

# **Application DeliveryFit Proposition commerciale**

## 1. Contexte

*Décrire ici le contexte global et le contexte lié au besoin métier.*

### **Contexte global**

Alisa’s Closet est un détaillant de vêtements français depuis plus de 10 ans, disposant d'une application en ligne permettant aux clients de commander et se faire livrer. Le système actuel centralise les expéditions depuis un seul centre de livraison, ce qui entraîne des délais et des coûts supplémentaires. L’entreprise souhaite optimiser ce processus en intégrant l’application DeliveryFit, capable de déterminer le point d’expédition optimal en fonction de la proximité et de la disponibilité des articles.

### **Fonctionnalités métier**

DeliveryFit interviendra lors de la validation du panier sur le site e-commerce existant. Elle récupérera l'adresse de livraison, la liste des articles achetés, et la liste des magasins de l’enseigne, pour déterminer le lieu d’expédition optimal (centre de livraison ou magasin) et renvoyer cette information au client avant paiement.

## 2. Réalisation du projet

### **2.1 Définition des tâches techniques**

* **Mise à jour de la base de données** : Mettre à jour la BDD pour permettre le stockage des magasins de l'enseigne, ajouter une table des magasins (id, adresse, nom du gérant, prénom du gérant, sexe du gérant, email du gérant, téléphone du gérant) et une table de jointure entre magasin et produits (id, id produit, id magasin, quantité disponible)
* **Créer l'application Frontend via Angular :** Créer la page de validation du panier avec les informations d'expédition, créer des validateurs de champs dans le but d'éviter l'envoi d'un formulaire incomplet ou dont l'adresse n'existe pas, créer un service permettant le transfert des informations de livraison au backend, créer des modèles de données pour faciliter la structure des DTOs, créer une méthode dans le service pour transmettre les informations de livraison (articles de la commande, contact, adresse de livraison) et création d'une page d'attente de la récupération d'un point d'expédition
* **Réaliser les tests unitaires de la partie frontend avec Jest :** Mise en place d’une batterie de tests en lien avec les formulaires (vérifications des envois de requêtes, des erreurs potentielles, présence de messages personnalisés et autre validation des champs)
* **Intercepter l'URL du site de vente en ligne et rediriger vers DeliveryFit :** Mettreen place un mécanisme permettant la redirection depuis le bouton « Commander »
* **Créer l'application Backend via Spring Boot** : Créer un repository permettant la récupération des établissements et de leurs stocks, créer un service permettant le traitement des établissement et de leurs stocks, créer une méthode permettant de récupérer en BDD les établissements disponibles pour une livraison dans un rayon pré-défini, comparer dans une méthode du service les distances entre les magasins et / ou le centre de livraison et le domicile, créer une méthode permettant le calcul du temps de trajet depuis chaque endroit vers le domicile de la personne et l'obtention de la route la plus rapide, créer un DTO d'un lieu d'expédition, d'une adresse de livraison, de la date potentielle d'expédition et estimation du temps de trajet / date d'arrivée, utiliser le DTO pour envoyer les informations au contrôleur, créer un contrôleur dans l'API pour, à partir d'un DTO d'informations de panier et d'une adresse client, retourner un DTO contenant la réponse appropriée
* **Faire des tests unitaires pour la partie backend** : Réaliser les tests des méthodes du service de traitement des établissements et des stocks, des retours et des potentielles levées d'exceptions, réaliser les tests du contrôleur dans le but de vérifier les codes de retour HTTP et les messages potentiels (code de statut en lien avec des problèmes d’authentification par exemple)
* **Finaliser la partie frontend suite au retour de l’API** : Créer la page de confirmation de la livraison possible (bouton confirmer la livraison / procéder au paiement), créer une page d'erreur informant qu'il n'est pas possible d'effectuer la livraison pour l'adresse demandée (message d'erreur libre), implémenter des tests de la page d'attente / confirmation de la commande potentielle, avec vérification de messages d'erreur en cas de soucis et créer la redirection de DeliveryFit vers le site de vente en ligne une fois la livraison confirmée par l’utilisateur
* **Réaliser des tests de type E2E pour valider le fonctionnement de l’application** : Vérifier que l’application DeliveryFit se lance bien au bon moment de la réalisation d’une commande par l’utilisation, que les requêtes entre l’API et l’application frontend ont bien lieu et retournent les bonnes valeurs. Vérifier également que les messages d’erreurs soient affichés à l’utilisateur en cas de soucis éventuel et enfin que la redirection en cas d'appui sur le bouton de validation de la livraison retourne vers l’application de base et au service de paiement
* **Créer un mécanisme de notification au magasin expéditeur de l'arrivée d'une nouvelle commande à préparer**: Mettre en place un potentiel micro-service dont le déclenchement serait basé sur la réception d’un message provenant d’une queue RabbitMQ ou d’un flux avec Kafka et qui aurait pour but l’envoi d’un email au gérant d’un magasin pour l’informer d’une nouvelle commande

### **2.2 Gestion des points de complexité**

Les points de complexité ont été évalués en commun, en fonction de l’expérience personnelle de l’équipe, en utilisant le principe de la suite de Fibonacci. L’ensemble de l’équipe a d’abord donné une première estimation. À la suite de cette estimation, la point a été rediscuté ensemble et les composantes ré-évaluées (difficulté actuelle de la tâche, nécessité de recherches préalables à sa réalisation, risques éventuels). Les tâches ont été ordonnées de sorte à ce quelles suivent une logique de construction de l’application.

Les tâches les plus simples se sont vues attribuer la valeur de 1 et les plus complexes sont évaluées à un score de 8. Les plus complexes sont en général dans la section backend de l’applicatif et sont en lien avec la réalisation de l’algorithme de sélection automatique de l’établissement d’expédition. Cet algorithme va demander de la recherche afin de combiner en une logique métier un équivalent en classique « shortest path » et à une vérification des stocks d’un établissement. Cela se fera sans doute sous la forme d’une itération qu’il faudra ensuite optimiser de sorte à ne pas causer de ralentissement majeur dans l’application et une attente trop longue pour le client lors de sa commande (pouvant causer une perte d’intérêt majeur).

|  |  |  |  |
| --- | --- | --- | --- |
| Tâche technique | Complexité | Temps | Marge |
| Ajouter une table des magasins en base de données | 1 | 0,1 | 0 |
| Ajouter une table de jointure pour suivre les stocks des magasins | 1 | 0,1 | 0 |
| Réaliser la redirection vers l’application DeliveryFit | 2 | 0,1 | 0 |
| Réaliser la redirection vers l’application de base une fois confirmation de la commande | 2 | 0,1 | 0 |
| Créer une page de formulaire pour l’adresse de livraison | 3 | 0.5 | 0,25 |
| Tester le comportement de la page du formulaire dans plusieurs navigateurs desktop / mobile | 5 | 0,5 | 0,25 |
| Réaliser la vérification des champs du formulaire avant son envoi potentiel | 3 | 0,1 | 0 |
| Réaliser la logique métier permettant l’envoi du formulaire à l’API backend | 3 | 0,5 | 0 |
| Créer une page d’attente de traitement par le backend | 3 | 0,1 | 0 |
| Créer la page de confirmation de livraison et de sélection du point de livraison | 3 | 0,5 | 0,25 |
| Gérer les messages d’erreurs potentiels au sein de la page de confirmation | 3 | 0,5 | 0,25 |
| Gérer l’affichage et le comportement de la page de validation de livraison dans plusieurs navigateurs | 5 | 0.5 | 0,25 |
| Créer le point d’entrée de l’API permettant le traitement des requêtes en lien avec une livraison | 3 | 0,5 | 0 |
| Réaliser une récupération de l’ensemble des points d’expédition potentiels dans une zone géographique donnée située autour d’un emplacement | 3 | *0,25* | *0,1* |  |
| Réaliser l’algorithme de sélection du magasin ou du centre de livraison le plus proche | 8 | 0.5 | 0,25 |
| Ajouter la vérification des stocks du magasin trouvé par l’algorithme afin de confirmer ou infirmer une livraison possible de cet emplacement | 5 | 0,5 | 0,25 |
| Réaliser des tests de fonctionnement de l’interface utilisateur dans l’application frontend | 3 | 0,5 | 0,5 |
| Réaliser les tests de la logique métier dans la section frontend | 5 | 0,5 | 0,25 |
| Réaliser des tests des contrôleurs et du fonctionnement de l’API dans le backend | 5 | 0,5 | 0,25 |
| Faire des tests des méthodes de sélection des établissements de livraison potentiel dans la logique de l’API | 5 | 0,5 | 0,25 |
| Réaliser des tests E2E pour valider la navigation depuis, dans et vers l’application DeliveryFit | 3 | 0,25 | 0 |
| Réaliser les tests E2E d’envois des requêtes entre le frontend et le backend | 5 | 0,25 | 0,25 |
| Vérifier la sécurité de l’application DeliveryFit (authentification requise, routes et méthode de transfert des données entre backend et frontend) | 8 | 0,5 | 0,25 |
| Réaliser la conteneurisation de l’application frontend | 5 | 0,2 | 0 |
| Faire la mise en conteneur de l’application backend | 5 | 0,2 | 0 |
| Vérifier la communication des conteneurs de l’ensemble de l’applicatif | 3 | 0,2 | 0 |

### **2.3 Coûts**

*Les coûts ont été calculés en se basant sur la séparation des compétences au sein de l’équipe. La présence d’un développeur frontend et d’un développeur backend permet dans un premier temps la séparation de l’applicatif et de son développement en deux grandes sections (l’application Angular et le microservice en Spring Boot). De la sorte, il est possible de permettre le travail en parallèle de deux individus. L’architecte se chargera se son côté des tâches plus en lien avec de la conception et du déploiement, de sorte à ne pas avoir à intervenir dans le travail des deux autres membres. Il se chargera aussi de la présentation des livrables au client, ce également dans le but de ne pas ralentir le développement et d’offrir des réponses plus adaptées aux questions possibles du client, ce via sa plus grande expérience.*

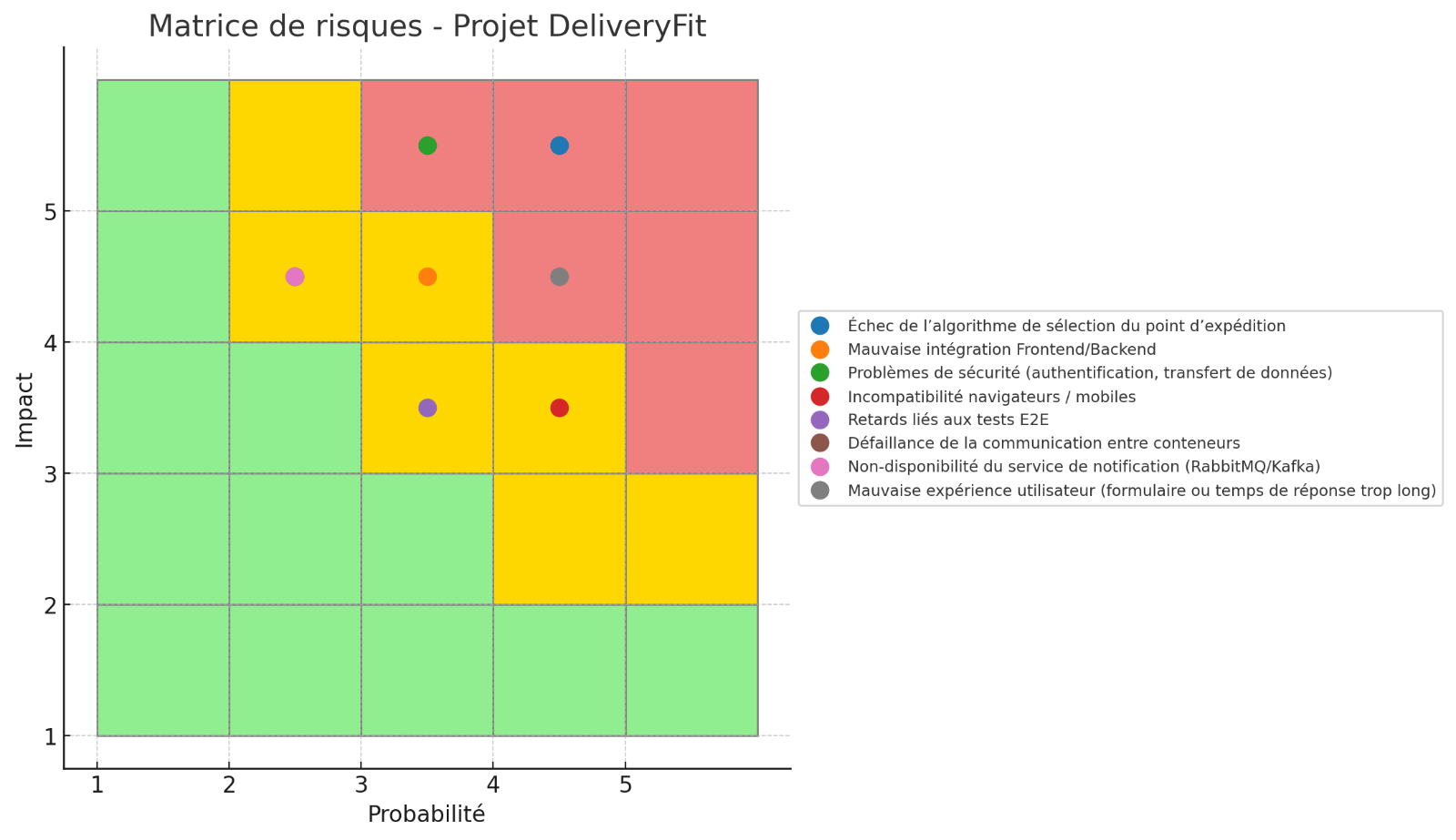
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tâche | Temps | Membre de l’équipe | TJM | Total |
| Ajouter une table des magasins en base de données | 0,1 | *Architecte* | *800* | *80* |
| Ajouter une table de jointure pour suivre les stocks des magasins | 0,1 | *Architecte* | *800* | *80* |
| Réaliser la redirection vers l’application DeliveryFit | 0,1 | *Dev Front* | *700* | *70* |
| Réaliser la redirection vers l’application de base une fois confirmation de la commande | 0,1 | *Dev Front* | *700* | *70* |
| Créer une page de formulaire pour l’adresse de livraison | 0.75 | *Dev Front* | *700* | *525* |
| Tester le comportement de la page du formulaire dans plusieurs navigateurs desktop / mobile | 0,75 | *Dev Front* | *700* | *525* |
| Réaliser la vérification des champs du formulaire avant son envoi potentiel | 0,1 | *Dev Front* | *700* | *70* |
| Réaliser la logique métier permettant l’envoi du formulaire à l’API backend | 0,5 | *Dev Front* | *700* | *350* |
| Créer une page d’attente de traitement par le backend | 0,1 | *Dev Front* | *700* | *70* |
| Créer la page de confirmation de livraison et de sélection du point de livraison | 0,75 | *Dev Front* | *700* | *525* |
| Gérer les messages d’erreurs potentiels au sein de la page de confirmation | 0,75 | *Dev Front* | *700* | *525* |
| Gérer l’affichage et le comportement de la page de validation de livraison dans plusieurs navigateurs | 0.75 | *Dev Front* | *700* | *525* |
| Créer le point d’entrée de l’API permettant le traitement des requêtes en lien avec une livraison | 0,5 | *Dev Back* | *700* | *350* |
| Réaliser une récupération de l’ensemble des points d’expédition potentiels dans une zone géographique donnée située autour d’un emplacement | *0,35* | *Dev Back* | *700* | *245* |
| Réaliser l’algorithme de sélection du magasin ou du centre de livraison le plus proche | 0.75 | *Dev Back* | *700* | *525* |
| Ajouter la vérification des stocks du magasin trouvé par l’algorithme afin de confirmer ou infirmer une livraison possible de cet emplacement | 0,75 | *Dev Back* | *700* | *525* |
| Réaliser des tests de fonctionnement de l’interface utilisateur dans l’application frontend | 1 | *Dev Front* | *700* | *700* |
| Réaliser les tests de la logique métier dans la section frontend | 0,75 | *Dev Front* | *700* | *525* |
| Réaliser des tests des contrôleurs et du fonctionnement de l’API dans le backend | 0,75 | *Dev Back* | *700* | *525* |
| Faire des tests des méthodes de sélection des établissements de livraison potentiel dans la logique de l’API | 0,75 | *Dev Back* | *700* | *525* |
| Réaliser des tests E2E pour valider la navigation depuis, dans et vers l’application DeliveryFit | 0,25 | *Architecte* | *800* | *200* |
| Réaliser les tests E2E d’envois des requêtes entre le frontend et le backend | 0,5 | *Architecte* | *800* | *400* |
| Vérifier la sécurité de l’application DeliveryFit (authentification requise, routes et méthode de transfert des données entre backend et frontend) | 0,75 | *Architecte* | *800* | *600* |
| Réaliser la conteneurisation de l’application frontend | 0,2 | *Architecte* | *800* | *160* |
| Faire la mise en conteneur de l’application backend | 0,2 | *Architecte* | *800* | *160* |
| Vérifier la communication des conteneurs de l’ensemble de l’applicatif | 0,2 | *Architecte* | *800* | *160* |
| Présentation du premier livrable au client | 0,1 | *Architecte* | *800* | *80* |
| Réunion à l’issue du premier sprint | 0,1 | *Tous* | *2200* | *220* |
| Présentation de l’applicatif final au client | 0,1 | *Architecte* | *800* | *80* |

### 

### **2.4 Risques identifiés**

L’identification des risques a été réalisée en équipe. Chaque risque a été évalué sous la forme de sa probabilité d’apparition et de son impact sur le projet global. En multipliant ces deux valeurs (dont les limites vont de 1 à 5), on obtient un niveau de risque. Ce niveau permet la quantification du risque. Plus ce niveau est grand, plus le risque est élevé. Pour chaque risque, un plan d’action a été prévu de sorte à pouvoir anticiper au maximum l’arrivée de ce problème lors du développement / livraison du produit au client.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Risque** | **Probabilité (1–5)** | **Impact (1–5)** | **Niveau (P×I)** | **Plan d’action** |
| Échec de l’algorithme de sélection du point d’expédition | 4 | 5 | 20 | Réaliser des prototypes, tests unitaires et optimisation avant mise en production |
| Mauvaise intégration Frontend/Backend | 3 | 4 | 12 | Mettre en place des tests E2E et API automatisés |
| Problèmes de sécurité (authentification, transfert de données) | 3 | 5 | 15 | Audit sécurité, mise en place de JWT, tests d’intrusion |
| Incompatibilité navigateurs / mobiles | 4 | 3 | 12 | Tests multi-navigateurs et responsive design |
| Retards liés aux tests E2E | 3 | 3 | 9 | Planification progressive des tests et automatisation |
| Défaillance de la communication entre conteneurs | 2 | 4 | 8 | Utiliser Docker Compose/Kubernetes, monitoring de la communication inter-services |
| Non-disponibilité du service de notification (RabbitMQ/Kafka) | 2 | 4 | 8 | Mettre en place un mécanisme de retry et un monitoring de la queue |
| Mauvaise expérience utilisateur (formulaire ou temps de réponse trop long) | 4 | 4 | 16 | Optimisation du backend, UX testing et feedback utilisateurs |



## 3. Synthèse

*A l’heure actuelle, le budget final est de* ***8 535€*** *si l’on réalise un algorithme ne sélectionnant pas forcément le point d’expédition optimal mais proposant l’ensemble des centres d’expédition disposant d’un stock. Le déploiement de l’applicatif n’est également pas réalisé via Docker dans ce cas de figure, mais simplement confié à une équipe en charge de ce dernier.*

*Ce budget est estimé dans le cas où le développement se voit entravé par soucis techniques ou des retours négatifs au niveau du produit, ce dernier entrainant un retour sur le travail préalable. Une marge a été estimée au niveau de chaque tâche en temps supplémentaire en cas de correctif.*

*Il est également possible d’optimiser l’application, en proposant une* ***conteneurisation*** *du produit et la réalisation d’un* ***algorithme plus avancé*** *de sélection du point d’expédition basé sur la vérification, après la distance à parcourir pour le livreur, des stocks du magasin. Cette proposition est estimée à un budget de* ***9 335€****, avec**encore une fois des marges en cas de retour négatifs à la suite du premier livrable ou de soucis techniques.*