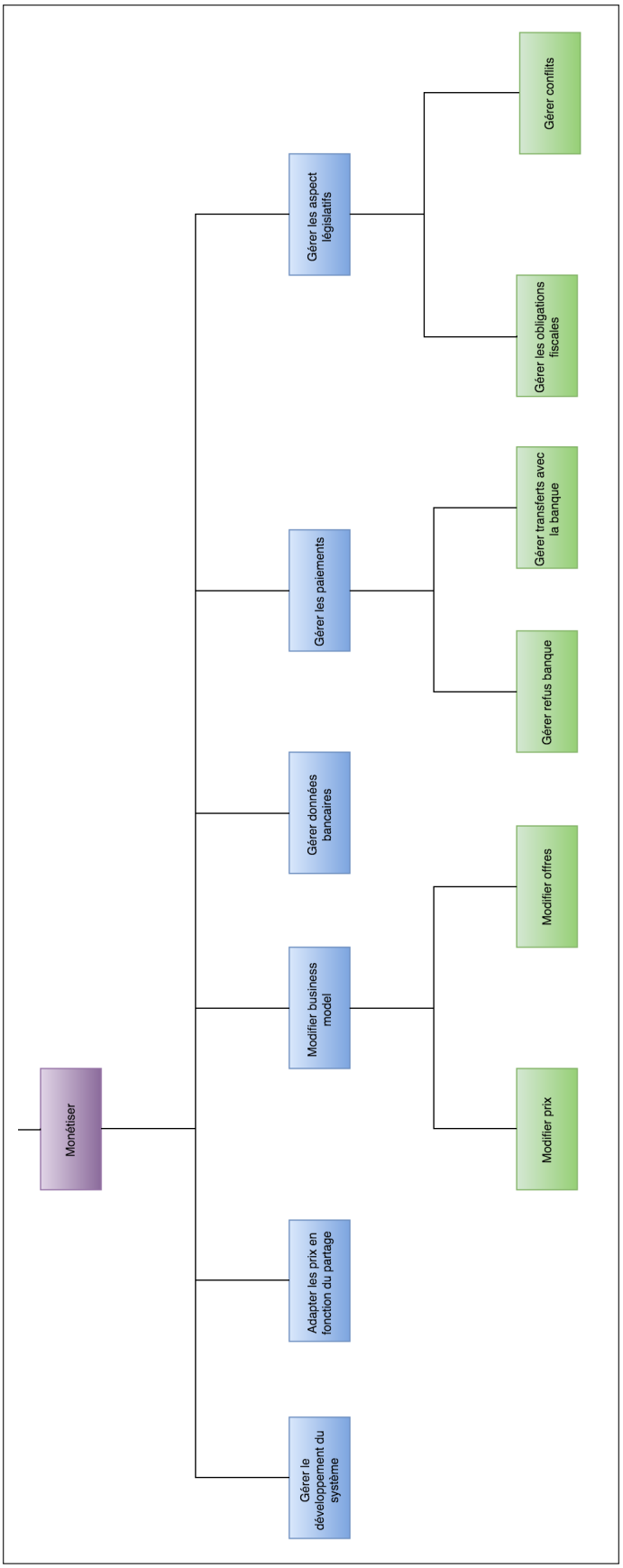

FIGURE 2.14 – UP : N1, monétiser

pouvoir avoir une action au sein même du système sur ces problématiques, qui se poseront tôt ou tard.

Le niveau « gérer les paiements » est important et permettra à votre application de survivre. Sans paiement, il n'est pas possible de maintenir l'application ! Il s'agira ici d'initier toutes les démarches auprès des banques concernant les paiements réguliers, mensuels, et de gérer cette problématique de la « récupération de l'argent due. »

Le niveau « gérer les conflits » (N3) est important. En effet, une application de la taille de la vôtre aura souvent affaire à des conflits, qu'ils soient financiers ou légaux. Il n'est pas possible de prévoir tous les cas possibles, et il arrive qu'un problème survienne en production : il faut pouvoir y faire face, c'est la fonction qui est décrite ici.

Notez que ce niveau (N1) « monétiser » est vital pour l'application, car il concerne directement le financement de cette dernière. Sans monétisation, pas d'application !

Risques

Le niveau « gérer donné bancaire » (N2) est très critique. En effet, s'il y a un problème, cela impacte considérablement l'application. Comment avoir confiance en une application non sécurisée et mal construite sur le plan des paiements ? Le risque est ici optimal.

2.4 Conclusion client

La conception détaillée de votre produit met en exergue certaines problématiques ; des choix seront attendus concernant les points suivants :

— Sécurisation des données.
— Évolution du business modèle et des offres proposées.
— Aspect « confiance » de vos utilisateurs vis-à-vis des données et de ce que vous en faites.

Le produit pourra certainement révolutionner la manière de se déplacer des gens, mais il faudra être attentif à ces problématiques. Une bonne communication pourra jouer en votre faveur concernant les points cités (sécurité, donnée utilisateurs…).

Je reste évidemment à votre entière disposition si vous avez des questions concernant le modèle que je vous fournis. J'espère pouvoir travailler avec vous pour de futures conceptions !

Partie 3

Conclusion

3.1 Retour sur l'estimation des couts

Pour rappel, les couts de l'étude étaient estimés comme suis :

- Entre 32h30 et 35h00 de travail en tout sur le projet (incluant choix du sujet, conception des diagrammes UP et des diagrammes divers / uses cases, rédaction du rapport).
- Entre 4 et 5 niveaux de UP.

À la fin du projet, voici la réalité des choses :

- 34H00 de travail en tout sur le projet, réparties comme suis :
  - 12 heures de recherche sur papier / esquisses / diagrammes.
  - 8 heures de réalisation sur ordinateur des schémas
  - 10 heures d'écriture de rapport
  - 4 heures de refactoring (replacer les éléments dans les bonnes sections)
- Entre 3 et 4 niveaux de UP.

Je considère que le cout n'a pas été dépassé. J'ai su m'organiser et ne pas aller trop loin / approfondir pour respecter au mieux ce budget. Il me semble important dans un projet de garder en tête qu'il n'est pas toujours possible, surtout en conception, de tout réaliser de manière parfaite. J'ai donc tiré parti du temps qui m'était imparti pour réaliser ce projet.

Concernant le nombre de niveaux de l'estimation, il est vrai que faire 5 niveaux de UP était un peu ambitieux. Je n'avais au moment de l'estimation que très peu d'idée sur le fonctionnement de la méthodologie UP et avec le retour d'expérience, j'aurais dû fixer entre 2 et 3 niveaux de UP. La

conception avec autant de niveaux m'a pris en effet beaucoup de temps, et était très certainement beaucoup trop précise pour les besoins de ce projet. J'ai néanmoins pris plaisir à décrire sous forme de diagramme le système. [1]

Je n'étais pas à l'aise avec la méthodologie UP d'un premier abord et j'ai donné trop d'importance à la première partie, à savoir l'élaboration. J'ai mis dans mon rapport beaucoup trop de détails dans cette partie. J'ai donc pris beaucoup de temps pour « refacturer » le rapport (4 h) mais cela m'a permis de mieux appréhender la méthode UP.

3.2 Retour d'expérience sur la méthodologie UP

Les points positifs de cette méthode sont qu'il est toujours agréable de travailler en mode itératif. Nous avons l'habitude de fonctionner de cette manière (beaucoup de méthodes sont itératives) et les repères sont faciles à prendre, une fois le concept compris et maitrisé. La phase la plus longue a été pour moi de bien comprendre ce que ce projet attendait de nous.

Il est important de rester exhaustif et organisé lors de la rédaction d'un rapport de conception ; en effet, il est courant que le rapport soit « divisé » entre plusieurs destinataires, il faut donc que les informations soient bien réparties de manière à ce que chaque personne, avec ses compétences propres, puisse trouver l'information qui le concerne au bon endroit. Pour la synchronisation d'une équipe de 3 personnes, cela n'a pas beaucoup de sens, mais dans une équipe de 100 personnes (avec des 10aines de métiers confondus), cela est capital pour la bonne réussite du projet.

Le point négatif est qu'il est très difficile de ne pas faire de redites. Organiser ses propos et ses schémas pour ne pas avoir de redites n'est pas facile d'un premier abord, et modifier 6 diagrammes complexes après plusieurs heures de travail n'est pas toujours aisé…L'habitude de ce genre de méthode doit aider dans la visualisation à l'échelle macroscopique d'un projet, mais nous n'avons pas encore l'habitude en deuxième année et cela peut être déroutant.

Aussi, j'ai eu tendance à préciser les diagrammes UP « à l'extrême » ; cela a impliqué qu'il m'a été relativement difficile d'expliquer textuellement chaque niveau de manière extrêmement précise, au vu du temps qui m'était imparti. Je pense que si j'avais à refaire le projet, je ferai moins de niveaux UP (en largeur). Ceci dit, mon diagramme UP me semble assez explicite, même sans explications textuelles.

Je retiendrai que ce projet a été une ouverture sur le monde de la conception et m'a permis d'appréhender les concepts et les difficultés de ce domaine.

1. …Et je pense très sérieusement monter ma Startup en utilisant votre projet ;-)

Table des figures