# Application DeliveryFit - Proposition commerciale

## 1. Contexte

### 1.1 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.

### 1.2 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

- Cadrage fonctionnel et user stories

- Architecture technique et modélisation BDD

- Mise en place environnement dev/test

- Développement API (récupération données, algorithme de choix, vérification stock, gestion erreurs)

- Développement Front (intégration validation panier, écrans recherche et confirmation, redirection)

- Tests unitaires, intégration et affichage

- Recette interne et corrections

- Livraison finale et documentation

### 2.2 Gestion des points de complexité

Les points de complexité ont été attribués en fonction de la technicité, du nombre de dépendances et du volume de travail estimé.

|  |  |  |  |
| --- | --- | --- | --- |
| Tâche technique | Complexité | Estimation temps (j) | Marge |
| Cadrage fonctionnel et user stories | Moyenne | 1.0 | 0.1 |
| Architecture technique et modélisation BDD | Élevée | 1.5 | 0.2 |
| Environnement dev/test | Faible | 0.25 | 0.05 |
| Développement API Back End | Élevée | 3.0 | 0.5 |
| Développement Front End | Moyenne | 3.0 | 0.5 |
| Tests & intégration | Moyenne | 1.5 | 0.2 |
| Recette interne & corrections | Moyenne | 0.5 | 0.1 |
| Livraison & documentation | Faible | 0.5 | 0.05 |

### 2.3 Coûts

Le chiffrage a été réalisé sur la base des TJM fournis (Architecte : 800 €, Dev Front : 700 €, Dev Back : 700 €), avec optimisation pour respecter le budget maximal de 10 000 €.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tâche | Temps (j) | Membre de l’équipe | TJM (€) | Total (€) |
| Cadrage fonctionnel | 0.75 | Architecte | 800 | 600 |
| User stories | 0.25 | Architecte | 800 | 200 |
| Architecture technique & BDD | 0.75 | Architecte | 800 | 600 |
| Environnement dev/test | 0.25 | Architecte | 800 | 200 |
| Suivi des sprints & démo | 0.75 | Architecte | 800 | 600 |
| Livraison & documentation | 0.5 | Architecte | 800 | 400 |
| API Back End | 3.0 | Dev Back | 700 | 2100 |
| Front End | 3.0 | Dev Front | 700 | 2100 |
| Tests & intégration | 1.25 | Dev Back | 700 | 875 |
| Recette interne | 0.5 | Equipe partagée | 0 | 370 |

Total estimé : 9 170 € (marge restante : 830 € sur budget max 10 000 €).

### 2.4 Risques identifiés

- Risque technique : compatibilité avec le site existant et performance de l’algorithme.  
- Risque organisationnel : retard en cas de validations tardives du client.  
- Risque qualité : exactitude des données de stock.

### 2.5 Objectifs de performance

- Temps de calcul du point d’expédition : < 2 secondes.  
- Disponibilité des données : > 99 %.  
- UX fluide : moins de 0,5 seconde pour chargement écrans Front.

## 3. Synthèse

La solution proposée respecte le budget alloué (9 170 € sur 10 000 €) et se base sur une équipe de 3 personnes (Architecte, Dev Back, Dev Front) sur environ 12,5 jours-homme. Le projet sera mené en méthodologie Scrum sur 2 sprints, avec une livraison intermédiaire et une livraison finale. En option, une version améliorée pourrait inclure un module d’optimisation avancée des tournées de livraison, pour un budget estimé à 12 000 €.