<Swap> **Cahier des charges**</swap>

**Table des matières**

[Contexte et Présentation 4](#_Toc189433547)

[Objectif 4](#_Toc189433548)

[Enjeux 4](#_Toc189433549)

[Périmètre et Public Cible 5](#_Toc189433550)

[Besoins Fonctionnels 6](#_Toc189433551)

[Gestion des Utilisateurs 6](#_Toc189433552)

[Création de compte 6](#_Toc189433553)

[Authentification et Autorisations 6](#_Toc189433554)

[Profil 6](#_Toc189433555)

[Gestion des Tâches (Tasks) 7](#_Toc189433556)

[Création d’une tâche 7](#_Toc189433557)

[Consultation des tâches disponibles 7](#_Toc189433558)

[Ajout de Tags 7](#_Toc189433559)

[Suivi de statut 7](#_Toc189433560)

[Gestion des Propositions (TaskProposal) 7](#_Toc189433561)

[Postuler à une tâche 7](#_Toc189433562)

[Validation ou Refus 7](#_Toc189433563)

[Gestion des Compétences (Skills) 8](#_Toc189433564)

[Définition des Skills 8](#_Toc189433565)

[Gestion des Avis (Reviews) 8](#_Toc189433566)

[Notation 8](#_Toc189433567)

[Communication (Conversations, Messages) 8](#_Toc189433568)

[Conversations privées 8](#_Toc189433569)

[Interface d’Administration 8](#_Toc189433570)

[Gestion des données 8](#_Toc189433571)

[Autres Fonctions 9](#_Toc189433572)

[Recherche / Filtrage 9](#_Toc189433573)

[Notifications (optionnel) 9](#_Toc189433574)

[Besoins Non Fonctionnels 10](#_Toc189433575)

[Sécurité 10](#_Toc189433576)

[Performance et Scalabilité : 10](#_Toc189433577)

[Ergonomie 10](#_Toc189433578)

[Maintenance 10](#_Toc189433579)

[Respect de la vie privée 10](#_Toc189433580)

[Architecture Technique 11](#_Toc189433581)

[Choix des Technologies 11](#_Toc189433582)

[Diagramme UML 11](#_Toc189433583)

[Modèle de Données 12](#_Toc189433584)

[User (classe abstraite) 12](#_Toc189433585)

[Customer 12](#_Toc189433586)

[Agent 12](#_Toc189433587)

[Task 12](#_Toc189433588)

[TaskProposal 13](#_Toc189433589)

[Skill 13](#_Toc189433590)

[Tag 13](#_Toc189433591)

[Review 13](#_Toc189433592)

[Conversation 13](#_Toc189433593)

[Message 14](#_Toc189433594)

[Exigences de Qualité et Tests 15](#_Toc189433595)

[Tests Unitaires 15](#_Toc189433596)

[Tests Fonctionnels 15](#_Toc189433597)

[Analyse statique 15](#_Toc189433598)

[CI/CD 15](#_Toc189433599)

[Planning et Contraintes 16](#_Toc189433600)

[Étapes principales 16](#_Toc189433601)

[Contraintes 16](#_Toc189433602)

[Conclusion 17](#_Toc189433603)

# Contexte et Présentation

## Objectif

Nom du projet : Swap   
Permettre à des utilisateurs (Clients) de déléguer ou externaliser des tâches qu’ils ne souhaitent ou ne peuvent pas réaliser eux‐mêmes (récupérer un colis, annoncer une mauvaise nouvelle, etc.). Des prestataires (Agents) peuvent alors postuler pour effectuer ces tâches à la place des Clients, moyennant rémunération.

## Enjeux

Simplifier la vie de personnes qui n’ont pas le temps, ou qui ont des difficultés à accomplir certaines démarches.

Offrir un nouveau type de service qui “sous‐traite” même des actions très personnelles.

Centraliser la mise en relation entre Clients et Agents.

# Périmètre et Public Cible

**Clients :** toute personne souhaitant déléguer une tâche (ex. un particulier pressé, une personne qui souhaite éviter un déplacement, etc.).

**Agents :** des travailleurs indépendants ou toute personne souhaitant gagner de l’argent en rendant service (aller chercher des colis, discuter à la place d’un client, etc.).

**Administrateurs (ou équipe interne) :** gèrent la plateforme, modèrent le contenu, veillent à la sécurité et aux règles de fonctionnement.

# Besoins Fonctionnels

## Gestion des Utilisateurs

### Création de compte

Permettre à un utilisateur de s’inscrire en tant que Client ou Agent (ou possibilité de basculer d’un rôle à l’autre).

Stocker email, mot de passe, rôles, informations personnelles.

### Authentification et Autorisations

S’authentifier avec email + mot de passe.

Gérer différents rôles : ex. ROLE\_CUSTOMER, ROLE\_AGENT, ROLE\_ADMIN, ROLE\_BANNED

Gérer les restrictions (accès aux fonctionnalités selon le rôle).

### Profil

Mettre à jour ses informations (adresse, pseudo, téléphone, etc.).

Voir son tableau de bord (tâches publiées ou propositions envoyées).

## Gestion des Tâches (Tasks)

### Création d’une tâche

Un Client définit la description, le titre, l’éventuel lieu, la rémunération indicative.

La tâche est publiée avec un statut initial (ex. “PENDING”).

### Consultation des tâches disponibles

Un Agent peut parcourir les tâches publiées (filtrer par mot‐clé, tag, localisation, etc.).

### Ajout de Tags

Un Client peut associer des tags (catégories) à une tâche (ex. “Livraison”, “Urgent”).

Permet de classer et rechercher plus facilement les tâches.

### Suivi de statut

Le Client peut suivre l’évolution de sa tâche (en attente, en cours, terminée…).

## Gestion des Propositions (TaskProposal)

### Postuler à une tâche

Un Agent soumet une proposition avec un prix (ou des conditions).

Un statut de proposition (ex. “PENDING”, “ACCEPTED”, “REFUSED”) est géré.

### Validation ou Refus

Le Client peut accepter ou refuser la proposition d’un Agent.

Lorsqu’une proposition est acceptée, la tâche peut passer à “IN\_PROGRESS” (ou équivalent).

## Gestion des Compétences (Skills)

### Définition des Skills

Chaque Agent peut lister ses compétences (ex. “Force physique”, “Convaincre”, “Conduite”…).

Permet de filtrer les Agents selon leurs aptitudes.

## Gestion des Avis (Reviews)

### Notation

Permettre au Client de laisser un avis sur l’Agent (note + commentaire) après réalisation d’une tâche.

Permettre à l’Agent d’être évalué et de construire une réputation (rating global).

## Communication (Conversations, Messages)

### Conversations privées

Un Client et un Agent peuvent échanger des messages pour clarifier les détails d’une tâche, négocier.

Historique de chat (conversation), daté.

## Interface d’Administration

### Gestion des données

L’Administrateur peut accéder à un back‐office pour visualiser les utilisateurs, les tâches, les avis.

Possibilité de supprimer un contenu inapproprié (à savoir très peu) ou bloquer un utilisateur.

## Autres Fonctions

### Recherche / Filtrage

Rechercher des tâches selon différents critères (tag, localisation, date, etc.).

Rechercher des Agents par compétence, note moyenne.

### Notifications (optionnel)

Envoi d’emails ou de notifications lors de la création d’une proposition, validation d’une tâche, etc.

# Besoins Non Fonctionnels

## Sécurité

Système d’authentification robuste (hashage des mots de passe).

Gestion fine des autorisations (un Client ne peut modifier que ses tâches, un Agent ne peut voir que ses propres propositions, etc.).

## Performance et Scalabilité :

L’application doit supporter un certain volume d’utilisateurs et de tâches.

Possibilité de faire évoluer l’infrastructure.

## Ergonomie

Interface utilisateur claire, accessible sur mobile et desktop.

Processus de publication de tâche simple et rapide.

## Maintenance

Code organisé, tests unitaires et fonctionnels.

Gestion du versionnement (Git).

## Respect de la vie privée

Collecte de données personnelles limitée (uniquement ce qui est nécessaire).

Conformité RGPD (dans l’UE).

# Architecture Technique

## Choix des Technologies

Framework : Symfony (PHP).

Base de données : PostgreSQL.

ORM : Doctrine pour la gestion des entités et relations.

Front‐end : Twig

## Diagramme UML

**Une image contenant texte, diagramme, Plan, schématique

Description générée automatiquement**

## Modèle de Données

### User (classe abstraite)

Représente la **base** commune à tous les utilisateurs (champs : email, mot de passe, rôles…).

On distingue ensuite deux types d’utilisateurs réels : **Customer** (client) et **Agent** (prestataire), qui héritent de **User**.

Dans une base de données, cela peut être géré via une colonne de discrimination (single table inheritance) ou via deux tables séparées (joined table inheritance), selon votre choix.

### Customer

Hérite de **User** : ce sont les **clients** qui ont besoin de déléguer une tâche.

Peut créer plusieurs Task pour décrire ce qu’il veut faire réaliser (ex. “Aller chercher un colis”, “Annoncer une mauvaise nouvelle à ma place”, etc.).

Peut laisser des **reviews** (avis) sur les agents qui ont travaillé pour lui.

### Agent

Hérite de **User** : ce sont les **prestataires** disponibles pour effectuer les tâches à la place du client.

Peut avoir plusieurs **skills** (compétences).

Peut envoyer des **TaskProposal** (des propositions) pour “appliquer” sur une tâche et proposer un prix.

Peut recevoir des **Review** (avis) laissés par les clients.

### Task

Représente une **tâche** ou un **service** que le client veut déléguer.

Chaque Task appartient à un et un seul Customer.

Peut recevoir plusieurs **TaskProposal** d’agents différents.

Peut être associée à plusieurs **Tag** (catégories, mots-clés).

### TaskProposal

Correspond à une **offre** ou une **proposition** faite par un agent pour une tâche précise.

Permet de définir un prix proposé, un statut (en attente, accepté, refusé, etc.).

Lié à un unique Task et un unique Agent.

### Skill

Représente une **compétence** (ex. “Combat”, “Discrétion”, “Transport”, “Éloquence”, etc.) qu’un agent peut posséder.

Il y a une relation “many-to-many” entre Agent et Skill, car un agent peut avoir plusieurs compétences et une compétence peut être partagée par plusieurs agents.

### Tag

Représente un **mot-clé** ou une **catégorie** pour mieux **classifier** les tâches (ex. “Urgent”, “Risque élevé”, “Physique”, “Livraison”, etc.).

Il y a une relation “many-to-many” entre Task et Tag, car une même tâche peut avoir plusieurs étiquettes et un tag peut être associé à plusieurs tâches.

### Review

C’est un **avis** déposé par un **Customer** sur un **Agent** (ou éventuellement l’inverse si vous le permettez).

Contient généralement une note (rating), un commentaire (comment), une date de création (createdAt).

Permet d’évaluer la qualité du travail effectué ou l’expérience du client.

### Conversation

Permet de représenter un **canal de discussion** entre un **Customer** et un **Agent** (ou plusieurs participants, selon l’implémentation).

Possède une date de démarrage.

Peut contenir plusieurs **Message**.

Sert à échanger des informations sur la tâche, négocier les détails, etc.

### Message

Correspond à un **message** écrit dans une Conversation.

Possède un contenu (content), une date d’envoi (sentAt).

Est relié à la Conversation dans laquelle il est posté.

Est relié à un User pour savoir quel message à été écrit par qui.

# Exigences de Qualité et Tests

## Tests Unitaires

Vérifier la logique métier (un Agent ne peut pas postuler deux fois à la même tâche, etc.).

## Tests Fonctionnels

Scénarios de bout en bout (création de compte, publication de tâche, proposition, acceptation, etc.).

## Analyse statique

Utilisation de Symfony Insights pour vérifier la qualité du code.

## CI/CD

Pipeline d’intégration continue pour lancer les tests, le linting et le déploiement automatique.

# **Planning et Contraintes**

## Étapes principales

Conception de la base de données et des entités.

Développement des fonctionnalités principales.

Intégration du front‐end et tests.

Hebergement

Veille post-prod

## Contraintes

Respecter la sécurité des échanges (HTTPS obligatoire).

Respecter les deadlines.

# Conclusion

Le présent cahier des charges définit les fondations du projet Swap. Il décrit les principales fonctionnalités, la structure de données, les exigences techniques, ainsi que les contraintes de sécurité et de qualité.