Cahier des Charges Fonctionnel et Technique du Projet ServicesBLADI

GitHub Copilot (basé sur l’analyse du projet) 13 mai 2025

**Version :** 1.0

**Contents**

1. [Introduction](#_bookmark0) 3
   1. [Objet du Document](#_bookmark1) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
   2. [Contexte du Projet](#_bookmark2) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
   3. [Public Cible](#_bookmark3) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
   4. [Définitions et Acronymes](#_bookmark4) . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
2. [Description Générale du Projet](#_bookmark5) 3
   1. [Nom du Projet](#_bookmark6) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
   2. [Mission et Vision](#_bookmark7) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
   3. [Objectifs Principaux](#_bookmark8) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
3. [Acteurs et Rôles](#_bookmark9) 4
   1. [Client (client)](#_bookmark10) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
   2. [Expert (expert)](#_bookmark11) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5
   3. [Administrateur (admin)](#_bookmark12) . . . . . . . . . . . . . . . . . . . . . . . . . . . 5
4. [Fonctionnalités Détaillées](#_bookmark13) 6
   1. [Module Transverse : Gestion des Comptes et Authentification (Application](#_bookmark14)

[accounts)](#_bookmark14) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6

* 1. [Module : Catalogue et Gestion des Services (Application services)](#_bookmark15) . . . 6
  2. [Module : Gestion des Demandes et Interactions (Application requests)](#_bookmark16) . 7
  3. [Module : Gestion des Ressources et Informations (Application resources)](#_bookmark17) 8
  4. [Module : Interface d’Administration Sécurisée](#_bookmark18) . . . . . . . . . . . . . . . 8
  5. [Fonctionnalités Transverses Additionnelles](#_bookmark19) . . . . . . . . . . . . . . . . . 9

1. [Architecture Technique](#_bookmark20) 10
   1. [Architecture Globale](#_bookmark21) 10
   2. [Technologies Backend](#_bookmark22) 10
   3. [Technologies Frontend](#_bookmark23) 10
   4. [Base de Données](#_bookmark24) 10
   5. [Structure du Projet Django](#_bookmark25) 11
   6. [Environnement de Développement et de Production (préconisations)](#_bookmark26) 11
2. [Spécifications Non Fonctionnelles](#_bookmark27) 11
   1. [Performance](#_bookmark28) 11
   2. [Sécurité](#_bookmark29) 12
   3. [Fiabilité et Disponibilité](#_bookmark30) 12
   4. [Maintenabilité et Évolutivité](#_bookmark31) 12
   5. [Ergonomie et Expérience Utilisateur (UX)](#_bookmark32) 12
   6. [Internationalisation (i18n)](#_bookmark33) 12
3. [Perspectives d’Évolution](#_bookmark34) 13

# Introduction

## Objet du Document

Ce document constitue le cahier des charges fonctionnel et technique du projet **Services- BLADI**. Il a pour objectif de décrire de manière exhaustive les fonctionnalités attendues, les spécifications techniques, les contraintes et les objectifs de la plateforme. Il servira de référence tout au long du cycle de vie du projet.

## Contexte du Projet

**ServicesBLADI** est une application web conçue pour servir de guichet unique numérique, facilitant la mise en relation entre des utilisateurs (principalement des Marocains Rési- dant à l’Étranger - MRE, mais aussi d’autres clients) et des experts proposant une gamme variée de services. Le projet vise à simplifier l’accès à ces services, à optimiser la commu- nication et à centraliser la gestion des demandes et des documents.

## Public Cible

Ce document s’adresse à :

* + - L’équipe de développement et de maintenance.
    - Le chef de projet et les parties prenantes.
    - Toute personne impliquée dans l’évaluation ou l’évolution de la plateforme.

## Définitions et Acronymes

* + - **MRE** : Marocain Résidant à l’Étranger
    - **CRUD** : Create, Read, Update, Delete
    - **UI** : User Interface (Interface Utilisateur)
    - **UX** : User Experience (Expérience Utilisateur)
    - **API** : Application Programming Interface
    - **SGBD** : Système de Gestion de Base de Données
    - **i18n** : Internationalisation

# Description Générale du Projet

## Nom du Projet

ServicesBLADI

## Mission et Vision

* + - **Mission** : Fournir une plateforme centralisée, sécurisée, et conviviale permettant aux utilisateurs d’accéder facilement à des services spécialisés (administratifs, fis-

caux, touristiques, immobiliers, investissement) et aux experts de gérer efficacement les demandes clients.

* + - **Vision** : Devenir la plateforme de référence pour les MRE et autres citoyens cher- chant des services fiables et accessibles liés au Maroc, en favorisant la transparence et l’efficacité.

## Objectifs Principaux

* + - Offrir un catalogue de services clair, structuré et facilement navigable.
    - Simplifier et digitaliser le processus de demande de service et de prise de rendez- vous.
    - Assurer une communication fluide, traçable et sécurisée entre les clients et les ex- perts.
    - Permettre une gestion documentaire centralisée, sécurisée et efficace.
    - Fournir une interface d’administration robuste pour la gestion complète de la plate- forme (utilisateurs, services, demandes, contenus).
    - Garantir la sécurité des données personnelles et des transactions.
    - Supporter l’internationalisation pour une accessibilité accrue (Français, Anglais, Arabe).

# Acteurs et Rôles

La plateforme **ServicesBLADI** identifie trois rôles principaux, définis dans le modèle

Utilisateur de l’application accounts via le champ account\_type.

## Client (client) Fonctions :

* + - Créer et gérer son compte personnel (profil, informations, préférences linguistiques).
    - Consulter le catalogue des services par catégorie.
    - Soumettre des demandes de service détaillées.
    - Suivre l’état d’avancement de ses demandes.
    - Télécharger, gérer et partager des documents relatifs à ses demandes.
    - Prendre, consulter et gérer des rendez-vous avec les experts.
    - Communiquer avec les experts assignés via une messagerie intégrée.
    - Recevoir des notifications (internes à la plateforme, potentiellement email).
    - Consulter les ressources d’information (FAQ, guides, informations consulaires).
    - Réinitialiser son mot de passe.

## Expert (expert) Fonctions :

* + - Créer et gérer son profil professionnel (spécialités, compétences, langues parlées, biographie, tarif horaire, disponibilités).
    - Recevoir et consulter les demandes de service qui lui sont assignées.
    - Accepter, rejeter ou demander des informations complémentaires pour une de- mande.
    - Mettre à jour le statut des demandes en cours de traitement.
    - Gérer son calendrier de rendez-vous avec les clients.
    - Communiquer avec les clients via la messagerie intégrée.
    - Accéder aux documents partagés par les clients pour les demandes concernées.
    - Potentiellement contribuer à la base de ressources (selon droits accordés par l’admin).
    - Recevoir des notifications.

## Administrateur (admin)

**Fonctions** (détaillées dans ADMIN\_ACCESS\_GUIDE.md et ADMIN\_INTERFACE\_GUIDE.md) :

* + - Accéder à une interface d’administration sécurisée et dédiée.
    - Gérer l’ensemble des comptes utilisateurs (CRUD, activation/désactivation, modi- fication du type de compte, réinitialisation de mot de passe).
    - Gérer le catalogue des services (catégories, types de service, services spécifiques - CRUD).
    - Superviser et gérer toutes les demandes de service (assignation aux experts, suivi global, intervention si nécessaire).
    - Gérer les documents de la plateforme (vérification, validation, rejet, suppression).
    - Superviser et gérer les rendez-vous.
    - Gérer la base de ressources (FAQ, articles, documents téléchargeables - CRUD, gestion de la visibilité).
    - Modérer et superviser les communications (messages) si nécessaire.
    - Consulter un tableau de bord avec des statistiques et indicateurs de performance de la plateforme.
    - Effectuer des actions en masse sur les utilisateurs et les documents (cf. admin\_bulk\_actions.py).
    - Configurer certains paramètres de la plateforme.
    - Créer d’autres comptes administrateurs (via script create\_admin.py ou interface).

# Fonctionnalités Détaillées

## Module Transverse : Gestion des Comptes et Authentification (Appli- cation accounts)

**Modèles de données :** Utilisateur (modèle customisé héritant d’AbstractUser), Client, Expert, Admin (profils spécifiques), Address, Notification (notifications générales de compte).

## Fonctionnalités :

* + - Inscription des utilisateurs avec sélection du type de compte (client par défaut, expert soumis à validation potentielle).
    - Connexion sécurisée basée sur l’email et mot de passe (utilisation de accounts.backends.EmailBa
    - Déconnexion.
    - Processus de réinitialisation de mot de passe par email (utilisation des vues Django

auth\_views).

* + - Gestion des profils utilisateurs (modification des informations personnelles, photo de profil, etc.).
    - Vérification d’email (fonctionnalité à envisager pour is\_verified).
    - Gestion des adresses multiples par utilisateur.
    - Système de notifications internes pour les actions liées au compte.

## Module : Catalogue et Gestion des Services (Application services)

**Modèles de données :** ServiceCategory, ServiceType, Service (modèle de base), et modèles spécifiques héritant de Service (TourismService, AdministrativeService, InvestmentService, RealEstateService, FiscalService).

## Fonctionnalités :

* + - **Pour l’Administrateur :**
      * Gestion CRUD des catégories de services (nom, description, slug, icône, tra- ductions fr/ar).
      * Gestion CRUD des types de services au sein des catégories (nom, description, prix de base, traductions fr/ar).
      * Gestion CRUD des services spécifiques (titre, description détaillée, prix final, durée, assignation d’expert par défaut si applicable, statut, champs spécifiques par type de service).
      * Script create\_services.py pour initialisation.

## Pour le Client/Visiteur :

* + - * Consultation du catalogue des services par catégorie et type.
      * Affichage détaillé de chaque service (description, prix, prérequis, etc.).
      * Navigation intuitive (ex: pages dédiées Tourisme.html, Administrative.html, etc., gérées par services.views et frontend\_urls.py).
  1. **Module : Gestion des Demandes et Interactions (Application requests) Modèles de données :** ServiceRequest, RendezVous, Document, Message, Notification

(spécifique aux requêtes).

## Fonctionnalités :

* + - **Gestion des Demandes de Service (ServiceRequest) :**
      * Client : Création d’une demande liée à un service, description du besoin, ajout de pièces jointes initiales, spécification de la priorité et date souhaitée.
      * Client/Expert : Suivi du statut de la demande (Nouveau, En attente d’infos, En cours, Complété, Annulé, Rejeté).
      * Expert : Prise en charge, mise à jour du statut, communication avec le client.
      * Admin : Supervision, assignation, intervention.

## Gestion des Rendez-vous (RendezVous) :

* + - * Client/Expert : Proposition, acceptation, annulation, reprogrammation de RDV.
      * Attributs : date/heure, durée, type (en personne, vidéo, téléphone), statut, notes.

## Gestion des Documents (Document) :

* + - * Utilisateurs : Téléchargement de documents (divers formats) liés à une de- mande ou un RDV.
      * Attributs : type de document, nom, fichier, date de téléversement, statut (en attente de vérification, vérifié, rejeté).
      * Admin : Vérification, validation, rejet des documents. Vue delete\_document\_view.py pour suppression.
      * Filtres de template personnalisés pour l’affichage des documents (ex: document\_filters.py)

## Messagerie Interne (Message) :

* + - * Communication écrite et asynchrone entre client et expert (et potentiellement admin).
      * Lien optionnel à une ServiceRequest.
      * Notification de nouveaux messages.

## Notifications Spécifiques (requests.models.Notification) :

* + - * Alertes pour les mises à jour de demandes, nouveaux messages, confirmations de RDV, actions sur les documents.

## Tableaux de Bord Utilisateurs (dashboard\_views.py, dashboard\_views\_new.py)

**:**

* + - * Client : Vue d’ensemble de ses demandes, RDV, documents, messages.
      * Expert : Vue d’ensemble des demandes assignées, RDV, documents clients, messages.

## Module : Gestion des Ressources et Informations (Application resources)

**Modèles de données :** Resource, ResourceFile, ResourceLink, ConsulateEmbassy, FAQ.

## Fonctionnalités :

* + - **Pour l’Administrateur :**
      * Gestion CRUD des ressources (articles, guides, formulaires téléchargeables, liens externes).
      * Organisation par catégories (administratif, fiscal, etc.).
      * Gestion des langues et formats disponibles pour chaque ressource.
      * Gestion CRUD des informations sur les consulats/ambassades.
      * Gestion CRUD de la Foire Aux Questions (FAQ) par catégorie.

## Pour le Client/Expert (client\_views.py, expert\_views.py dans resources)

**:**

* + - * Consultation et recherche dans la base de ressources.
      * Téléchargement de fichiers.
      * Accès à la FAQ.
      * Consultation des informations consulaires.

## Module : Interface d’Administration Sécurisée

**Accès :** Via URL dédiée (ex: /admin/dashboard/ ou /management/secure8765/login/), protégé par authentification spécifique aux comptes admin. (cf. ADMIN\_ACCESS\_GUIDE.md). **Templates dédiés :** Structure admin/base.html et pages dérivées (cf. admin).

**Vues dédiées :** admin\_views.py dans l’application requests (et potentiellement dans d’autres apps pour la gestion spécifique).

**Fonctionnalités Clés** (cf. ADMIN\_INTERFACE\_GUIDE.md et frontend\_urls.py pour les routes /admin/.../) :

* + - Tableau de bord administrateur : Statistiques clés, aperçu de l’activité.
    - Gestion complète des utilisateurs (liste, filtres, recherche, ajout, modification, sup- pression, activation/désactivation).
    - Gestion des demandes de service (liste, filtres, assignation, suivi détaillé).
    - Gestion des documents (liste, filtres, vérification, rejet, suppression).
    - Gestion des rendez-vous (liste, filtres, complétion, annulation).
    - Gestion des ressources (liste, filtres, ajout, modification, suppression, gestion de la visibilité).
    - Gestion des messages (supervision, marquage comme lu).
    - Actions en masse (ex: admin\_bulk\_toggle\_users\_status, admin\_bulk\_delete\_users, admin\_bulk\_verify\_documents, admin\_bulk\_reject\_documents).
    - Utilisation d’AJAX pour certaines opérations pour une meilleure UX (via Django REST Framework si configuré pour ces vues).

## Fonctionnalités Transverses Additionnelles

* + - **Internationalisation (i18n) :**
      * Support des langues : Français (défaut), Anglais, Arabe (configuré dans settings.PY:LANGU et LOCALE\_PATHS).
      * Utilisation de gettext\_lazy dans les modèles et vues.
      * Sélecteur de langue pour l’utilisateur.
      * Context processor language\_context pour la gestion de la langue dans les templates.

## Interface Utilisateur (UI) et Expérience Utilisateur (UX) :

* + - * Utilisation de templates Django (template).
      * Styling avec Bootstrap 5 (via crispy-forms et crispy\_bootstrap5 pour les formulaires Django, et inclusion manuelle pour le reste).
      * Conception responsive pour l’adaptabilité sur différents appareils.
      * Navigation claire et intuitive.

## Sécurité :

* + - * Protection CSRF (Cross-Site Request Forgery) activée par Django.
      * Validation des mots de passe (AUTH\_PASSWORD\_VALIDATORS).
      * Permissions basées sur account\_type et décorateurs (@login\_required, véri- fications spécifiques dans les vues).
      * Échappement des données en sortie pour prévenir les attaques XSS (géré par Django templating).
      * Protection contre les injections SQL (gérée par l’ORM Django).
      * Configuration ALLOWED\_HOSTS.
      * Middleware de sécurité Django (SecurityMiddleware, XFrameOptionsMiddleware).

## API (Interne) :

* + - * Utilisation potentielle de Django REST Framework pour des appels AJAX depuis le frontend (notamment pour l’interface admin) pour des actions dy- namiques sans rechargement de page.

## Gestion des messages utilisateurs :

* + - * Utilisation du framework de messages Django pour les retours d’action (succès, erreur, information).
      * Middleware personnalisé servicesbladi.middleware.MessageMiddleware.

# Architecture Technique

## Architecture Globale

Architecture N-Tiers classique pour une application web :

* + - **Client (Navigateur Web)** : Interface utilisateur rendue en HTML, CSS, JavaScript.
    - **Serveur Web/Application (Django)** : Logique métier, gestion des requêtes, interaction avec la base de données.
    - **Serveur de Base de Données (MySQL)** : Stockage et gestion des données persistantes.

## Technologies Backend

* + - **Langage** : Python (version 3.x, ex: 3.13 comme indiqué par les pycache ).
    - **Framework** : Django (version 5.2.1 comme indiqué dans settings.py).

## Bibliothèques Python notables :

* + - * djangorestframework : Pour la création d’API (utilisé pour des fonctionnal- ités admin AJAX).
      * django-crispy-forms et crispy-bootstrap5 : Pour un rendu amélioré des formulaires Django avec Bootstrap 5.
      * Pilote MySQL (ex: mysqlclient).

## Technologies Frontend

* + - **Langages** : HTML5, CSS3, JavaScript.
    - **Framework CSS** : Bootstrap 5 (pour le design responsive et les composants UI).
    - **JavaScript** : Vanilla JS ou jQuery (selon vendor) pour l’interactivité côté client.

## Base de Données

* + - **SGBD** : MySQL (configuré dans settings.PY:DATABASES).
    - **Nom de la base** : servicesbladi.
    - **ORM** : Django ORM pour l’interaction avec la base de données.
    - **Migrations** : Système de migration Django pour gérer l’évolution du schéma de la base.

## Structure du Projet Django

* + - **Projet principal** : servicesbladi (contient settings.py, urls.py principal).

## Applications modulaires :

* + - * accounts : Gestion des utilisateurs et authentification.
      * services : Gestion du catalogue des services.
      * requests : Gestion des demandes, RDV, documents, messages.
      * resources : Gestion des ressources informationnelles.
    - **Templates** : Dans template, structurés par application ou par section (client, expert, admin).
    - **Fichiers Statiques** : Dans static (CSS, JS, images, vendor).
    - **Fichiers Médias** : Dans media (documents uploadés par les utilisateurs).

## Environnement de Développement et de Production (préconisations)

* + - **Développement** : Serveur de développement Django (manage.py runserver), SQLite pour des tests rapides (bien que MySQL soit configuré), DEBUG = True.

## Production :

* + - * Serveur WSGI (ex: Gunicorn, uWSGI).
      * Serveur Web (ex: Nginx, Apache) pour servir les fichiers statiques et comme reverse proxy.
      * Base de données MySQL en production.
      * DEBUG = False.
      * Gestion des variables d’environnement pour les configurations sensibles (SE- CRET\_KEY, DB credentials).
      * HTTPS pour sécuriser les communications.

# Spécifications Non Fonctionnelles

## Performance

* + - Temps de réponse rapides pour les interactions utilisateur.
    - Optimisation des requêtes base de données (utilisation de select\_related, prefetch\_related).
    - Pagination pour les listes longues (notamment dans l’interface admin).
    - Mise en cache (caching de templates, de requêtes) à envisager pour les sections fréquemment accédées.
    - Optimisation des images et des fichiers statiques.

## Sécurité

* + - Protection contre les vulnérabilités web courantes (OWASP Top 10) : XSS, CSRF, SQL Injection (principalement gérées par Django).
    - Mots de passe hashés (géré par Django).
    - Permissions et contrôles d’accès stricts basés sur les rôles.
    - Validation des données en entrée (côté client et serveur).
    - Utilisation de HTTPS en production.
    - Mises à jour régulières des dépendances.

## Fiabilité et Disponibilité

* + - Minimisation des points de défaillance.
    - Gestion robuste des erreurs et exceptions avec retours clairs à l’utilisateur.
    - Sauvegardes régulières de la base de données en production.

## Maintenabilité et Évolutivité

* + - Code source clair, commenté et bien structuré (respect des conventions PEP 8).
    - Conception modulaire (applications Django distinctes).
    - Utilisation de migrations Django pour faciliter les mises à jour de la base de données.
    - Documentation technique à jour (ce cahier des charges, guides admin).
    - Facilité d’ajout de nouvelles fonctionnalités ou de nouveaux types de services.

## Ergonomie et Expérience Utilisateur (UX)

* + - Interface intuitive et facile à prendre en main pour tous les types d’acteurs.
    - Parcours utilisateur logiques et fluides.
    - Design responsive assurant une bonne expérience sur ordinateurs, tablettes et mo- biles.
    - Messages d’erreur et de succès clairs et informatifs.
    - Temps de chargement des pages optimisés.

## Internationalisation (i18n)

* + - La plateforme doit être facilement traduisible dans d’autres langues.
    - Les contenus statiques et dynamiques doivent supporter la traduction.
    - Le choix de la langue par l’utilisateur doit être mémorisé.

# Perspectives d’Évolution

* **Système de paiement en ligne** : Intégration d’une passerelle de paiement pour les services payants.
* **Notifications avancées** : Notifications par email et/ou SMS en plus des notifica- tions internes.
* **Système de notation et d’évaluation** : Permettre aux clients de noter les experts et les services.
* **Tableaux de bord avancés** : Plus de graphiques et d’analyses pour les experts et administrateurs.
* **Calendrier interactif avancé** : Intégration de calendriers externes (Google Cal- endar, Outlook) pour les experts.
* **Chat en temps réel** : Remplacer la messagerie asynchrone par un chat instantané.
* **API Publique** : Pour permettre des intégrations avec des systèmes tiers.
* **Application Mobile** : Développement d’applications natives (iOS/Android) ou PWA.
* **Module de e-learning/formation** : Si pertinent pour certains services ou pour les experts.
* **Gamification** : Pour encourager l’utilisation de la plateforme.

Ce cahier des charges vise à fournir une vision complète du projet **ServicesBLADI**. Il sera mis à jour au fur et à mesure de l’évolution du projet.