

CAHIER DES CHARGES OFFICIEL

Fonctionnalité : Demande BTP avec Devis PDF et Délai (Modèle type inDrive)

1. Présentation générale

La plateforme est une application de mise en relation entre clients et artisans du BTP. Elle agit comme intermédiaire neutre, sans intervenir dans la négociation, l'exécution ou la facturation des travaux. Cette fonctionnalité permet aux clients de publier des demandes libres et aux artisans d'y répondre par devis PDF dans un délai limité.

2. Objectifs

- Faciliter la publication de besoins BTP
- Professionnaliser les réponses via devis PDF
- Introduire un mécanisme de temps limité
- Forcer l'amélioration des demandes sans réponse

3. Périmètre

Fonctionnalités incluses : création de demandes, gestion des délais, réception de devis PDF, sélection finale.

Fonctionnalités exclues : paiement en ligne, contrat, suivi de chantier, intervention financière.

4. Acteurs

- Client : publie les demandes et choisit un artisan
- Artisan : répond par devis PDF
- Système : gère délais et règles
- Administrateur : supervision

5. Fonctionnalités côté Client

Le client crée une demande avec un titre et une description détaillée. La description doit inclure la localisation, la surface, le budget estimatif et les contraintes. Il consulte les devis reçus, télécharge les PDF, et sélectionne un artisan.

En cas d'absence de devis à l'expiration du délai, la demande passe au statut expiré et le client doit obligatoirement modifier la description avant de relancer.

6. Fonctionnalités côté Artisan

L'artisan consulte les demandes ouvertes et peut répondre à une demande en envoyant un devis au format PDF. Un seul devis est autorisé par demande et il ne peut pas être modifié après envoi.

7. Règles métier

- La plateforme ne fixe aucun prix
- Le devis PDF fait foi
- Une demande a un délai obligatoire
- Une demande assignée est verrouillée

8. Cycle de vie d'une demande

Créée → Ouverte → Expirée (si aucun devis) → Modification obligatoire → Ouverte → Assignée

9. Notifications

Clients : nouveau devis, délai proche, demande expirée, devis accepté.

Artisans : nouvelle demande, devis accepté ou refusé.

10. Sécurité et conformité

Vérification du format PDF, limitation de taille, stockage sécurisé, traçabilité des actions et respect de la confidentialité.

11. Contraintes techniques

Backend Laravel, base de données relationnelle, stockage local ou cloud, scheduler pour la gestion des expirations.

12. Conclusion

Cette fonctionnalité renforce la crédibilité de la plateforme, simplifie l'expérience client, professionnalise les artisans et respecte le rôle d'intermédiaire neutre.

Workflow de travail – Standard Projet

Ce document décrit **ma logique de travail standard** pour tout nouveau projet, quel que soit le langage ou le framework utilisé. Cette méthode est obligatoire et sert de référence.

Philosophie générale

Un projet réussi commence toujours par une réflexion claire, structurée et documentée.

Aucun code n'est écrit sans : - une compréhension complète du besoin - un cahier des charges - une définition claire du MVP

Cahier des charges (OBLIGATOIRE)

Chaque projet commence par un **gros cahier des charges**.

Contenu minimum :

- Objectif du projet
- Problème à résoudre
- Utilisateurs cibles
- Contraintes (techniques, budget, délai)
- Fonctionnalités globales

 **Aucun développement sans validation du cahier des charges.**

Définition du MVP (Minimum Viable Product)

Avant toute implémentation, le projet est découpé en :

- Fonctionnalités **MVP** (essentiels)
- Fonctionnalités secondaires
- Fonctionnalités futures (roadmap)

 Le MVP doit : - apporter une vraie valeur - être livrable rapidement - être testable

Architecture & logique technique

Avant le code, une **architecture claire** est définie.

Logique standard (exemple API / Backend) :

1. Migrations de base de données
2. Modèles (ORM / Eloquent / Entities)
3. Services métier (Business Logic)
4. Controllers (API / Web)
5. Form Requests (Validation)
6. API Resources (Transformation des données)
7. Routes

📌 Chaque couche a **une seule responsabilité**.



Règles de développement

- Séparation stricte des responsabilités
 - Controllers fins (pas de logique métier)
 - Logique métier uniquement dans les Services
 - Validation centralisée
 - Réponses API normalisées
 - Code lisible, testable et maintenable
 - 💡 **En cas d'ambiguïté, poser des questions avant toute action**
 - **Toujours produire un plan d'implémentation avant de coder**
-



Plan d'implémentation (OBLIGATOIRE)

Avant toute nouvelle fonctionnalité :

- Analyser le besoin
- Identifier les ambiguïtés
- Poser des questions de clarification
- Générer un **plan d'implémentation détaillé**
- Analyser et valider ce plan avant toute écriture de code

📌 **Aucune fonctionnalité n'est implémentée sans plan validé.**



Ordre de développement

Le développement suit toujours cet ordre :

1. Base de données (migrations)
2. Modèles et relations
3. Services métier
4. Controllers
5. Validation
6. Transformation des réponses
7. Routes

📌 **Jamais l'inverse.**

🤔 **Réutilisation & amélioration des fonctionnalités**

- La majorité des fonctionnalités implémentées sont :
 - inspirées de fonctionnalités existantes dans d'autres applications
 - réimplémentées de manière propre
 - améliorées (performance, UX, sécurité, maintenabilité)
 - Toujours analyser l'existant avant de réinventer
 - Documenter les améliorations apportées
-

🧠 **Évolution du projet**

- Chaque nouvelle fonctionnalité suit le même cycle
 - Le MVP est protégé (pas de régression)
 - Refactoring uniquement si justifié
-

🧐 **Validation & qualité**

Avant toute livraison : - Tests manuels ou automatisés - Vérification des règles métier - Vérification des cas limites - Relecture du code

🧑 **Règle d'or**

Pas de code sans vision.
Pas de vision sans cahier des charges.
Pas de raccourci sur la qualité.

🔔 **Utilisation**

Ce fichier doit être : - conservé à la racine du projet - utilisé comme référence pour tout nouveau projet - partagé avec les collaborateurs