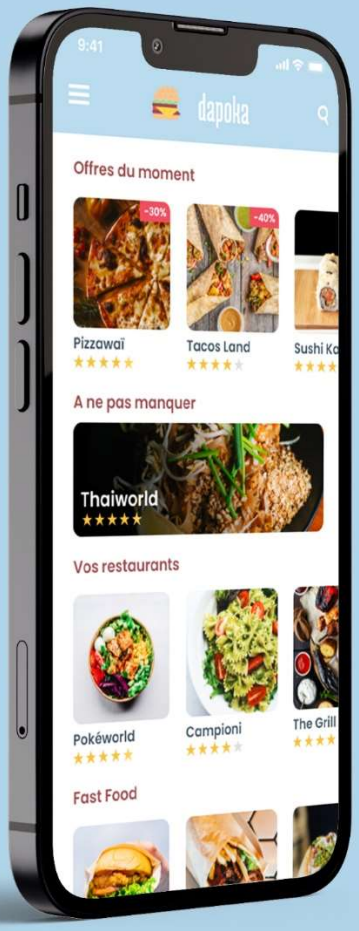
cesi ÉCOLE D'INGÉNIEURS

2022



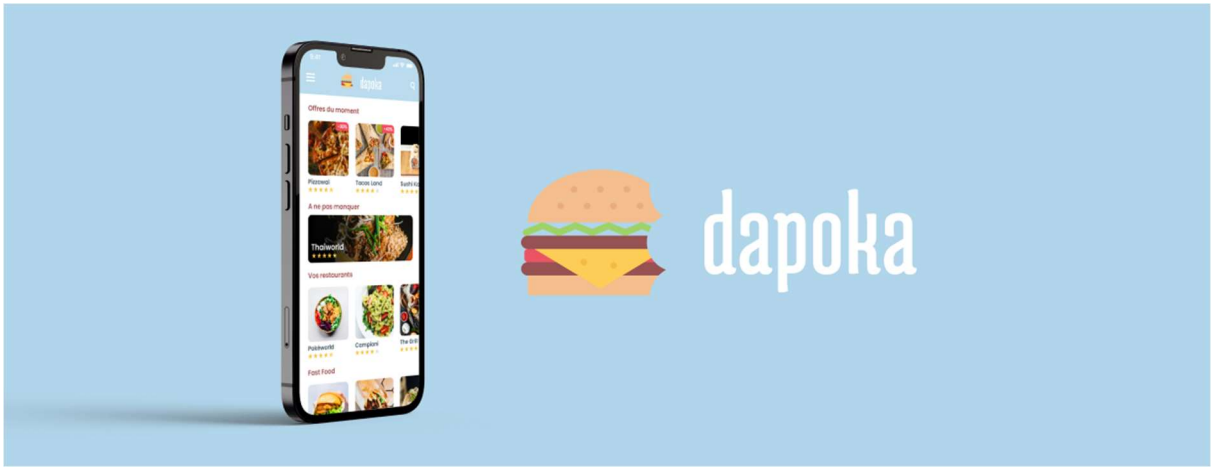
Rapport d'analyse du cahier des charges

Arthur LECRAS - Lucas AGNES - Dorian BUQUET

dapoka

Introduction

Notre équipe de production s'est vu attribuer un nouveau projet : développer une application de livraison de restauration à domicile. Cette plateforme a pour mission de faire le lien entre les restaurateurs qui vendent leurs plats, les livreurs qui doivent transporter les commandes et les clients qui souhaitent se restaurer depuis leur domicile. Cette plateforme doit être simple d'utilisation et intuitive pour ses différents types d'utilisateurs.



Nous avons décidé de baptiser ce nouveau service de restauration à domicile au nom de Dapoka. Provenant du grec ancien, de la contraction de "dô" et "apokathistēmon" (δῶ αποκαθιστημον) qui signifie "lier restaurant", la toute nouvelle plateforme Dapoka a pour ambition de devenir le vrai lien entre les consommateurs et les restaurateurs. Ces derniers ont énormément souffert durant la récente crise sanitaire mondiale, et avec les restrictions d'interdiction de déplacement des clients, il est devenu très difficile pour eux d'éviter la faillite. Dapoka facilitera donc la vente en apportant aux restaurateurs une nouvelle clientèle sans avoir besoin de se rendre sur place.

Sommaire

Présentation de l'équipe

Avant d'entrer dans le vif du sujet, voici une brève présentation des membres de notre équipe de projet menée par Arthur LECRAS. Comme vous pourrez le constater, nos profils différents vont pouvoir se compléter afin de mener notre projet à bien, une véritable richesse pour Dapoka.

Arthur LECRAS

Âge : 21 ans

Spécialités :

- Logiciel : C# .net
- Data : SQL, NoSQL
- Développement WEB (Front et Back) : Node, TypeScript…
- Déploiement et intégration continue
- GIT : GitLab et GitHub
- Capacité d'assimilation importante et rapide

Lucas AGNÈS

Âge : 21 ans

Spécialités :

- Conception d'interface utilisateur
- Spécialisation en mise en place d'expérience utilisateur
- Création d'identitée visuelle
- Développement WEB (Front)
- Gestion des RGPD et Mentions légales
- Présentation de projet et réponse d'appel d'offre

Dorian BUQUET

Âge : 22 ans

Spécialités :

- Conaissance du PHP & du SQL
- Expérience en développement d'application chez le client
- Développement WEB (Back)
- GIT (tortoiseGIT)
- Développement d'API en .net Core/Framework

Identification des dates importantes

Après l'analyse des différentes phases décrites dans le cahier des charges, nous avons construit le diagramme de Gantt suivant :

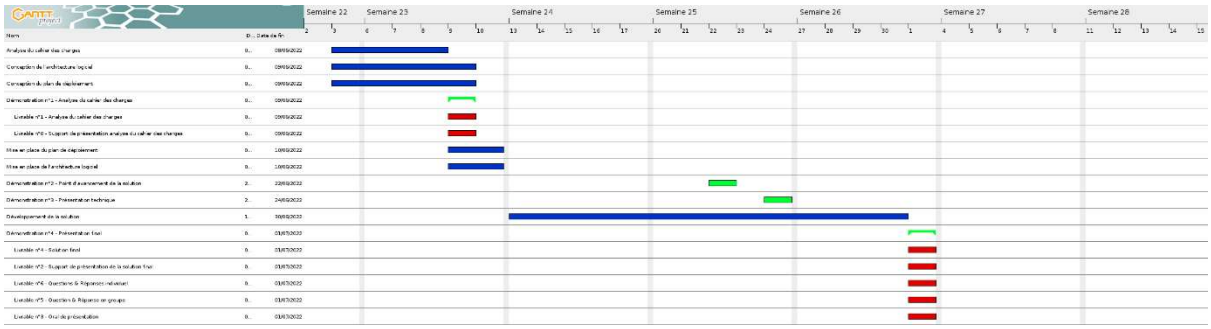


Figure 1 : Diagramme de Gantt prévisionnel et récapitulatif des phases du projet.

Sur le diagramme précédent, nous avons mis en évidence les phases de travail en bleu, les rendus de livrables en rouge et les démonstrations en vert.

Nous avons repéré quatre démonstrations :

1. la première s'effectuera le jeudi 9 juin, elle représentera le livrable n°0 et 1,
2. la deuxième se déroulera le mercredi 22 juin, nous y présenterons l'avancée du projet,
3. la troisième se passera deux jours après la première, le vendredi 24 juin et fera l'objet d'une démonstration technique,
4. la quatrième sera le vendredi 1 juillet, la présentation finale du projet ainsi qu'un récapitulatif éventuel des tâches non accomplies. Cette présentation fera l'objet des livrables 2 à 6.

Nous avons également décelé sept livrables :

1. livrable n°0 : le document récapitulant l'analyse du cahier des charges, à rendre le jeudi 9 juin,
2. livrable n°1 : le support de présentation montrant l'analyse du cahier des charges, à rendre le jeudi 9 juin,
3. livrable n°2 : le support de présentation de la présentation final, à rendre le vendredi 1er juillet,
4. livrable n°3 : l'oral de présentation, se déroulera le vendredi 1er juillet,
5. livrable n°4 : la solution finale, à rendre le vendredi 1er juillet,
6. livrable n°5 : les questions et réponses en groupe, se déroulera le vendredi 1er juillet,
7. livrable n°6 : les questions et réponses individuelles, se déroulera le vendredi 1er juillet.

Organisation du groupe et outils

Organisation générale

Pour l'organisation générale du projet, nous avons opté pour la méthodologie Agile Scrum, car elle répond totalement aux besoins cycliques de gestion entre les prosits et les différentes phases de projets comme nous pouvons l'observer sur le schéma suivant :

Figure 2 : Schémas représentant les prosits et les phase de projet.

Pour planifier et organiser le projet avec la méthodologie Scrum, nous avons utilisé Trello pour l'organisation et la répartition des tâches, et Gantt pour effectuer un planning prévisionnel.

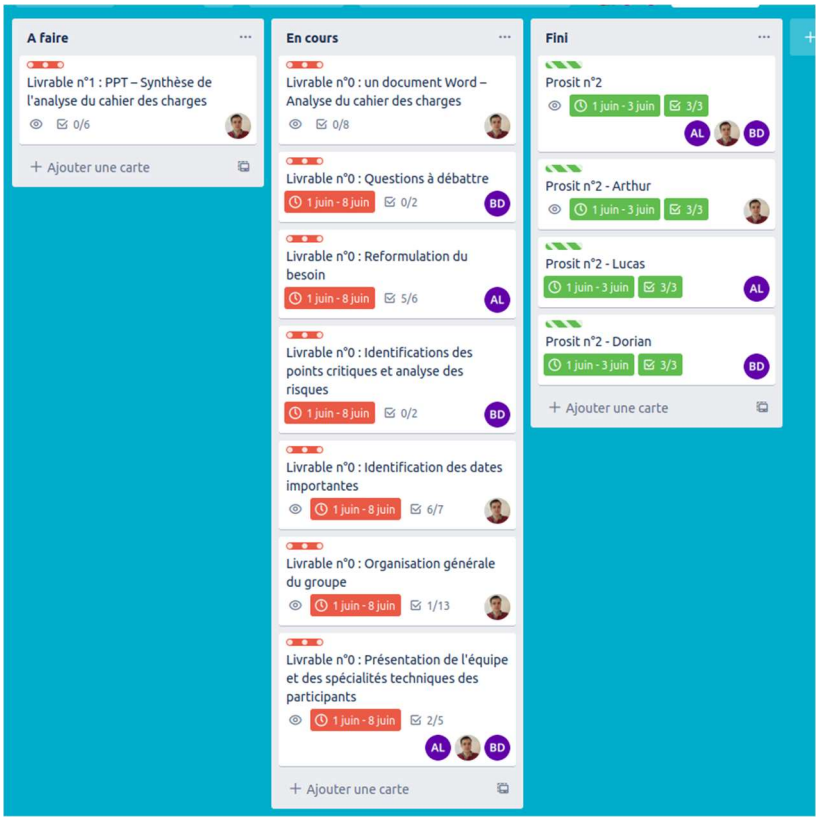


Figure 3 : Image du Trello

Pour pouvoir communiquer ou débattre tout le long du projet, nous utilisons l'outil collaboratif discord.

Durant ce projet, nous aurons besoin de produire plusieurs documentations, pour cela, nous utiliserons la suite office 365 dont les documents résultants sont stockés sur un OneDrive afin de pouvoir y travailler à plusieurs. Un autre outil vient s'ajouter aux documents dans le but de produire des schémas explicatifs : Draw.Io.

Avant de commencer à construire les différents modules du projet pour obtenir l'application fonctionnelle, nous devons passer par plusieurs phases de réflexion. La première n'est autre que la phase de conception, et nous y utiliserons l'outil Visual Paradigm pour produire les différents diagrammes UML. La seconde phase de réflexion concerne le design des différentes interfaces graphiques et pour cela, nous utilisons l'outil collaboratif Figma.

À la suite de ces phases de réflexions, nous pourrons commencer à créer la solution technique, nous aurons besoin de plusieurs IDE, Visual Studio code et WebStorms selon les préférences de chacun pour le développement WEB et de Visual Studio pour développer l'application C#. Pour centraliser le code sources des solutions, nous utilisons le système de gestion de version Git héberger sur GitHub, l'outil Git kraken viendra en complément afin d'appréhender plus facilement Git.

Organisation du projet

Plan de développement

L'architecture générale et logicielle du projet nous est imposée, celle-ci doit respecter un équilibre entre une architecture micro-service et orienté service, respectant l'architecture logiciel "Layered pattern".

Dans le cahier des charges nous avons :

Services	Quoi
Gestion Authentification	Créer compte : création, validation du compte
	Connexion compte : vérification, remember (token), double authentification
	Déconnexion
Gestion comptes	Modifier : modifier email, informations personnel, gestion moyen paiement...
	Supprimer
	Suspendre
Gestion restaurants	**Articles (plat, boisson, sauce, accompagnement, etc...) :**
	Créer
	Modifier
	Supprimer
	Menus (composition d'articles) :
	Créer
	Modifier
	Supprimer
Notifications	Envoi de mail
	Messages envoyés aux applications
Gestion Paiement	Payer une commande
	Créer une demande de remboursement
	Modifier une demande de remboursement
	Visualiser une demande remboursement
Gestion commandes	Créer : ajouter
	Modifier : ajouter/enlever articles, valider
	Supprimer : refuser ou annuler
	Visualiser une commande
	Visualiser l'historique des commandes
Gestion livraisons	Créer : adresses
	Modifier : localisation, adresses, accepter
	Supprimer : refuser ou annuler
Gestion promotion	Créer
	Appliquer
	Supprimer
Gestion statistique	**Restaurant :**
	Nombre de comande/jours
	Produits les plus vendues
	Commercial :
	Restaurant le plus vendeurs
	Technique :
	Connexions
	Téléchargement des composants
Supervision	Afficher l'ensemble des commandes en cours
	Performances serveurs
	Ajouter, supprimer des composants réutilisables
	Télécharger un composant

Figure 4 - Tableau des micro-services.

L’architecture générale et logicielle peut-être ainsi modélisée de la manière suivante :

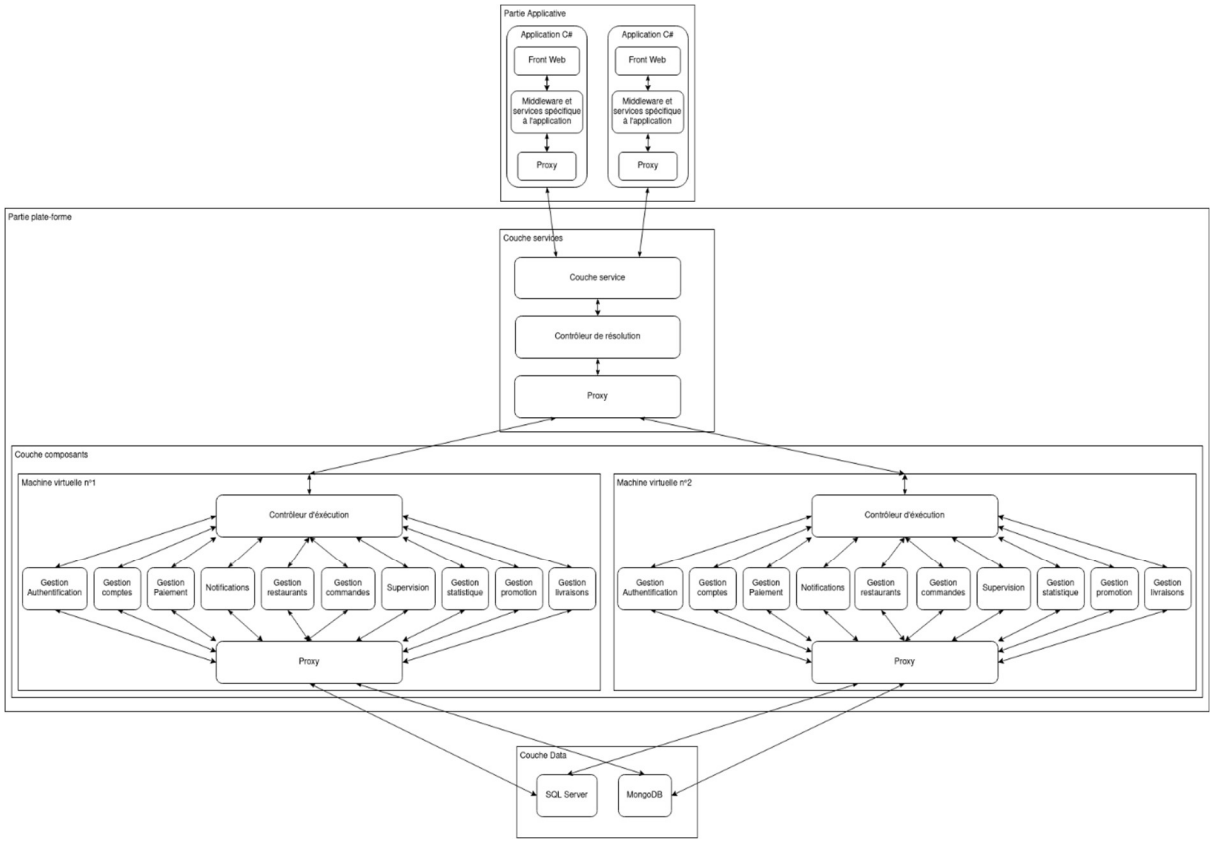


Figure 5 : Schémas de l’architecture technique avec les micro-services.

Plan de déploiement

Stratégie (ou Architecture) de déploiement

Nous avons défini plusieurs critères pour définir un déploiement sans soucis :

- n'affecte pas ou très peu les utilisateurs de nos services, exemple, nous actualisons le système de paiement, les utilisateurs devront pouvoir continuer à payer,
- être capable de faire marche arrière en cas de problème, ou de non-satisfaction des modifications apportées par le déploiement d'une nouvelle version, exemple changement de design d'une fonctionnalité.

Pour répondre à ces deux contraintes, nous avons jugé que le déploiement canary correspondait le plus à ces contraintes. Il présente, cependant, le principal inconvénient d'être complexe et coûteux à mettre en place. Par ailleurs, il fonctionne en répartissant, une proportion d'utilisateurs sur l'ancienne version et la proportion restante sur la nouvelle version.

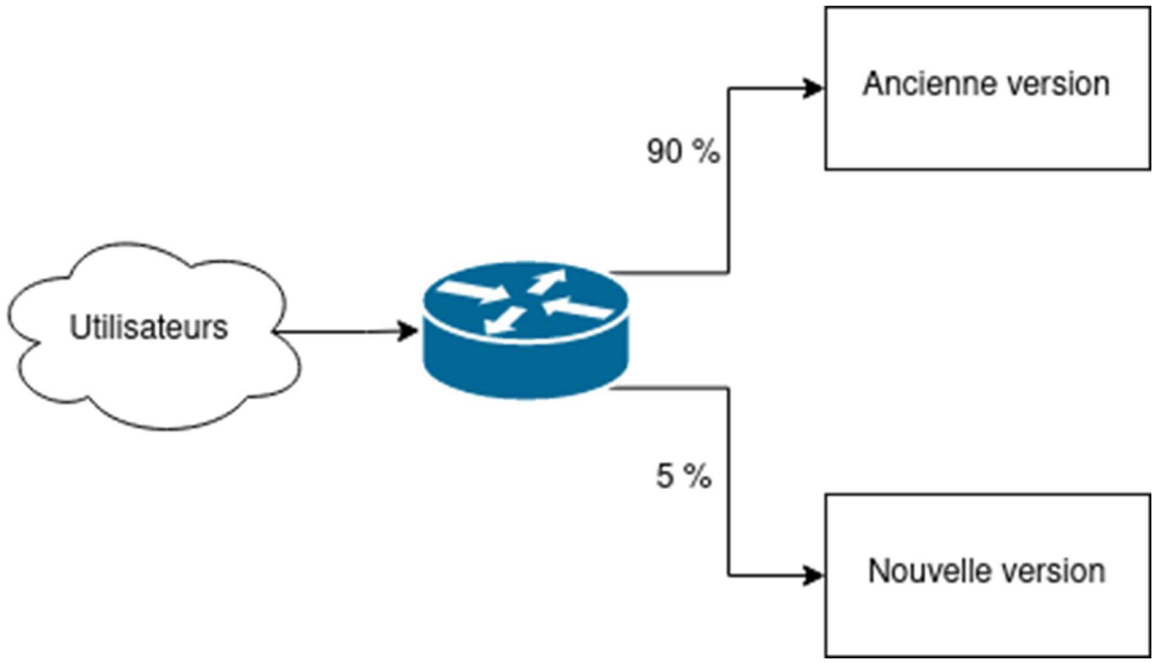


Figure 6 : Schémas d'explication du déploiement canary.

Infrastructure

Nous avons l'obligation de tester notre plateforme avant la mise en production définitive. Il nous faut donc une infrastructure capable de répondre à ce besoin. Pour cela, nous effectuons une étape entre le développement local et la mise en production que nous appellerons la mise en développement.

La mise en développement consistera à “push” le code source sur une branche nommée “development”. Lorsque le code source sera reçu par cette branche, un workflow va déployer le code source de la même manière que la mise en production, mais dans un environnement intermédiaire.

Pour revenir, sur le choix de notre infrastructure, celle-ci doit contenir aux moins deux environnements :

- L’environnement de développement n’a pas besoin d’être accessible par le grand public, il sera alors encadré par l’infrastructure “On-premise”.
- L’environnement de production est un environnement qui nécessite une attention particulière, car elle reçoit tous les utilisateurs et doit être assez robuste pour être doublé lors des déploiements. Pour cela, nous avons retenu le “private cloud” qui contrairement au “public cloud” de laisser uniquement le locataire utiliser les ressources de la machine.

L’ensemble des architectures devront pouvoir héberger plusieurs conteneurs Docker, il nous faut donc la possibilité d’installer des logiciels sur les machines. Pour l’infrastructure de l’environnement de développement, c’est assez facile, les serveurs sont hébergés au sein des locaux. Pour l’infrastructure de la production, nous il devra respecter le modèle IaaS.

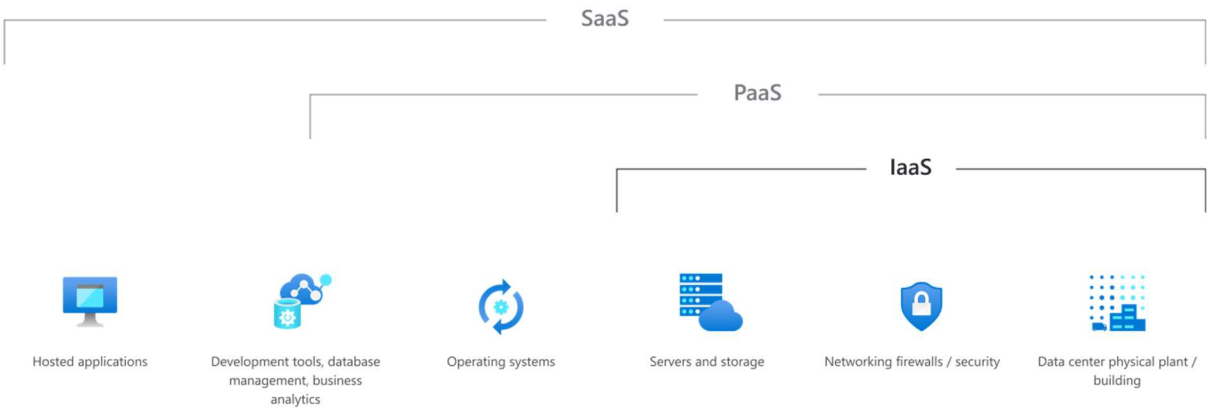


Figure 7 : Comparatif des différents modèles d’infrastructures.

Analyse du besoin

Pour réaliser cette plateforme de livraison de restauration à domicile, plusieurs versions d'interfaces sont nécessaires en fonction du rôle de l'utilisateur. Qu'il soit client, restaurateur, livreur ou encore membre de certains services internes de Dapoka comme les services commerciaux, techniques et de développement, l'utilisateur aura accès à une interface adaptée à lui et son besoin.

Le client, celui qui commande un plat chez un restaurateur pour se faire livrer à domicile, a besoin d'une version de la plateforme avec la possibilité de choisir son restaurant, d'effectuer une commande avec un ou plusieurs articles, tout en pouvant modifier, supprimer et payer cette même commande. Le client doit également pouvoir créer un compte afin de réaliser des commandes. Il doit aussi en avoir la gestion, avec la possibilité d'en éditer les informations, mais aussi de le supprimer. Enfin, l'utilisateur client doit également pouvoir suivre l'état de sa commande, éditer son historique des commandes, mais aussi de parrainer ses proches.

Le restaurateur, quant à lui, possède des besoins totalement différents. Si comme le client, il doit être en mesure de gérer son compte, en allant de la création, de l'édition, de la suppression mais aussi du système de parrainage (entre restaurateurs dans ce cas de figure cependant) son interface est différente, car de nouvelles fonctionnalités lui sont consacrées. Le restaurateur pourra créer, modifier, supprimer, éditer un article comme par exemple un plat, une boisson ou même une sauce. Il aura également la possibilité de créer, de modifier ou de supprimer un menu (sous forme de composition d'articles, comme un menu comprenant un plat, un dessert et une boisson). Le restaurateur pourra aussi avoir une gestion des commandes reçues en ayant la possibilité de les visualiser, de les valider, mais également d'en suivre le processus de livraison. Enfin, les restaurateurs seront en mesure d'éditer l'historique des commandes, mais aussi d'avoir accès à des statistiques concernant leurs ventes.

Enfin, le livreur, comme les deux précédents rôles, aura la possibilité de créer, gérer et supprimer son compte tout en parrainant d'autres livreurs. Cependant, il aura accès à une interface utilisateur totalement différente. Il aura le choix d'accepter ou de refuser de prendre en charge une livraison, mais également d'acquitter cette même livraison.

Maintenant, que nous avons listé les besoins des principaux acteurs du cycle d'utilisation de notre plateforme, nous allons nous intéresser aux besoins des développeurs tiers et des services internes : commerciaux et techniques. Les développeurs tiers sont les derniers à avoir dans leurs besoins la création, l'édition et la suppression de compte. Ils doivent pouvoir éditer les composants disponibles de la plateforme et doivent également être en mesure de les télécharger.

Les membres du service commercial de Dapoka ont quant à eux des besoins totalement différents des utilisateurs classiques que nous avons énumérés plus haut. Ils ont besoin de pouvoir éditer, suspendre, supprimer des comptes clients, mais aussi d'avoir un accès aux statistiques de la

plateforme et aux tableaux de bords de suivi de processus de commande en temps réel (avec par exemple, l'accès à la passation commande, à l'acceptation d'une commande, ou encore à l'acceptation et l'acquittement d'une livraison, et enfin le chiffre d'affaires transactionnel global en cours).

Enfin, le service technique devra être en mesure d'ajouter, supprimer des composants réutilisables, d'éditer les logs de connexions, les statistiques de performances des serveurs et micro-services, et les logs de téléchargement des composants réutilisables. Les membres du service technique de Dapoka doivent aussi pouvoir organiser les routes pour les résolutions des demandes entrantes et pouvoir déployer de nouveaux services ou micro-services sans interruption du service client.

Nous avons décidé de réaliser un diagramme bête à cornes, afin d'illustrer l'analyse fonctionnelle du besoin. Le diagramme bête à corne est un schéma qui permet de visualiser l'utilité pour l'utilisateur d'un produit, en l'occurrence la plateforme Dapoka, afin de savoir si elle répond à des besoins. Cet outil qui nous permet de comprendre correctement les besoins des utilisateurs répond à trois questions :

- À qui rend-il service ?
- Sur quoi agit-il ?
- Dans quel but ?

C'est grâce à ce diagramme que nous allons pouvoir démontrer que la conception de notre plateforme est nécessaire.

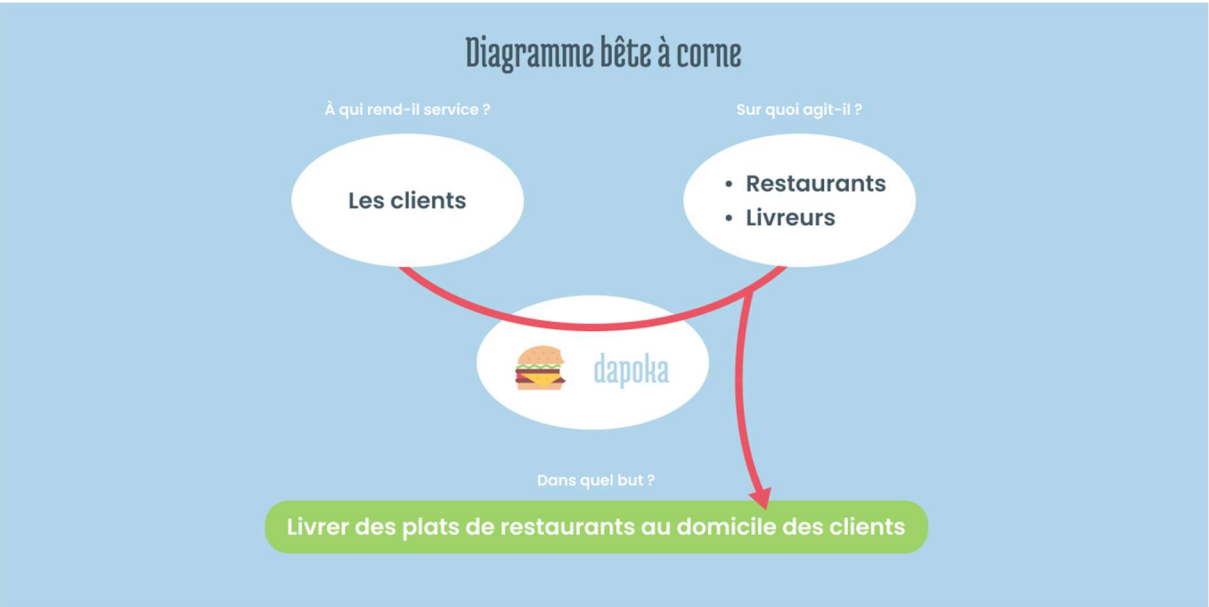


Figure 8 : Diagramme bête à corne

Analyse des risques

Lors du lancement de ce projet, nous avons décidé de réaliser une analyse des risques avec pour objectif de déterminer quels étaient les éléments critiques pour pouvoir convenir de solution limitant les potentiels dégâts produits par ces risques. Pour ce faire, nous avons réalisé un tableau regroupant les différents risques liés à notre projet, et nous avons déterminé leur criticité à partir de leur probabilité et leur gravité.

Nature du risques	Description	Gravité	Probabilité	Criticité	Conséquence si avéré
Juridique	Non-respect des délais	Grave	peu probable	critique	Délais
Technique	Impossibilité de passer une commande pour les utilsateur	Très grave	improbable	critique	Perte d'argent et de confiance
Humain	Maladies/décès de membre du projet	Grave	peu probable	critique	Délais
Juridique	Non-respect du cahier des charges	Grave	peu probable	critique	rupture du contrat
Technique	Faille de sécurité des données utilisateur	Grave	probable	critique	Perte d'argent et de confiance
Technique	Impossibilité pour les restaurateur de modifier leur offre	moyenement grave	improbable	peu critique	Perte d'argent et de confiance
Technique	failure bdd	Grave	probable	critique	Perte d'argent et de confiance
Technique	failure middleware	Grave	probable	critique	Perte d'argent et de confiance
Technique	failure des serveurs	Très Grave	probable	Trés critique	fin du monde

Figure 9 : Tableau d'analyse des risques

Questions à débattre

Dans la même logique, nous avons relevé des questions à débattre que nous allons présenter au client lors du prochain check up de l'avancement du projet. Cela permettra de s'assurer que notre vision du projet est bien la même que celle du client et d'éclaircir les points encore flous sur le projet.

Voici la liste des questions à débattre :

- Pourquoi utiliser un système micro-service et non SOA ou Monolithique ?
- Comment assurer la sécurité de la livraison côté livreur et client ?
- Le rôle de l'utilisateur (client, restaurateur ou livreur) est-elle liée au compte de l'utilisateur ou cumule-t-il les différents rôles ?
- Quels sont les attendus visuels du projet ?

Glossaire

Analyse fonctionnelle du besoin

L'analyse fonctionnelle du besoin est une analyse des fonctionnalités d'un produit selon les besoins de ses futurs utilisateurs.....13

Diagramme bête à cornes

Le diagramme bête à cornes se présente sous forme d'un graphique permettant de démontrer si un produit ou service s'adapte aux besoins des utilisateurs. Il correspond à l'image d'une bête à cornes, plus précisément à la tête d'un taureau. La corne est notamment représentée par la ligne qui relie les deux bulles qui se trouvent au-dessus.....13

Interface utilisateur

Une interface utilisateur, ou User Interface en anglais (UI), désigne l'ensemble des éléments graphiques et textuels qui permettent une interaction entre l'utilisateur et le site internet, l'application ou le logiciel. Cela inclut donc l'ensemble des items interactifs et « cliquables », tels que la barre de navigation du site, les boutons ou les liens. 12

Plan de déploiement

Le plan de déploiement est un outil efficace pour les équipes responsables de l'amélioration de la qualité, car il leur permet de définir clairement la façon dont elles envisagent de planifier le déploiement des améliorations réalisées.....10

Table des figures