RAPPORT DE STAGE

Titre Professionnel Développeur Web et Web Mobile

Présenté par

Adenan KHACHNANE

Encadrant

Yacine FEKHAR

SOMMAIRE :

1 Remerciements

Je tiens à exprimer ma profonde gratitude envers les formateurs Yacine FEKHAR et Ryad CHAOUAR pour leur accompagnement et leurs précieux conseils tout au long de cette formation. Leur expertise et leur pédagogie ont été d'une valeur inestimable, enrichissant grandement mon expérience professionnelle. Leur soutien a été une source d'inspiration et a contribué de manière significative à mon développement personnel et professionnel. Je suis reconnaissant d'avoir eu l'opportunité de bénéficier de leur savoir-faire et de leur guidance, et je suis certain que les enseignements reçus seront bénéfiques pour mes projets futurs.

Mes remerciements s'étendent également à l'ensemble du personnel et des formateurs du centre de formation de l'AFPA, dont l'engagement et le professionnalisme ont facilité mon apprentissage et mon intégration dans le monde professionnel.

Cette période de stage a été enrichissante à bien des égards, et je suis reconnaissant pour le soutien et l'opportunité qui m'ont été offerts.

2 Introduction

2.1 Introspection :

Depuis mon plus jeune âge, la technologie a toujours captivé mon intérêt. Ma curiosité pour ce domaine m'a naturellement orienté vers une licence en Conception et Fabrication Assistée par Ordinateur (CFAO), où j'ai eu mon premier contact significatif avec la programmation informatique. Utilisant Visual Basic for Applications (VBA) pour concevoir des éléments 3D sur Catia V5, j'ai découvert une passion pour donner vie à des idées au travers de lignes de code.

Avant cette expérience, j'avais déjà exploré le monde de la programmation numérique, programmant des machines à contrôle numérique (programmation point par point). Cette immersion précoce dans le monde de la création et de la technologie a jeté les bases de mon intérêt croissant pour le développement informatique. Cette immersion précoce dans le monde de la création et de la technologie a jeté les bases de mon intérêt croissant pour le développement informatique.

Suite à mon expérience enrichissante dans un bureau des méthodes, un désir de changement s'est manifesté, me poussant à entreprendre une introspection profonde pour aligner ma carrière sur mes aspirations personnelles, ma créativité et mes compétences interpersonnelles. Cette réflexion m'a conduit à saisir une opportunité de formation proposée par l'AFPA, marquant un tournant décisif dans mon parcours professionnel.

Au cours de cette formation, j'ai acquis un éventail de compétences en développement informatique, incluant HTML, CSS, JavaScript, PHP, ainsi que l'utilisation du framework Symfony avec Twig, SQL, et l'architecture MVC et N-tier. L'apprentissage de Visual Studio Code, la méthodologie Scrum, et la conception de bases de données ont également enrichi mon arsenal de développeur.

Cette formation m'a finalement mené à un stage final où, dans une équipe de 4 personnes, nous avons conçu et développé une application destinée à un client spécialisé dans la réparation d'appareils électroniques. Ce projet a non seulement été l'occasion d'appliquer concrètement mes compétences nouvellement acquises mais a aussi renforcé ma conviction que la voie du développement informatique était celle que je souhaitais poursuivre.

Ce stage a été une étape cruciale dans ma transition vers une carrière qui non seulement fait écho à mes passions mais me permet également de mettre en pratique mes compétences techniques et créatives au sein d'une équipe dynamique. C'est une période qui a marqué une évolution significative tant sur le plan professionnel que personnel, m'ouvrant les portes d'un univers où innovation et technologie se rencontrent pour créer des solutions impactantes.

2.2 Outils utilisés

Whimsical :

Figma :

VSCode :

Mamp :

Trello :

GitHub :

Voici les principaux outils utilisés pendant mon stage : Whimsical pour les wireframes, Figma pour les maquettes, Visual Studio Code pour la programmation, MAMP comme environnement de développement local, Trello pour la gestion de projet et le suivi des tâches, GitHub pour la centralisation du projet.

2.3 Présentation de l'entreprise et du contexte du stage

L'entreprise d'accueil pour mon stage, DevFix, se distingue dans le secteur de la réparation électronique par sa spécialisation dans une vaste gamme de produits technologiques, tels que les smartphones, tablettes, ordinateurs, smartwatches, et consoles de jeux. Reconnue pour son expertise technique approfondie et son engagement envers la satisfaction client, DevFix s'efforce de fournir des services de réparation rapides, fiables et de haute qualité.

Le contexte de mon stage chez DevFix était axé sur un besoin spécifique de l'entreprise : le développement d'une application sur mesure destinée à optimiser la gestion de la création de devis pour leurs services de réparation. Cette application était envisagée pour répondre précisément aux besoins uniques de l'entreprise, en simplifiant la génération de devis et en améliorant à la fois l'organisation interne et l'interaction avec la clientèle.

Ma contribution au stage consistait à participer au développement de cette solution applicative, en utilisant les compétences développées durant ma formation. Opérant en étroite collaboration avec l'équipe de DevFix et mes collègues, et dans un contexte de travail entièrement en télétravail, ce projet m'a offert une opportunité unique de saisir les défis techniques et opérationnels rencontrés par l'entreprise. Il a également été l'occasion de contribuer activement à l'élaboration de solutions innovantes visant à répondre efficacement à ces défis.

2.4 Objectifs du stage

Mon immersion chez DevFix avait pour but d'intégrer une équipe dédiée à la conception d'une solution numérique innovante, ciblant l'efficacité des processus de devis. Le stage visait non seulement à enrichir mon expérience professionnelle par l'application pratique de mes acquis académiques mais aussi à me confronter aux réalités du travail collaboratif à distance, en affinant mes compétences en développement une application au sein d'un environnement dynamique.

2.5 Présentation du projet

Le cœur de ma mission de stage résidait dans le développement d'une plateforme digitale pour DevFix, destinée à transformer leur manière de générer des devis et leurs processus de réparation. Ce projet s'inscrivait dans une démarche d'amélioration continue, visant à automatiser les tâches récurrentes et à optimiser la gestion clientèle grâce à des fonctionnalités innovantes.

2.6 Contexte et besoin

Face à une demande croissante et à la variété des interventions techniques, DevFix recherchait une solution pour harmoniser ses pratiques et accélérer leurs processus de prise en charge. L'objectif était de minimiser les erreurs et les délais, impactant directement la satisfaction de la clientèle et l'organisation interne.

2.6.1 Pourquoi cette application est nécessaire

L'application se révélait indispensable dans la stratégie de DevFix pour moderniser ses opérations. Elle devait permettre une gestion plus fluide des demandes de réparation, en centralisant les informations et en simplifiant les interactions, afin de soutenir la croissance de l'entreprise et d'améliorer son service client.

2.7 Objectifs de l'application

L'ambition de l'application englobait la facilitation de la création de devis, l'amélioration du suivi des réparations et la centralisation des données clients. Elle devait également fournir une base de données exhaustive pour une identification rapide des besoins de réparation, alignant DevFix sur les attentes actuelles du marché.

3 Analyse et conception du système

3.1 Cahier des charges

L'élaboration de la plateforme DevFix repose sur un cahier des charges détaillé, définissant clairement les attentes et les besoins fonctionnels du site web de réparation téléphonique. Ce document stratégique sert de pilier pour le développement du projet, visant à créer une interface conviviale permettant aux clients de générer des devis de manière rapide et intuitive pour leurs appareils électroniques endommagés.

Les objectifs primordiaux du site incluent la facilitation de la génération de devis à travers un processus de sélection simplifié d'appareils, marques, modèles, et types de pannes. Il est impératif que chaque utilisateur puisse créer et accéder à son profil personnel, y stocker ses informations de base, et consulter l'historique de ses devis, garantissant ainsi une expérience utilisateur enrichissante et personnalisée.

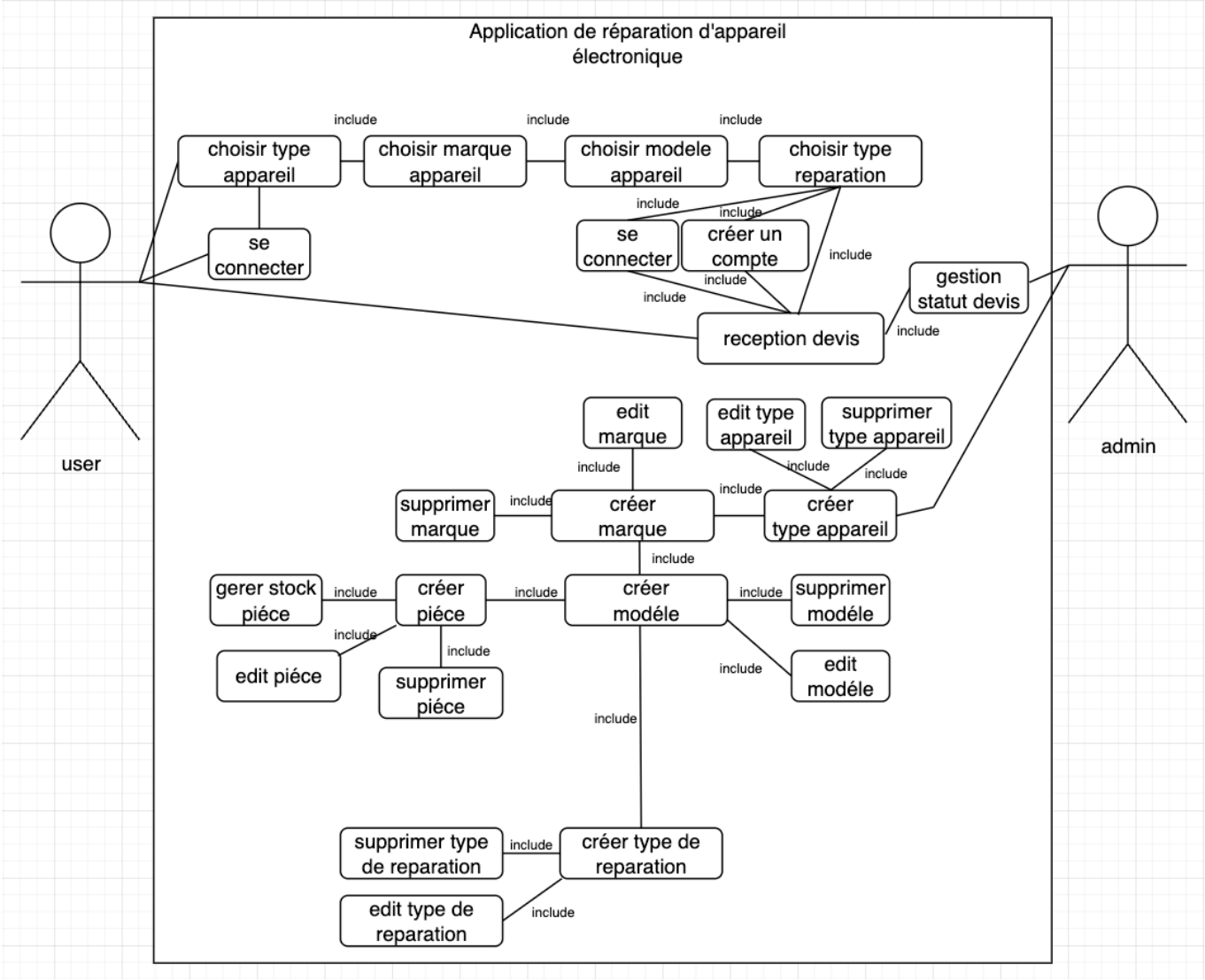
Le panel administrateur constitue un autre aspect crucial du cahier des charges, offrant une gestion complète des devis, l'historique des interactions clients, ainsi que la capacité de gérer le stock de pièces détachées de manière efficace. Ce système doit permettre une vue d'ensemble précise des opérations, l'ajout de nouvelles références produit, et l'alerte en cas de faiblesse du stock, assurant la continuité des services proposés par DevFix.

Sur le plan technique, le projet s'appuie sur des technologies éprouvées telles que PHP, HTML, JavaScript, et CSS, avec l'intégration du framework Symfony et du gestionnaire de base de données MySQL via Doctrine. Cette infrastructure est choisie pour sa robustesse, sa flexibilité, et sa compatibilité avec les standards actuels de développement web.

En matière de design et d'ergonomie, une attention particulière est portée à l'interface utilisateur pour qu'elle soit à la fois intuitive et alignée avec l'identité visuelle de DevFix.

3.2 Diagramme de cas d'utilisation

Le diagramme de cas d'utilisation mis à jour fournit une cartographie précise des interactions entre les utilisateurs, l'application de réparation d'appareils électroniques et les administrateurs. Il représente visuellement les fonctionnalités essentielles offertes par l'application, en alignement avec les spécifications détaillées dans le cahier des charges.



Pour l'Utilisateur (User) :

- **Choisir le type d'appareil :** L'interface permet aux utilisateurs de sélectionner la catégorie de l'appareil à réparer, étape initiale pour orienter le devis.
- **Choisir la marque de l'appareil :** Après avoir spécifié le type, la liste des marques correspondantes est mise à disposition pour affiner la demande.

- **Choisir le modèle de l'appareil :** Ce choix permet de préciser le devis en se basant sur les spécificités du modèle sélectionné**.**
- **Choisir le type de réparation :** Les utilisateurs déterminent la nature de la réparation, ce qui influence le coût et les spécifications du devis.
- **Se connecter ou créer un compte :** Fondamental de créer un compte ou se connecter, cet élément lie toutes les transactions au profil de l'utilisateur.
- **Réception de devis :** Une fois le processus complété, le devis est généré et peut être consulté par l'utilisateur.

Pour l'Administrateur (Admin) :

- **Gestion du statut des devis :** Permet à l'administrateur de suivre et de mettre à jour le statut des devis des clients.
- **Créer/Éditer/Supprimer marque, modèle, type d'appareil** : Gère les listes des marques, modèles et types d'appareils disponibles pour la réparation.
- **Gérer le stock de pièces :** Inclut la surveillance et la mise à jour de l'inventaire des pièces détachées, ainsi que la possibilité de créer, éditer et supprimer des pièces.
- C**réer/Éditer/Supprimer type de réparation :** Administre les différents types de services de réparation que l'entreprise offre.

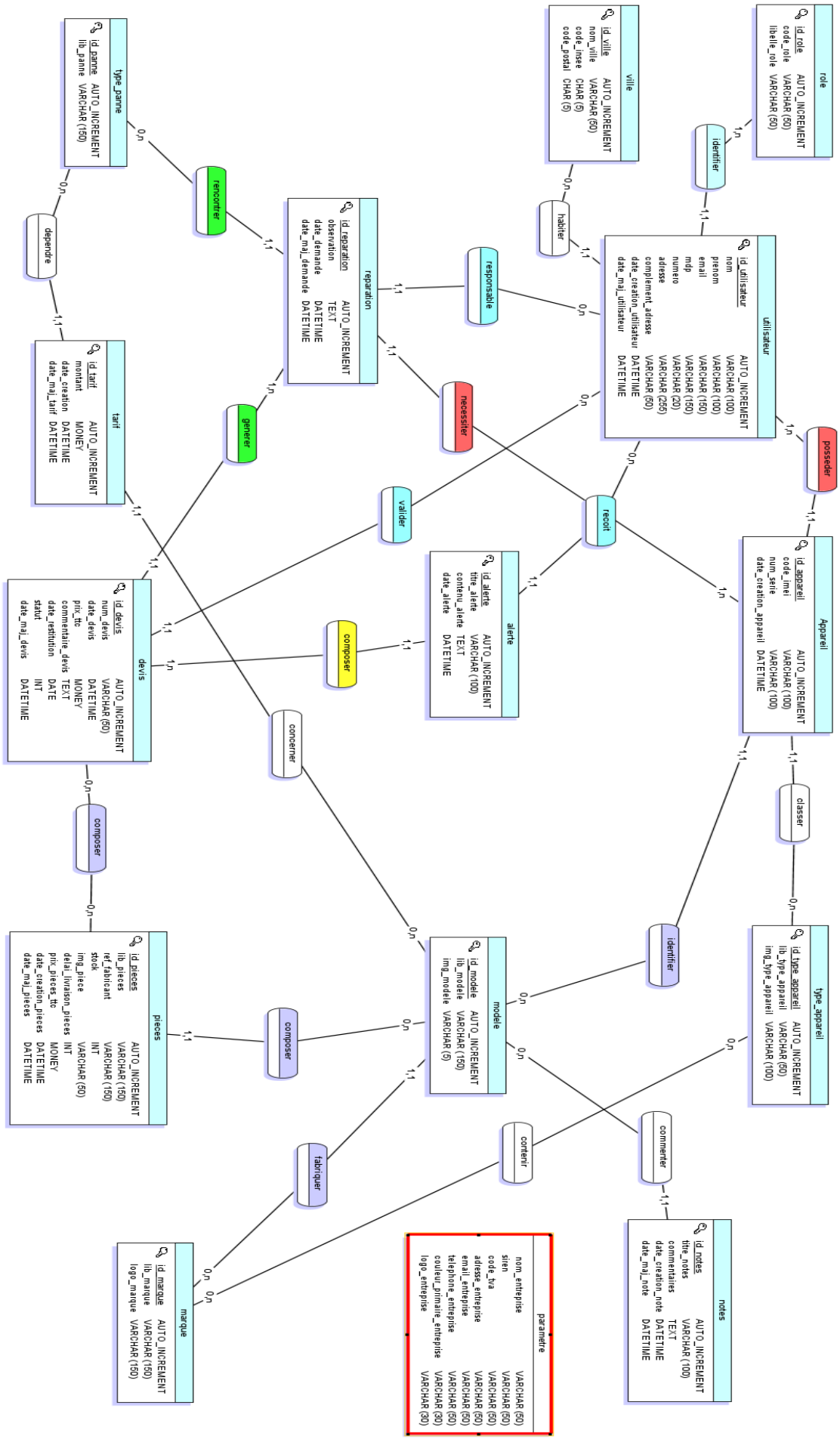
Ces fonctionnalités sont conçues pour assurer que l'application réponde efficacement aux besoins des utilisateurs et de l'administration de DevFix. Le diagramme sert de référence, garantissant que toutes les interactions sont prises en compte et que l'expérience utilisateur est fluide et intuitive. Cela souligne également l'importance de la flexibilité et de la mise à jour constante des données pour répondre aux attentes changeantes des clients et aux exigences du marché.

3.3 Dictionnaire des données

Le dictionnaire des données est un inventaire essentiel qui précise les attributs nécessaires au développement de notre Modèle Conceptuel de Données (MCD). Cette étape initiale nous a permis d'établir les fondations conceptuelles de notre système d'information avant de passer à la phase de Modèle Logique de Données (MLD), où ces concepts seront transformés en structures de tables pour la base de données.

Nom	Type	Commentaire	Exemple
code_imei	Texte	Identifiant unique international pour chaque appareil mobile	123456789012345
num_serie	Texte	Numéro de série du produit pour suivi et identification	ABCD1234567890
nom	Texte	Nom de famille du client pour l'identification et le contact	Dupont
prenom	Texte	Prénom du client pour l'identification personnelle	Jean
email	Texte	Adresse e-mail du client pour la communication	jean.dupont@example.com
mdp	Texte	Mot de passe crypté pour la sécurisation du compte client	**********
lib_panne	Texte	Description de la panne pour le diagnostic et la réparation	Ecran cassé
lib_marque	Texte	Nom de la marque de l'appareil pour le catalogue de réparation	Samsung
logo_marque	Texte	Chemin d'accès au logo de la marque pour l'affichage	/chemin/vers/logo.png
lib_modele	Texte	Nom du modèle de l'appareil pour le suivi des réparations	Galaxy S20
lib_pieces	Texte	Désignation des pièces détachées pour la gestion de stock	Batterie
ref_fabricant	Texte	Référence du fabricant pour l'identification des pièces	REF12345
stock	Entier	Quantité de pièces en stock pour la gestion d'inventaire	100
date_demande	Date	Date et heure de la demande de réparation	2024-02-20 10:00:00
date_devis	Date	Date et heure de la création du devis	2024-02-20 10:30:00
prix_ttc	Décimal	Prix total toutes taxes comprises de la réparation	250.00.00
prix_pieces_ttc	Décimal	Prix total TTC des pièces utilisées dans la réparation	150.00.00
lib_statut	Texte	Libellé du statut de la réparation pour le suivi	En attente de pièces
num_devis	Texte	Numéro du devis pour le suivi et l'historique	DEV-2024-001
observation	Texte	Commentaires ou remarques supplémentaires	Client pressé, réparation urgente
date_restitution	Date	Date prévue pour la restitution de l'appareil réparé	2024-02-25
code_roles	Texte	Code des rôles utilisateurs pour la gestion des permissions	ROLE_ADMIN
nom_admin	Texte	Nom de l'administrateur pour l'identification interne	admin
email_admin	Texte	Adresse e-mail de l'administrateur pour le contact professionnel	admin@example.com
mdp_admin	Texte	Mot de passe de l'administrateur pour l'accès sécurisé	**********
code_admin	Texte	Code unique d'identification de l'administrateur	ADM001
is_valider	Entier	Indicateur de validation d'une action ou d'une transaction	1
code_role	Texte	Code identifiant le rôle d'un utilisateur dans le système	ROLE_USER
libelle_role	Texte	Libellé du rôle pour la compréhension des permissions	Utilisateur
numero	Texte	Numéro de téléphone du client pour le contact	1234567890123
date_creation	Date	Date et heure de création d'un enregistrement dans la base	2024-02-20 09:00:00
date_maj	Date	Date et heure de la dernière mise à jour d'un enregistrement	2024-02-20 11:00:00
label	Texte	Étiquette pour identifier divers éléments comme les alertes	URGENT
date	Date	Date et heure générique pour le suivi des événements	2024-02-20 12:00:00
lib_version	Texte	Libellé de la version d'un élément ou d'une fonctionnalité	v2.0
adresse_facturation	Texte	Adresse de facturation du client pour les transactions	123 Rue Principale, 75001 Paris
img_piece	Texte	Chemin d'accès aux images des pièces détachées	/chemin/vers/image.jpg
complement_adresse	Texte	Informations supplémentaires pour l'adresse de facturation	Bâtiment B, 2ème étage
nom_ville	Texte	Nom de la ville pour l'adresse de l'utilisateur	Paris
nom_entreprise	Texte	Nom commercial de l'entreprise pour l'identification	ABC Services
siren	Texte	Numéro SIREN de l'entreprise pour des fins légales et administratives	123456789
code_tva	Texte	Numéro de TVA de l'entreprise pour la facturation	FR12345678901
type_entreprise	Texte	Catégorisation de l'entreprise selon son statut juridique	SARL
adresse_entreprise	Texte	Adresse postale de l'entreprise	456 Avenue des Entreprises, 75002 Paris
img_modele	Texte	Lien vers l'image représentative du modèle d'appareil	/chemin/vers/image_modele.jpg
email_entreprise	Texte	Adresse e-mail de contact de l'entreprise	contact@abcservices.com
telephone_entreprise	Texte	Numéro de téléphone pour joindre l'entreprise	33123456789
couleur_primaire_entrepri	Texte	Code couleur primaire utilisé dans le branding de l'entreprise	#FF0000
logo_entreprise	Texte	Chemin d'accès au logo de l'entreprise	/chemin/vers/logo_entreprise.png
delai_livraison_pieces	Entier	Durée estimée pour la livraison des pièces détachées	7

3.4 Modèle Conceptuel de Données (MCD)



Dans le cadre de la conception d'une base de données, la phase de modélisation est essentielle pour définir la structure et les relations entre les différentes entités. Le Modèle Conceptuel de Données (MCD) représente cette structure de manière abstraite, en mettant en évidence les entités, leurs attributs et les relations qui les lient. Le MCD présenté est la version 13 (v13), ayant été versionné et modifié progressivement pour plusieurs raisons.

- Suite aux réunions avec le client, nous avons identifié la nécessité d'ajouter des entités supplémentaires, telles que l'entité **appareil**, qui reflète la version du modèle du type d'appareil.

- Une deuxième raison de modification était liée au constat que nous n'avions pas pleinement respecté la troisième forme normale (**3FN**).

- Enfin, lors du développement, nous avons remarqué certaines incohérences, telles que l'absence de relation entre les entités **type_appareil** et **marque**.

Ces observations ont conduit à des ajustements successifs du MCD afin de garantir sa cohérence et sa conformité aux besoins du projet.

Le MCD est composé de 15 entités principales, à savoir :

- rôle
- utilisateur
- ville
- appareil
- alerte
- type_appareil
- notes
- paramètre
- marque
- modèle
- pièces
- devis
- réparation
- tarif
- type_panne

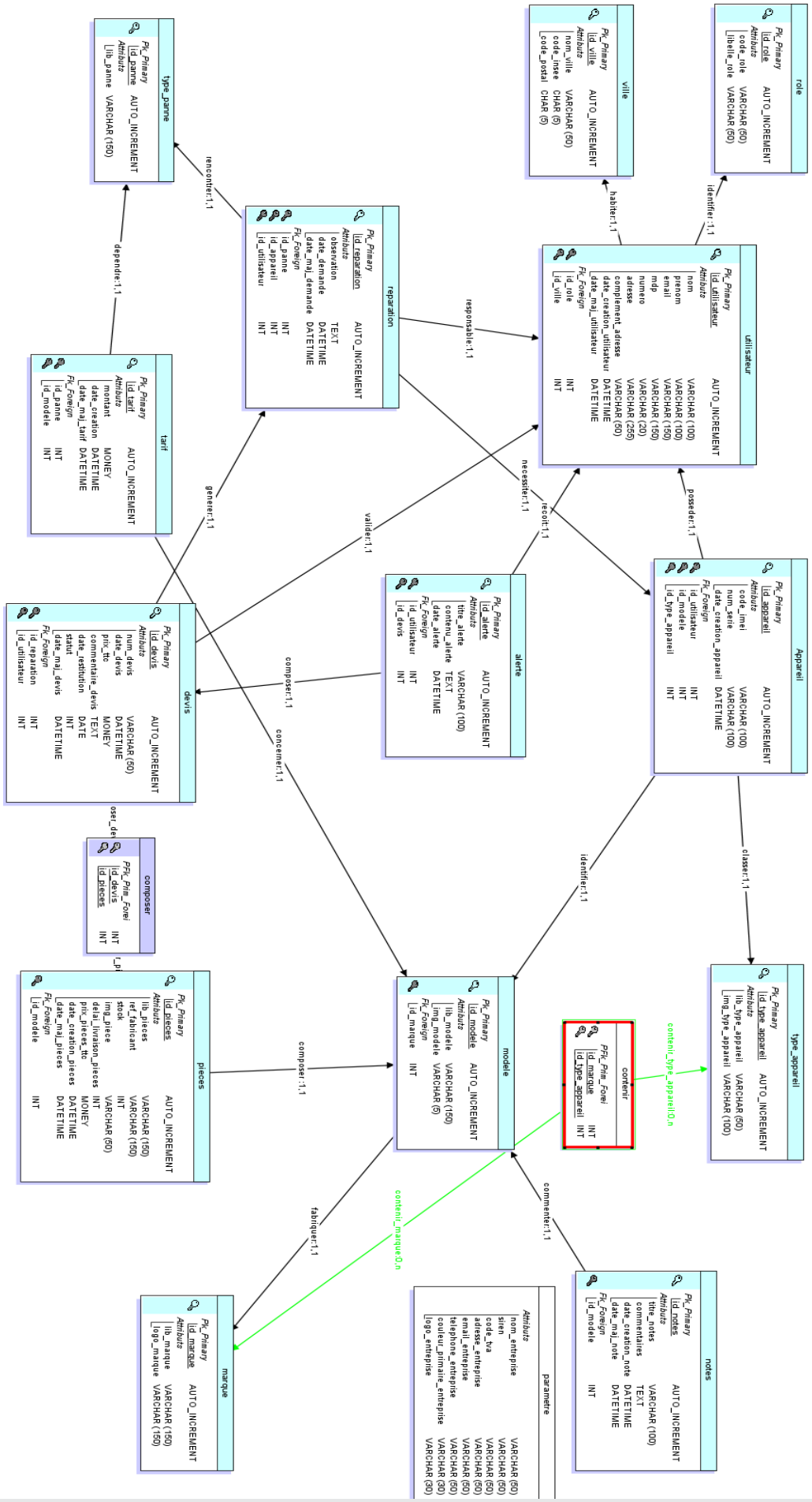
Prenons par exemple l'entité **utilisateur**. En utilisant cette entité comme exemple, nous allons explorer ses relations avec d'autres entités et déterminer les cardinalités associées. Cela nous permettra de comprendre comment les utilisateurs interagissent avec d'autres parties du système et quelles sont les contraintes qui régissent ces interactions.

Voici les relations avec l'entité **utilisateur** :

- Un **utilisateur** peut posséder plusieurs **appareils**, mais chaque **appareil** est possédé par un seul **utilisateur**.
- Un **utilisateur** peut habiter dans une seule **ville**, mais chaque **ville** peut être habitée par de nombreux **utilisateurs**.
- Un **utilisateur** peut être responsable de plusieurs **réparations**, tandis qu'une **réparation** a un seul responsable qui est un **utilisateur**.
- Un **utilisateur** peut recevoir plusieurs **devis**, mais chaque **devis** est lié à un seul **utilisateur**.
- Un **utilisateur** peut être associé à un seul **rôle**, tandis qu'un **rôle** peut être attribué à plusieurs **utilisateurs**.
- Un **utilisateur** peut recevoir plusieurs **alertes**, tandis qu'une **alerte** est destinée à un seul **utilisateur**.

En examinant le MCD, nous pourrons visualiser de manière claire et concise la façon dont les données sont organisées et interconnectées, ce qui constitue une étape cruciale dans le processus de conception d'une base de données robuste et fonctionnelle.

3.5 Modèle Logique de Données (MLD)



Dans notre Modèle Logique de Données (MLD), nous avons concrétisé la structure et les relations entre les entités définies dans le Modèle Conceptuel de Données (MCD) en une série de 17 tables. Cette expansion reflète l'ajout de deux tables intermédiaires, résultant de la résolution des associations **contenir** entre **type_appareil** et **marque**, et **composer** entre **devis** et **pieces**, qui facilitent la gestion des relations multiples entre ces entités.

Ces tables intermédiaires sont essentielles pour décrire les relations *many-to-many*, où un **type_appareil** peut être associé à plusieurs **marques** et inversement, une **marque** peut être reliée à plusieurs **types_appareils**. De même, un **devis** peut inclure plusieurs **pieces** et une **piece** peut faire partie de plusieurs **devis**, permettant ainsi une facturation et un suivi précis des composants utilisés lors des réparations.

Prenons l'exemple de l'entité **utilisateur** pour illustrer les relations dans le MLD :

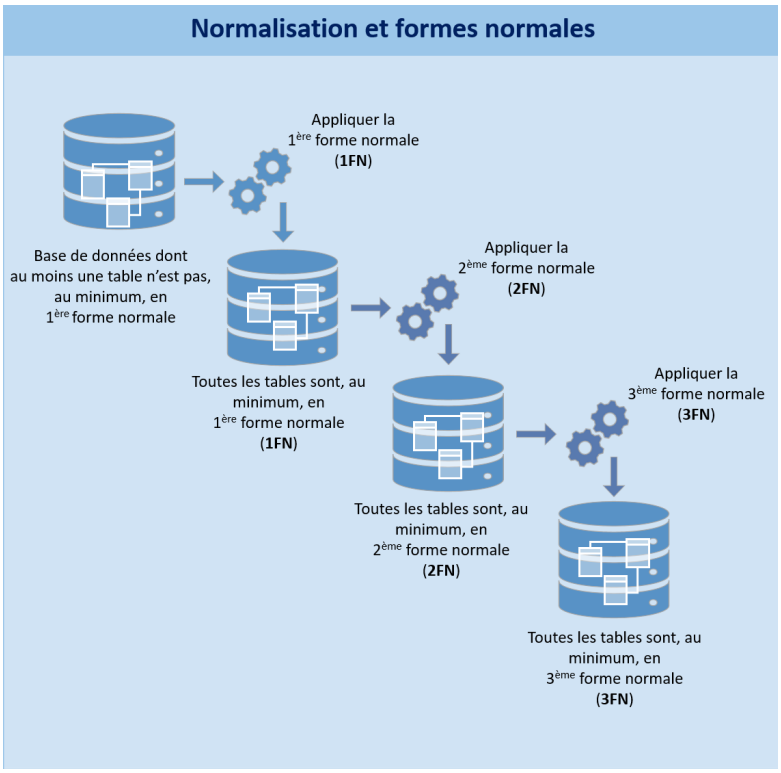
- Un **utilisateur** est au centre de plusieurs interactions. Il possède un ou plusieurs **appareils**, comme indiqué par la clé étrangère **id_utilisateur** dans la table **appareil**.
- La table **utilisateur** est aussi connectée à la table **ville** via la clé étrangère **id_ville**, dénotant qu'un utilisateur réside dans une ville spécifique, tandis qu'une ville peut être le domicile de nombreux utilisateurs.
- Chaque utilisateur peut initier plusieurs **reparations**, et est lié à elles par la clé étrangère **id_utilisateur** dans la table **reparation**.
- En outre, les **utilisateurs** peuvent recevoir diverses **alertes** et sont liés à ces dernières par la clé étrangère **id_utilisateur** dans la table **alerte**.
- Ils peuvent également générer de multiples **devis**, établis par la connexion via la clé étrangère **id_utilisateur** dans la table **devis**.
- Enfin, un **utilisateur** peut endosser un **role** défini par la clé étrangère **id_role** dans la table **utilisateur**, démontrant la distribution des rôles au sein de l'écosystème de l'application.

En résumé, le MLD offre une cartographie détaillée des composants de la base de données, soulignant les relations complexes et les fonctions multiples des **utilisateurs** au sein de DevFix. Cette structuration est le fondement d'une base de données robuste et opérationnelle, prête à prendre en charge la diversité des processus et des interactions au cœur de l'application DevFix.

3.5.1 La base de données

Notre architecture de base de données a été structurée selon les trois formes normales pour garantir efficacité et intégrité des données. En adoptant la première forme normale, nous avons éliminé les données répétitives, assurant que chaque champ contient des données uniques et indivisibles. La seconde forme normale a été mise en œuvre pour renforcer la dépendance des données vis-à-vis de la clé primaire, écartant ainsi les dépendances partielles. Enfin, la troisième forme normale a été appliquée pour supprimer les dépendances transitives, ce qui permet d'assurer une indépendance complète des attributs non-clés par rapport à tout autre attribut non essentiel.

Pour offrir une transparence totale et faciliter les processus de réplication ou de revue, les scripts détaillant l'initialisation et la création de la base de données ont été inclus en annexe. Vous trouverez le script d'initialisation dans l'**Annexe 2** et, de manière cohérente, le script de création dans l'**Annexe 1**, fournissant ainsi un guide pour la mise en place et la gestion de notre base de données



3.6 Maquettes de l'application

Dans le cadre de la conception de l'application, j'ai pris en charge le front-end de l'application utilisateur. Après avoir réalisé une première maquette qui a été initialement validée mais rejetée ultérieurement en raison de problèmes d'expérience utilisateur, j'ai pris l'initiative de revoir entièrement la conception du front-end. En utilisant les retours reçus et en tenant compte des besoins des utilisateurs, j'ai reconstruit le front avec une approche plus avancée. Cela a impliqué une refonte complète des maquettes afin de garantir une expérience utilisateur optimale.

3.6.1 Fonctionnalités détaillées des pages

a) Pour l'app utilisateur :

1) Page d'accueil :
   - La page d'accueil présente différentes sections telles que la sélection du type d'appareil, des informations sur l'application, une FAQ, un lien vers la boutique en ligne, un formulaire de contact, et un footer avec des liens vers les conditions légales et les services détaillés.
   - Fonctionnalités importantes : Formulaire de connexion.
2) Page de choix de la marque :
   - Permet à l'utilisateur de choisir la marque de son appareil.
3) Page de choix du modèle :
   - Permet à l'utilisateur de sélectionner le modèle de son appareil.
4) Page de choix du type de réparation :
   - Permet à l'utilisateur de choisir le type de réparation nécessaire.
5) Page d'inscription :
   - Affiche un récapitulatif du devis.
   - Formulaire de saisie des informations personnelles de l'utilisateur : nom, prénom, email, mot de passe, adresse, ville, code postal, numéro de téléphone, complément d'adresse (facultatif), code IMEI et numéro de série (facultatif).
6) Page de connexion :
   - Affiche un récapitulatif du devis si l'utilisateur possède déjà un compte et il est dans la page « devis ».
   - Formulaire de connexion avec champs pour l'email et le mot de passe.
7) Page conditions légales :
   - Page : Mentions Légales, Politique de Confidentialité, Conditions Générales, Politique Relative aux Cookies
8) Page de connexion après l'inscription :
   - Redirection vers cette page si l'utilisateur souhaite accéder à son devis dans son profil.
9) Tableau de bord de l'utilisateur :
   - Permet à l'utilisateur de voir la liste de ses devis, leur avancement et de modifier ses données personnelles.
10) Page de contact :
   - Formulaire de contact pour permettre aux utilisateurs de contacter l'entreprise, avec des champs pour le nom, l'email, le téléphone, le sujet et le message.

- Informations de l'entreprise telles que le numéro de SIRET, le téléphone de contact et l'email.

11) Page d'erreur de connexion :
    - Affiche un message d'erreur en cas de problème de connexion.

b) Pour l'app admin :

1) Tableau de bord de l'administrateur :
    - Affiche des chiffres clés comme le nombre de devis clôturés, le nombre d'utilisateurs et le chiffre d'affaires.
2) Page de gestion des devis :
    - Liste les devis avec des options pour les éditer, les accepter, et modifier leur état (accepté, en attente, en cours, terminé), ainsi que la possibilité d'affecter un devis à un employé.
3) Pages de gestion des entités :
    - Ajout, édition et suppression de pièces, types d'appareil, marques, modèles, types de réparation, clients et employés.

3.6.2 Évolution potentielle :

Dans le cadre des évolutions futures de l'application DevFix, une attention particulière sera portée à l'amélioration du processus de facturation. Ainsi, il est envisagé de développer une fonctionnalité permettant aux utilisateurs et aux administrateurs de générer des devis et des factures au format PDF directement depuis l'interface de l'application.

Cette fonctionnalité avancée offrira plusieurs avantages :

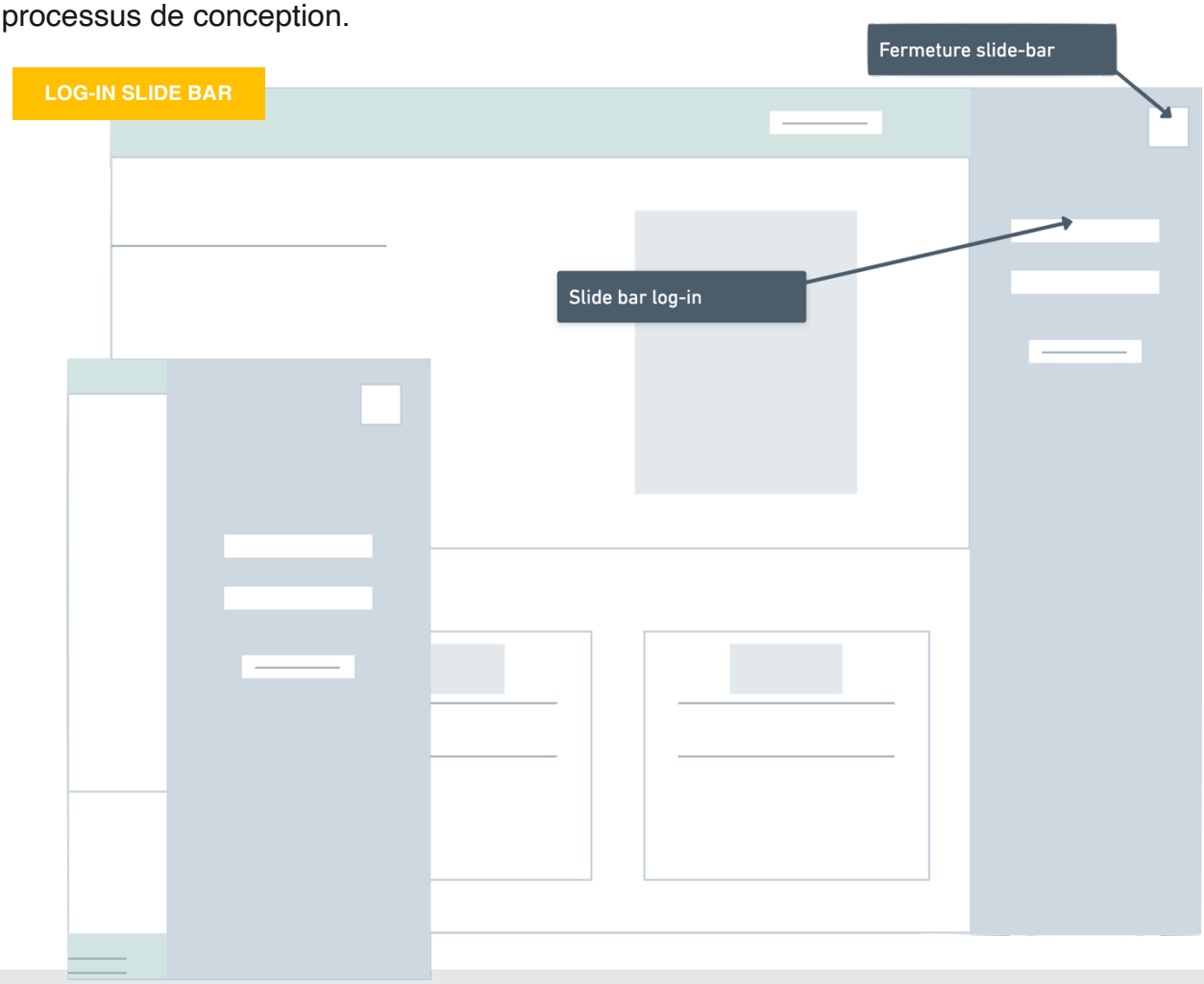
- **Accessibilité :** Les utilisateurs pourront télécharger ou imprimer les devis et factures pour leur propre comptabilité ou pour faciliter les démarches administratives.
- **Sécurité :** Les documents PDF pourront être sécurisés par des fonctionnalités telles que le filigrane et la signature électronique, assurant l'authenticité et la non-modification des informations.
- **Conformité :** Les factures générées respecteront les normes comptables en vigueur, incluant toutes les mentions légales nécessaires.
- **Personnalisation :** Les documents pourront être personnalisés avec le logo de l'entreprise, contribuant ainsi à renforcer l'image de marque de DevFix auprès de ses clients.

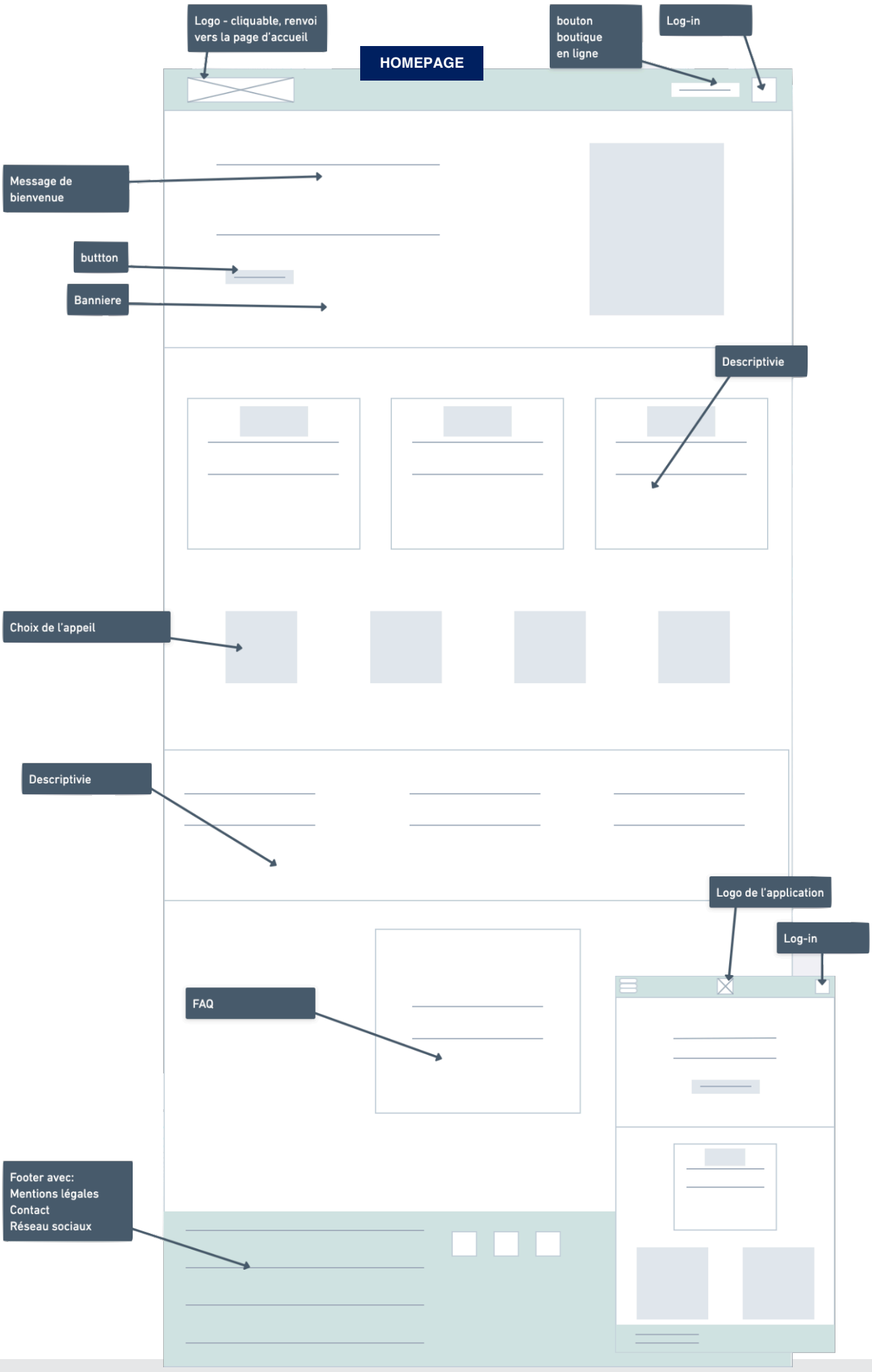
- **Automatisation :** La création des documents sera automatisée en se basant sur les données saisies dans l'application, réduisant les erreurs humaines et accélérant le processus global.
- **Intégration :** L'intégration de cette fonctionnalité avec les modules existants de l'application permettra une cohérence et une efficacité accrues dans le suivi des réparations et des paiements.

En ajoutant la création de devis et de factures en PDF, DevFix répondra encore mieux aux besoins des utilisateurs, tout en simplifiant et en modernisant les processus administratifs internes. Cela représente une étape significative vers une expérience client plus fluide et une gestion d'entreprise optimisée.

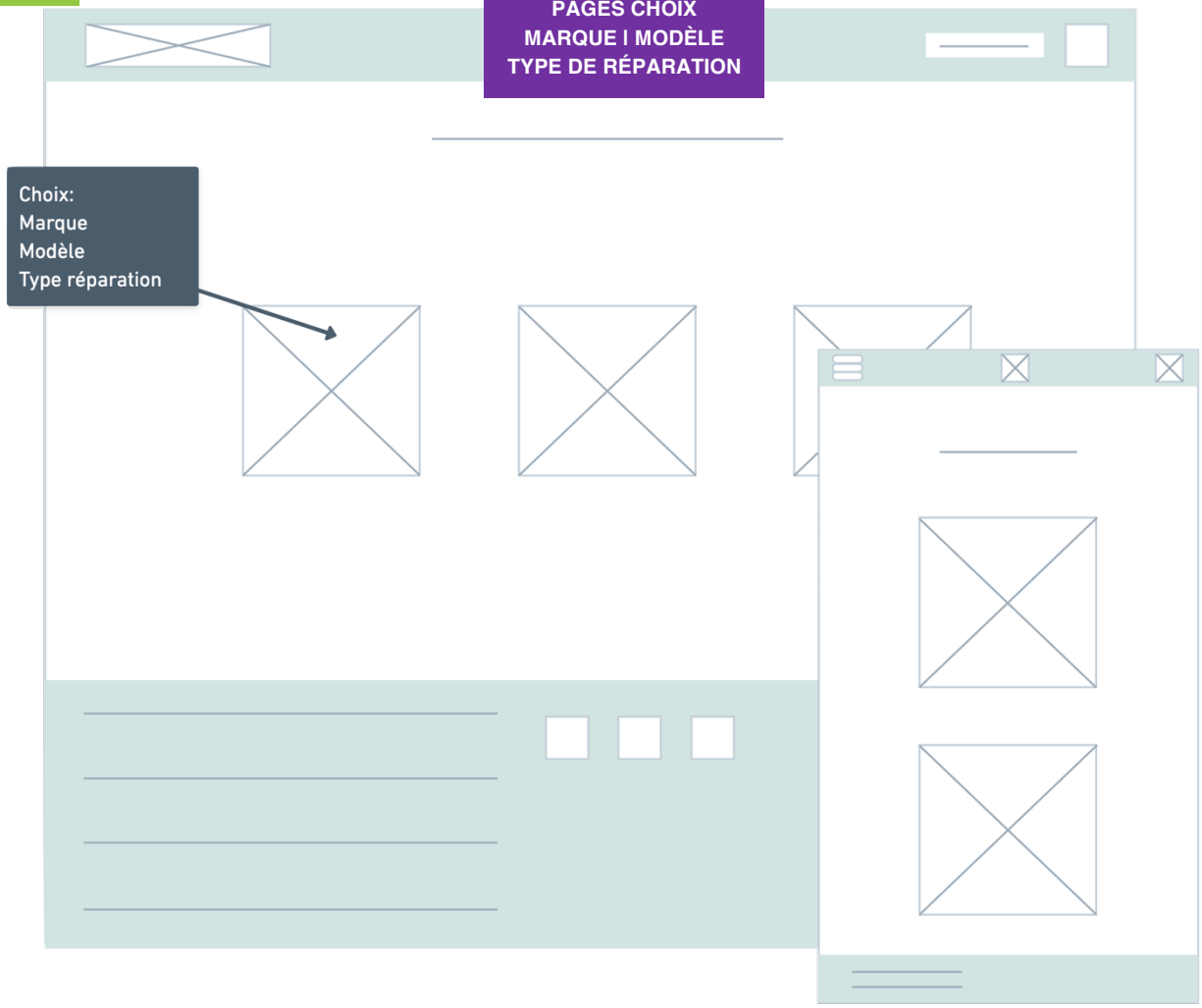
3.6.3 Exemples de Wireframes

Dans cette phase du développement, j'ai également été responsable de la création des wireframes. Ces schémas visuels ont été élaborés pour représenter de manière concrète et détaillée les différentes pages et interactions de l'application. En utilisant ces wireframes comme guide, j'ai pu diriger efficacement le développement de la nouvelle version de l'interface utilisateur, en assurant une cohérence et une clarté tout au long du processus de conception.

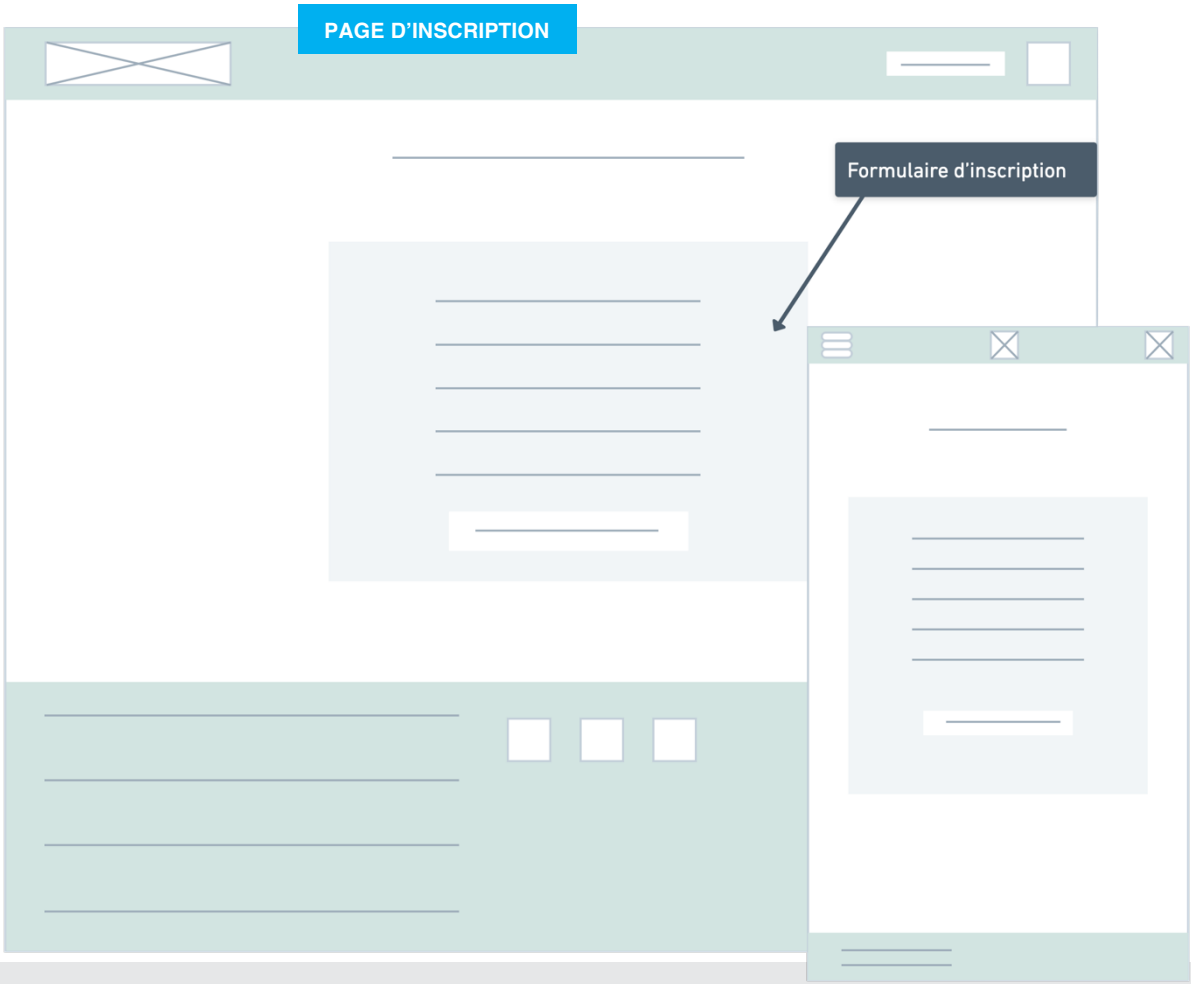


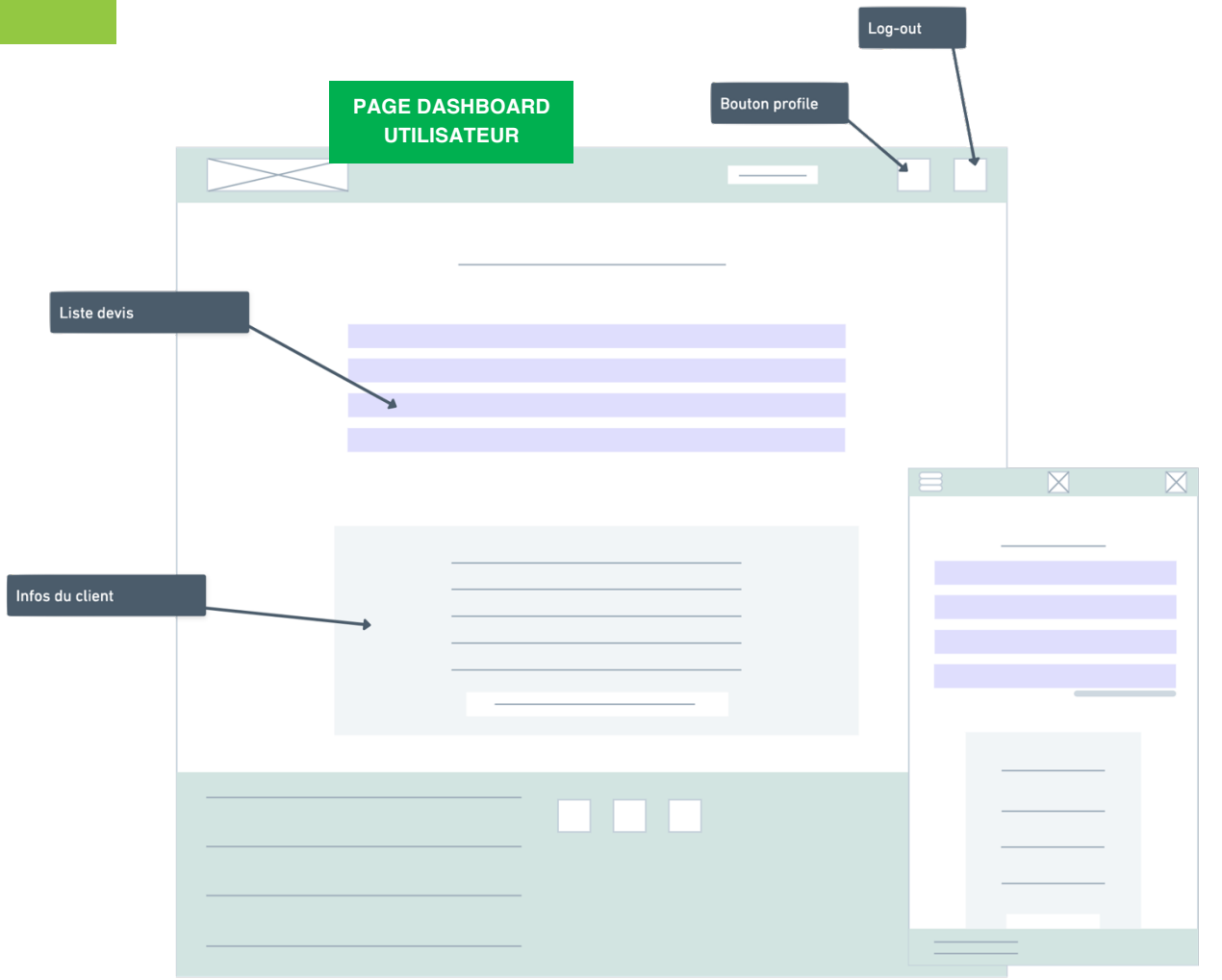

HOMEPAGE
Logo - cliquable, renvoi vers la page d'accueil
bouton boutique en ligne
Log-in
Message de bienvenue
buttton
Banniere
Descriptivie
Choix de l'appeil
Descriptivie
Logo de l'application
Log-in
FAQ
Footer avec:
Mentions légales
Contact
Réseau sociaux

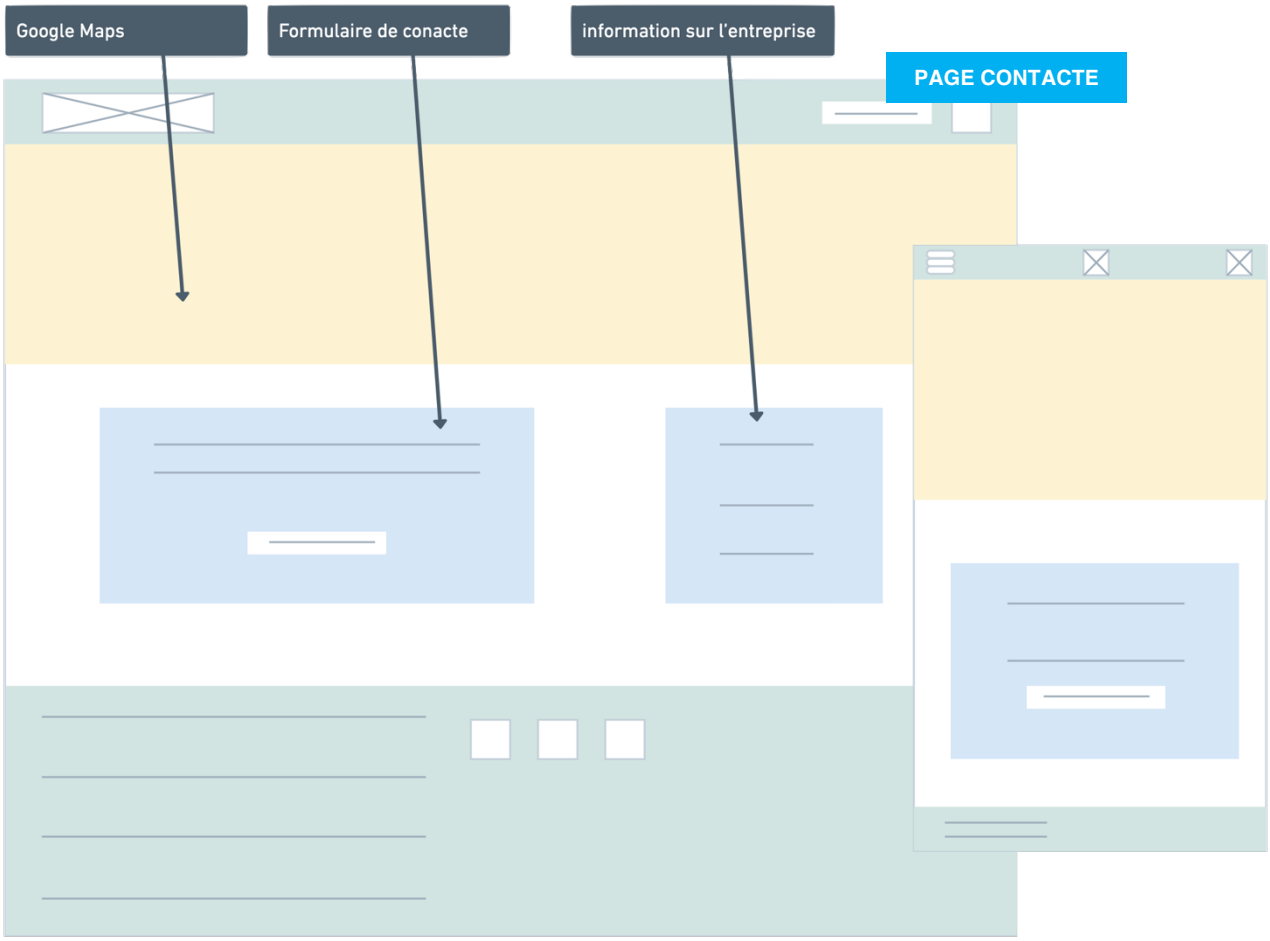
PAGES CHOIX
MARQUE I MODÈLE
TYPE DE RÉPARATION



PAGE D'INSCRIPTION



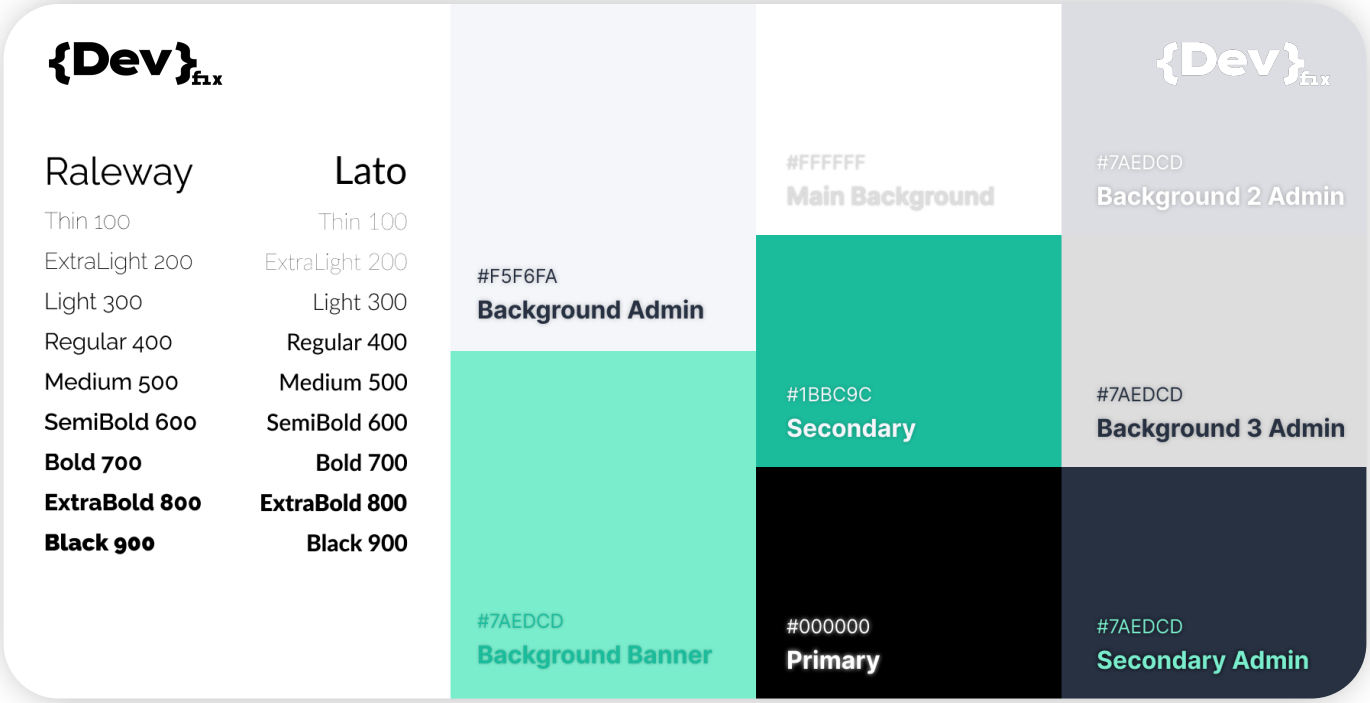

PAGE DASHBOARD UTILISATEUR
Log-out
Bouton profile
Liste devis
Infos du client


Google Maps
Formulaire de conacte
information sur l'entreprise
PAGE CONTACTE

3.6.4 Charte graphique et logo

La charte graphique de l'application DevFix est un élément clé de son identité visuelle, dictant l'utilisation cohérente des couleurs, des polices et du logo à travers toutes les interfaces utilisateur. La palette de couleurs choisie s'articule autour de teintes apaisantes et modernes, avec des nuances de vert turquoise (#1BBC9C et #7AEDCD) qui servent de couleurs secondaires, tant pour l'interface utilisateur que pour l'administration. Ces couleurs sont complétées par des fonds variés, du blanc pur (#FFFFFF) pour le fond principal de l'application, au gris doux (#F5F6FA) pour le fond de l'administration, et un bleu plus soutenu (#7AEDCD) pour les bannières.

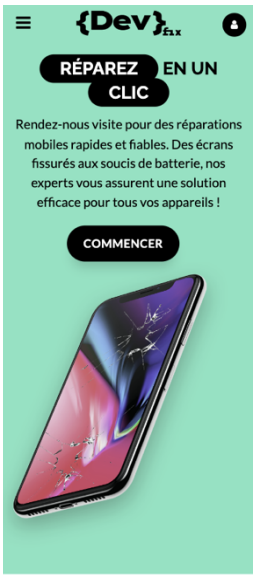
Les polices sélectionnées, Raleway et Lato, sont utilisées dans une gamme de poids allant de Thin 100 à Black 900, permettant une hiérarchisation claire du texte et une lisibilité optimale. La typographie a été choisie pour sa lisibilité et son esthétique contemporaine, contribuant à l'expérience générale de navigation.

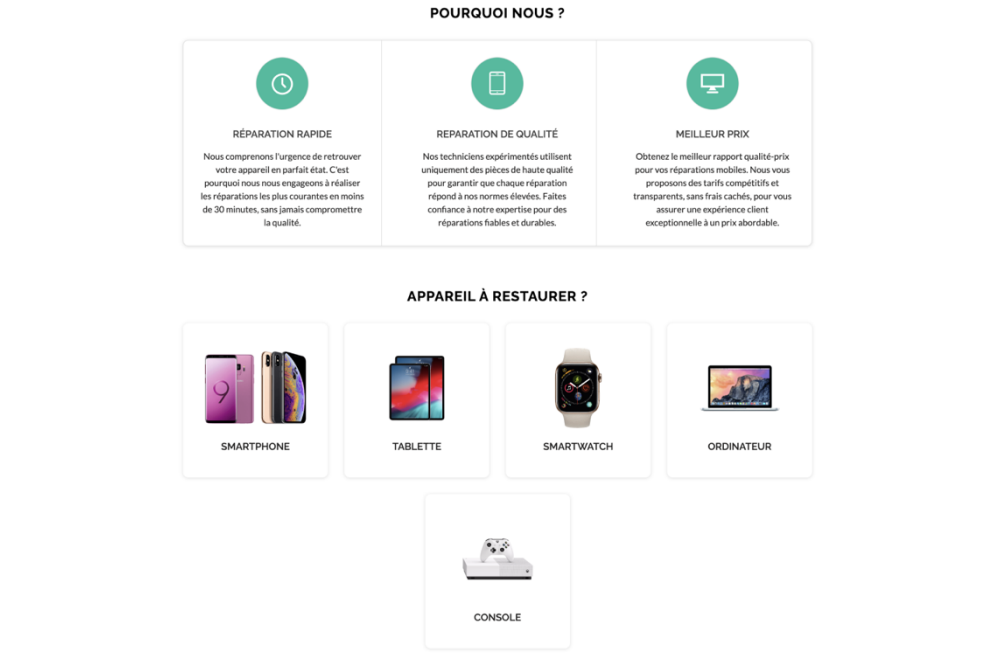


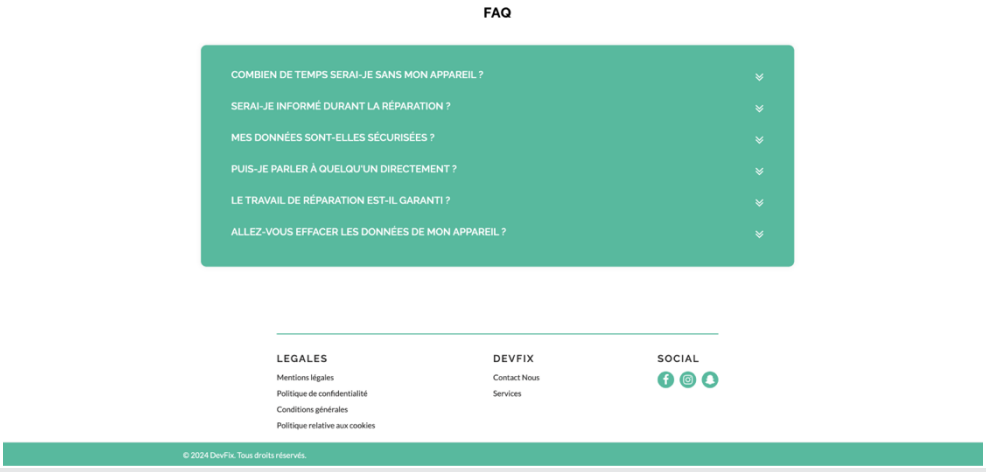
Le logo, en harmonie avec la charte graphique, intègre ces éléments pour créer une image de marque forte et reconnaissable. Il synthétise l'engagement de DevFix envers la qualité et la modernité, et est conçu pour être adaptable à divers supports et contextes, assurant la visibilité et la cohérence de la marque sur toutes les plateformes.

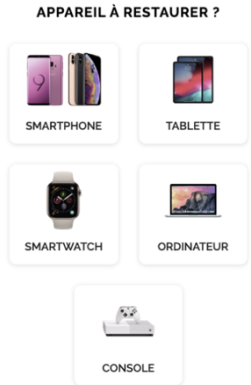
Cette charte graphique est conçue pour être facilement déployable et modifiable, garantissant que l'application puisse évoluer avec les tendances du design tout en maintenant une identité constante.

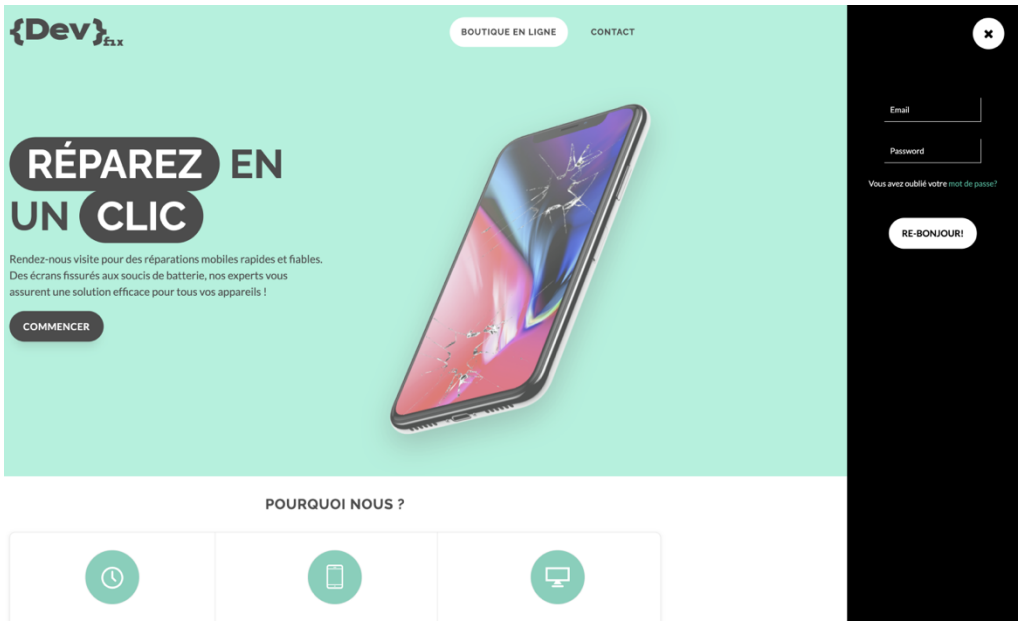
3.6.5 Exemples de maquettes



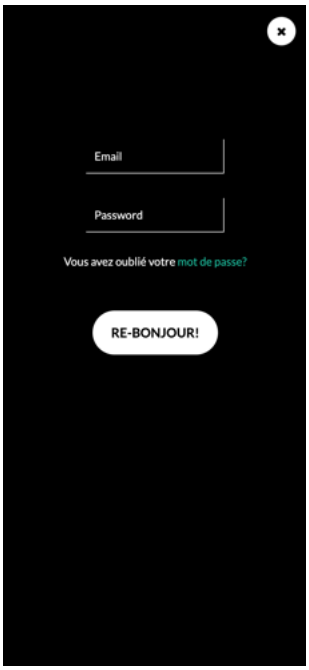


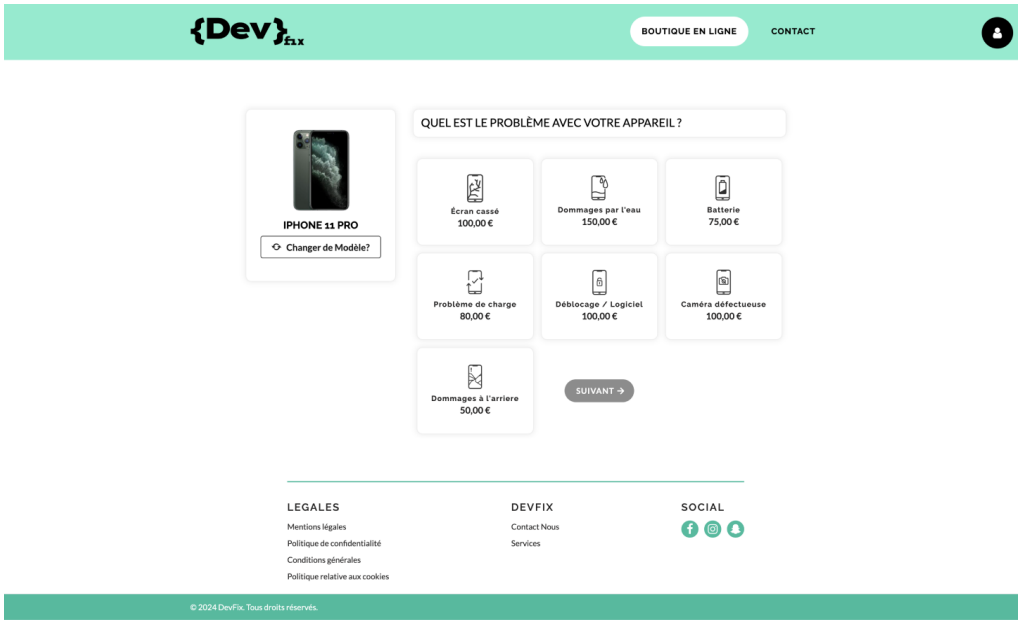




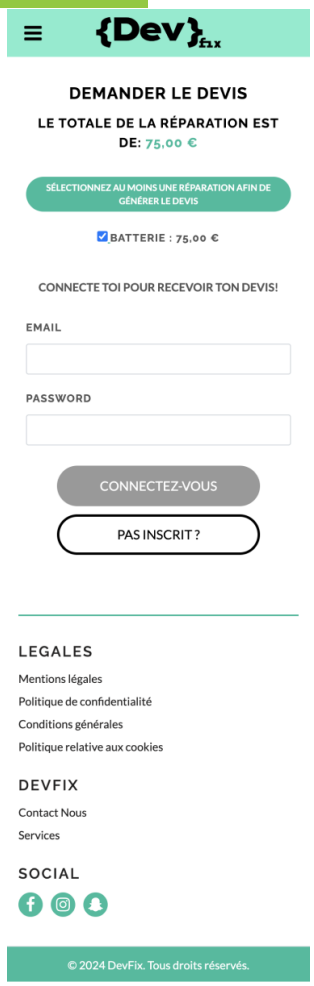

{Dev}fix
BOUTIQUE EN LIGNE
CONTACT
RÉPAREZ EN UN CLIC
Rendez-nous visite pour des réparations mobiles rapides et fiables. Des écrans fissurés aux soucis de batterie, nos experts vous assurent une solution efficace pour tous vos appareils !
COMMENCER
POURQUOI NOUS ?

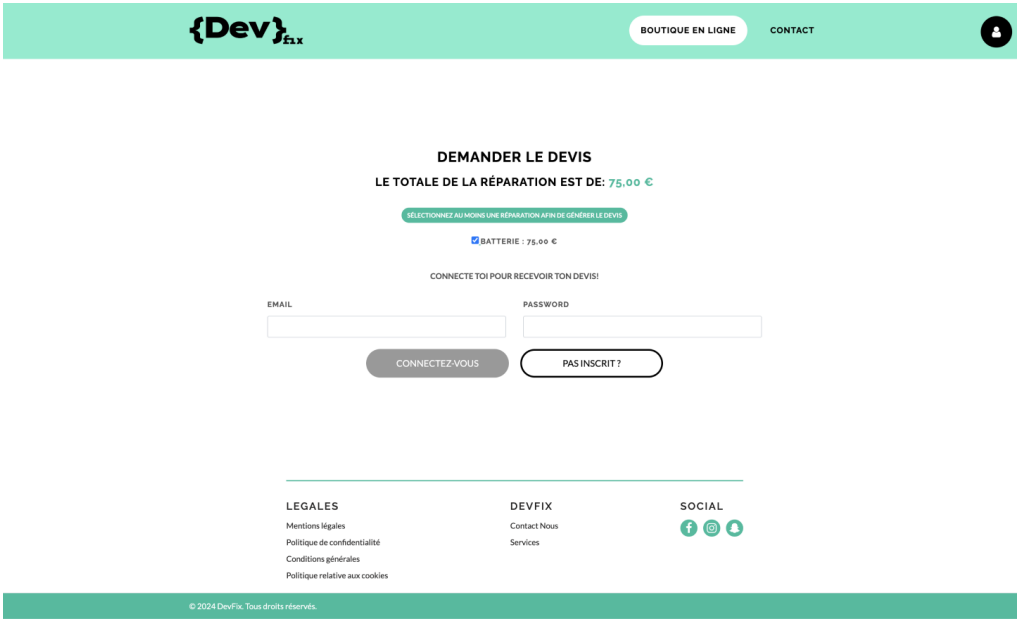
Email
Password
Vous avez oublié votre mot de passe?
RE-BONJOUR!


Email
Password
Vous avez oublié votre mot de passe?
RE-BONJOUR!

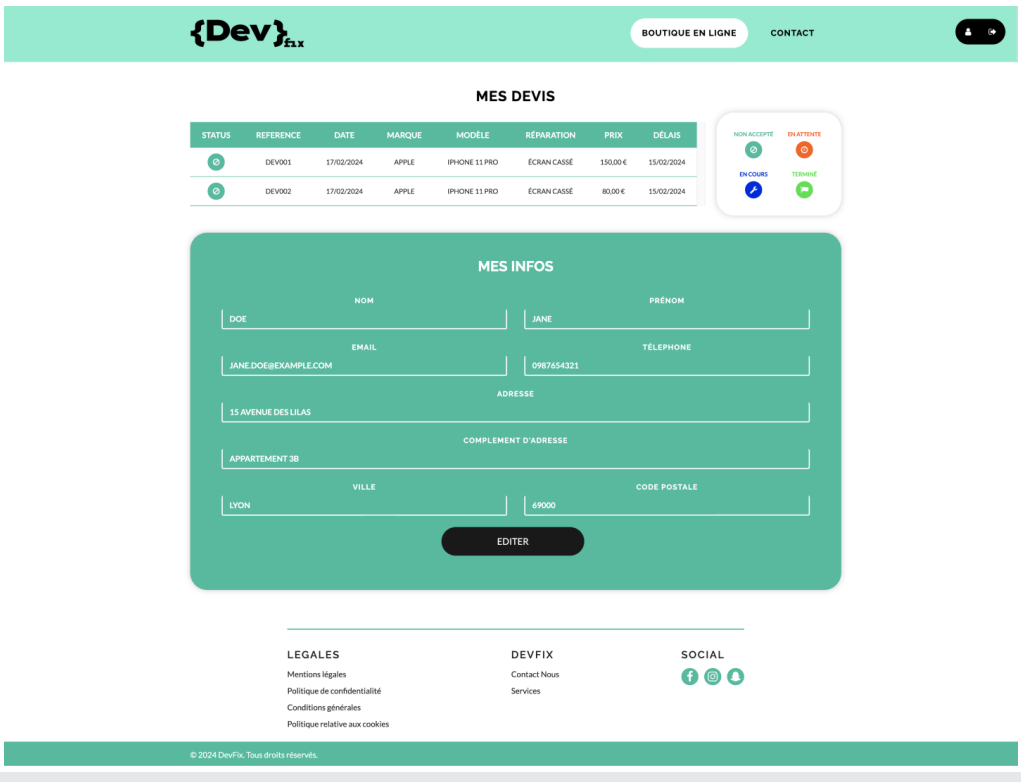

{Dev}fix
BOUTIQUE EN LIGNE
CONTACT
IPHONE 11 PRO
Changer de Modèle?
QUEL EST LE PROBLÈME AVEC VOTRE APPAREIL ?
Écran cassé 100,00 €
Dommages par l'eau 150,00 €
Batterie 75,00 €
Problème de charge 80,00 €
Déblocage / Logiciel 100,00 €
Caméra défectueuse 100,00 €
Dommages à l'arriere 50,00 €
SUIVANT →
LEGALES
Mentions légales
Politique de confidentialité
Conditions générales
Politique relative aux cookies
DEVFIX
Contact Nous
Services
SOCIAL
© 2024 DevFix. Tous droits réservés.


{Dev}fix
IPHONE 11 PRO
Changer de Modèle?
QUEL EST LE PROBLÈME AVEC VOTRE APPAREIL ?
Écran Cassé 100,00 €
Dommages Par L'eau 150,00 €
Batterie 75,00 €
Problème De Charge 80,00 €
Déblocage / Logiciel 100,00 €
Caméra Défectueuse 100,00 €
Dommages À L'arriere 50,00 €
SUIVANT →
LEGALES
Mentions légales
Politique de confidentialité
Conditions générales
Politique relative aux cookies
DEVFIX
Contact Nous
Services
SOCIAL
© 2024 DevFix. Tous droits réservés.


{Dev}fix
DEMANDER LE DEVIS
LE TOTALE DE LA RÉPARATION EST DE: 75,00 €
SÉLECTIONNEZ AU MOINS UNE RÉPARATION AFIN DE GÉNÉRER LE DEVIS
BATTERIE : 75,00 €
CONNECTE TOI POUR RECEVOIR TON DEVIS!
EMAIL
PASSWORD
CONNECTEZ-VOUS
PAS INSCRIT ?
LEGALES
Mentions légales
Politique de confidentialité
Conditions générales
Politique relative aux cookies
DEVFIX
Contact Nous
Services
SOCIAL
© 2024 DevFix. Tous droits réservés.


{Dev}fix
BOUTIQUE EN LIGNE
CONTACT
DEMANDER LE DEVIS
LE TOTALE DE LA RÉPARATION EST DE: 75,00 €
SÉLECTIONNEZ AU MOINS UNE RÉPARATION AFIN DE GÉNÉRER LE DEVIS
BATTERIE : 75,00 €
CONNECTE TOI POUR RECEVOIR TON DEVIS!
EMAIL
PASSWORD
CONNECTEZ-VOUS
PAS INSCRIT ?
LEGALES
Mentions légales
Politique de confidentialité
Conditions générales
Politique relative aux cookies
DEVFIX
Contact Nous
Services
SOCIAL
© 2024 DevFix. Tous droits réservés.


{Dev}fix
MES DEVIS
STATUS
REFERENCE
DATE
MARQUE
DEV009
19/02/2024
APPLE
DEV009
19/02/2024
APPLE
DEV011
21/02/2024
APPLE
NON ACCEPTÉ
EN ATTENTE
EN COURS
TERMINÉ
MES INFOS
NOM
DOE
PRÉNOM
JANE
EMAIL
JANE.DOE@EXAMPLE.COM
TÉLEPHONE
0987654321
ADRESSE
15 AVENUE DES LILAS
COMPLEMENT D'ADRESSE
APPARTEMENT 3B
VILLE
LYON
CODE POSTALE
69000
EDITER
LEGALES
Mentions légales
Politique de confidentialité
Conditions générales
Politique relative aux cookies
DEVFIX
Contact Nous
Services
SOCIAL
© 2024 DevFix. Tous droits réservés.

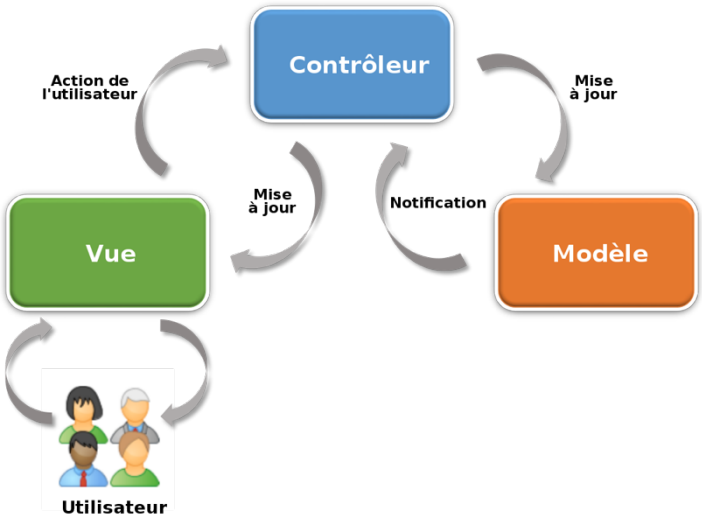

{Dev}fix
BOUTIQUE EN LIGNE
CONTACT
MES DEVIS
STATUS
REFERENCE
DATE
MARQUE
MODÈLE
RÉPARATION
PRIX
DÉLAIS
DEV001
17/02/2024
APPLE
IPHONE 11 PRO
ÉCRAN CASSÉ
150,00 €
15/02/2024
DEV002
17/02/2024
APPLE
IPHONE 11 PRO
ÉCRAN CASSÉ
80,00 €
15/02/2024
NON ACCEPTÉ
EN ATTENTE
EN COURS
TERMINÉ
MES INFOS
NOM
DOE
PRÉNOM
JANE
EMAIL
JANE.DOE@EXAMPLE.COM
TÉLEPHONE
0987654321
ADRESSE
15 AVENUE DES LILAS
COMPLEMENT D'ADRESSE
APPARTEMENT 3B
VILLE
LYON
CODE POSTALE
69000
EDITER
LEGALES
Mentions légales
Politique de confidentialité
Conditions générales
Politique relative aux cookies
DEVFIX
Contact Nous
Services
SOCIAL
© 2024 DevFix. Tous droits réservés.

4 Développement de l'application

4.1 Architecture logicielle

L'architecture logicielle de l'application repose sur le modèle MVC (Modèle-Vue-Contrôleur), une méthodologie éprouvée pour le développement web qui favorise la séparation des préoccupations. En utilisant Symfony 6.4, nous avons structuré le backend de manière à ce que chaque composant de l'application - qu'il s'agisse de la logique métier, de la manipulation des données ou de la présentation - ait un rôle clairement défini et indépendant. Symfony nous a permis de créer un ensemble cohérent de contrôleurs pour gérer les requêtes entrantes, de modèles pour représenter nos données et de vues pour afficher les informations aux utilisateurs.

Cette architecture logicielle a été conçue pour être robuste, évolutive et maintenable, en s'appuyant sur les meilleures pratiques et les outils modernes du développement web.



4.1.1 Mention de l'utilisation de Symfony 6.4, Twig, PHPMyAdmin, SQL

Twig a été adopté comme notre moteur de template pour la gestion des vues, principalement pour ses performances optimisées et sa facilité d'intégration avec Symfony. Twig nous permet de séparer le code HTML du code PHP, offrant ainsi une meilleure maintenance du code et des possibilités de personnalisation sans affecter la logique métier sous-jacente.

Concernant la gestion des données, PHPMyAdmin a été un outil précieux pour l'interaction avec notre base de données MySQL pendant la phase de développement. Nous avons utilisé l'ORM Doctrine de Symfony pour gérer les requêtes vers la base de données, ce qui nous a permis de manipuler les données de manière plus abstraite et sécurisée. Les entités ont été définies dans Symfony, et les scripts d'initialisation et de création ont été exécutés pour structurer notre base de données. La connexion entre Symfony et la base de données a été établie via la configuration dans le fichier **.env**, et les données des formulaires Twig sont récupérées et traitées en utilisant les composants <Form> de Symfony.

4.2 Développement frontend et backend

a) Frontend :

Pour le développement frontend de l'application DevFix, nous avons opté pour une combinaison de technologies web standards et éprouvées. Les maquettes ont été réalisées en utilisant HTML enrichi par le moteur de template Twig, permettant une intégration fluide avec les données dynamiques du backend. JavaScript a été appliqué pour dynamiser les interactions sur le client et offrir une expérience utilisateur réactive. Le CSS, quant à lui, a mis en forme l'application en respectant la charte graphique et en assurant une interface utilisateur esthétique et fonctionnelle. De plus, j'ai utilisé des classes de Bootstrap pour faciliter le développement et assurer la cohérence visuelle à travers l'ensemble de l'application.

b) Backend :

Du côté du backend, la logique métier est géré par Symfony, qui offre un cadre de travail structuré pour développer des applications web complexes. L'authentification est traitée par les composants de sécurité de Symfony, qui offrent un système robuste pour gérer les sessions utilisateur, l'encryption des mots de passe et la protection contre les vulnérabilités courantes. La gestion des données est orchestrée par l'ORM Doctrine, qui permet de manipuler la base de données via des objets PHP, rendant les interactions avec la base de données plus intuitives et sécurisées. Ce système abstrait les requêtes SQL, facilitant les opérations CRUD (créer, lire, mettre à jour, supprimer) et assurant l'intégrité des données.

4.3 Implémentation de l'orienté objet

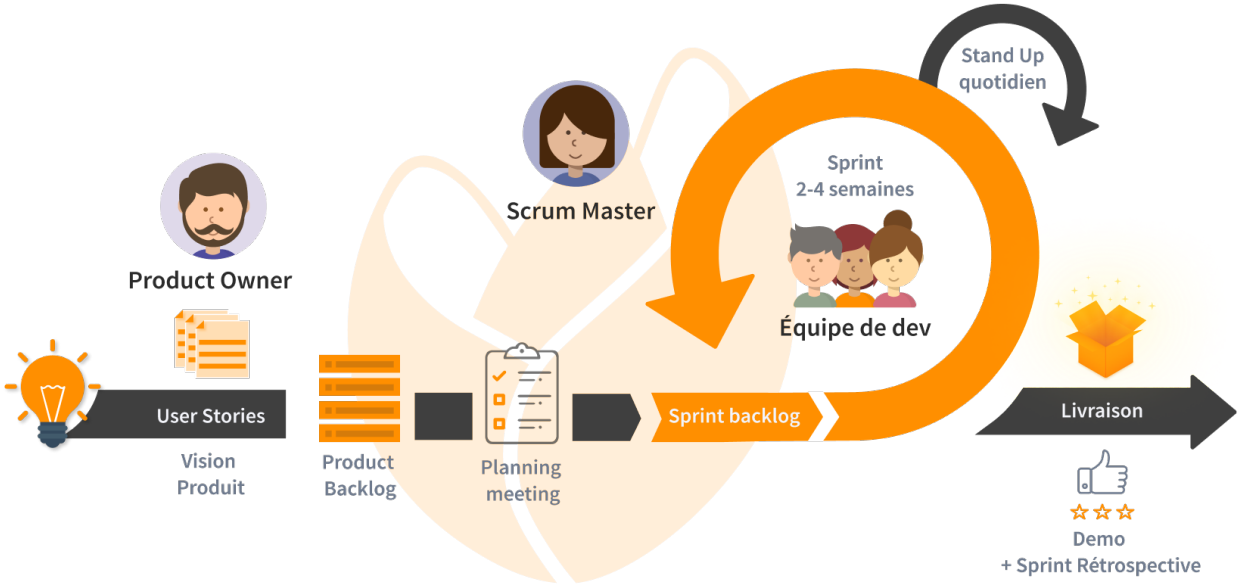
Dans la construction de DevFix, nous avons adopté une approche orientée objet pour organiser le code en unités modulaires. Cela implique que chaque fonctionnalité, comme la gestion des utilisateurs ou la création de devis, est encapsulée dans des classes avec des attributs et des méthodes spécifiques. Cette encapsulation facilite la maintenance du code, rendant le système plus robuste et évolutif.

De plus, nos classes de contrôleurs héritent de la classe de base de Symfony, **AbstractController**. Cette approche d'héritage nous permet d'utiliser les méthodes de base fournies par Symfony, offrant ainsi une structure cohérente et une utilisation efficace des fonctionnalités de ce framework. L'héritage nous permet également de bénéficier des mises à jour et des améliorations apportées à la classe de base, garantissant ainsi une évolutivité continue de notre application.

5 Méthodologie de travail

5.1 Adoption de la méthodologie Scrum

Pour le développement nous avons adopté la méthodologie Scrum, une approche agile qui nous a permis d'organiser notre travail en cycles de développement courts et itératifs, appelés sprints. Au cours de ces sprints, qui ont duré une semaine chacun, nous avons traversé les phases de planification, de développement, de revue et de rétrospective.



Planification du Sprint : Au début de chaque sprint, nous avons tenu une réunion de planification pour déterminer les fonctionnalités à développer, en les priorisant en fonction de leur valeur pour le projet et leur faisabilité technique.

Développement : Durant le sprint, l'équipe a travaillé sur les tâches définies, en maintenant une communication constante et en effectuant des mises à jour quotidiennes sur l'avancement. Cela nous a permis d'identifier et de résoudre rapidement les obstacles.

Revue du Sprint : À la fin de chaque sprint, nous présentions le travail accompli aux client (l'entreprise DevFix). Cette revue était l'occasion de démontrer les nouvelles fonctionnalités et de recueillir des commentaires pour améliorer les sprints suivants.

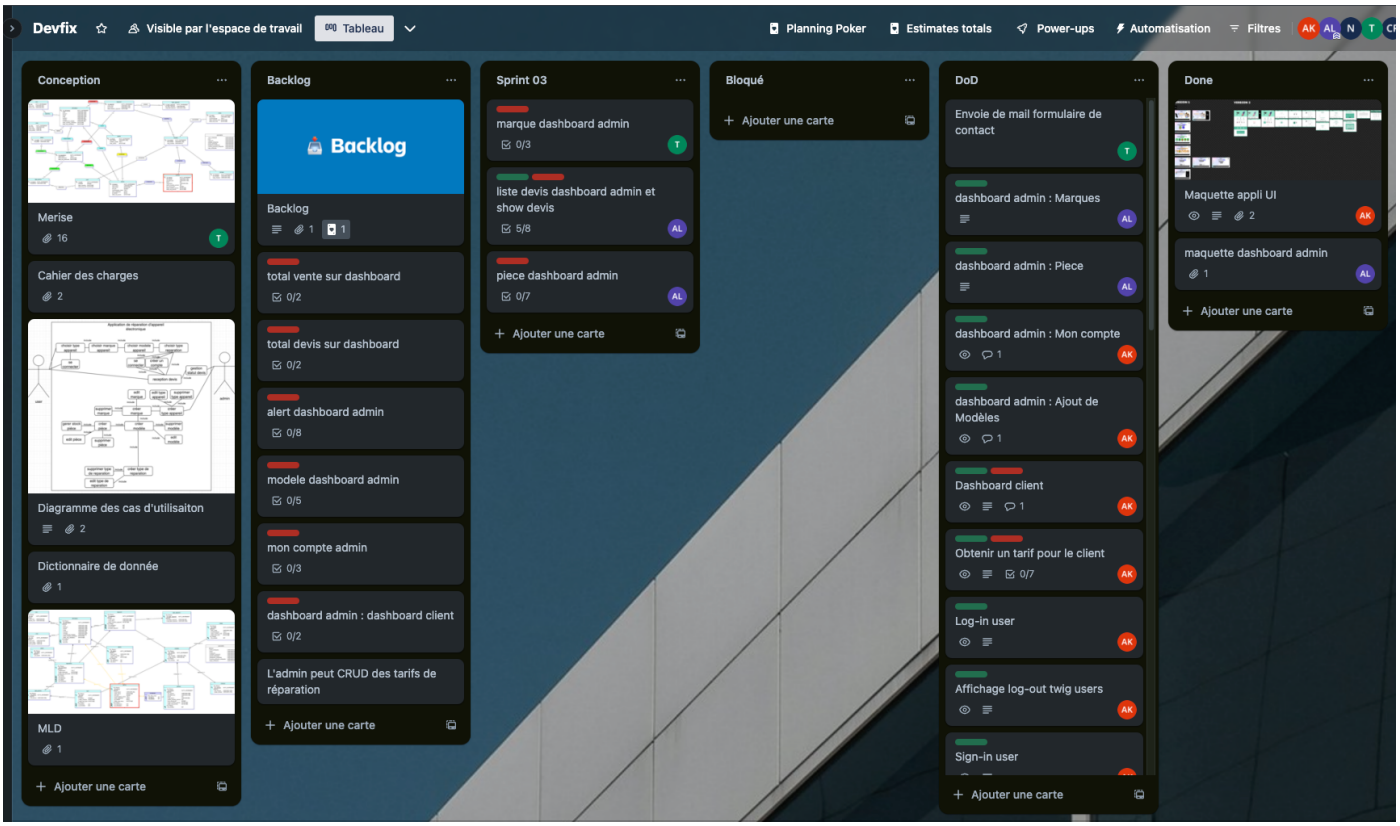
Rétrospective du Sprint : Après la revue, l'équipe se réunissait pour discuter des succès et des défis rencontrés pendant le sprint. Les leçons apprises étaient utilisées pour affiner notre processus pour les sprints suivants.

Grâce à Scrum, nous avons maintenu un rythme de développement soutenu et adaptatif, avec une amélioration continue grâce aux retours réguliers et à l'auto-évaluation. Cette méthode a joué un rôle crucial dans la réussite du projet, en nous aidant à rester alignés sur les objectifs du projet tout en étant réactifs aux besoins changeants.

5.2 Présentation d'une user story exemplative

La user story "Création de devis", que j'ai directement gérée, illustre notre processus de conversion des exigences utilisateurs en fonctionnalités tangibles. Trello a servi d'outil d'organisation pour cette tâche. Elle guide l'utilisateur depuis la sélection d'un appareil jusqu'à la génération d'un devis personnalisé accessible depuis son espace utilisateur.

Ce processus inclut le choix de l'appareil, de la marque, du modèle et de la réparation, et se conclut par une phase d'inscription ou de connexion pour visualiser le devis. Le back-end, où des validations et des requêtes via Doctrine et SQL garantissent la sécurité des données, ainsi que le front-end, où les validations et la protection contre les failles CSRF sont mises en avant, démontrent l'engagement envers la sécurité et la convivialité. Cette réalisation personnelle souligne notre capacité à aligner le développement sur les exigences de sécurité et les besoins des utilisateurs.



5.2.1 Descriptif de l'user story « Création de devis »

Critères d'acceptation révisés :

1. **Accès à la homepage :** L'utilisateur doit pouvoir naviguer facilement sur la page d'accueil, où il trouve les options nécessaires pour définir ses besoins.
2. **Sélection des options :** La possibilité de choisir le type d'appareil, la marque, le modèle, et le type de réparation.
3. **Inscription ou Connexion :**
    a) **Nouveaux utilisateurs :** Ils peuvent s'inscrire en remplissant un formulaire pour accéder à leur devis.
    b) **Utilisateurs existants :** Ils se connectent via un formulaire dédié pour retrouver leur devis.
4. **Formulaire d'inscription :** Doit collecter les informations essentielles telles que nom, email, téléphone, adresse, avec des champs facultatifs pour le code IMEI et le numéro de série.
5. **Redirection et Confirmation :** Après inscription ou connexion, l'utilisateur est redirigé vers son espace personnel où le devis est disponible.
6. **Création en BDD :** À chaque demande de devis, si un client n'a pas de compte, il est créé en base de données, ainsi qu'un devis associé à son compte.
7. **Intégration Dashboard Admin :** Le devis généré est également intégré au dashboard administrateur pour suivi.

Scénarios :

- **Nouveau client :** Un parcours sans compte préexistant, comprenant la sélection d'options, l'inscription, et la réception du devis dans l'espace utilisateur.
- **Client existant :** Un client avec compte accède à son devis après avoir sélectionné des options et s'être connecté.
- **Client indécis :** Possibilité de sélectionner des options sans finaliser la demande de devis.

Cette user story encapsule non seulement le cheminement fonctionnel mais sert également de fondement pour expliquer l'implémentation technique, de la sécurité aux requêtes Doctrine et SQL, illustrant une interaction complète entre l'utilisateur et le système.

6 Développement Technique de l'User Story "Création de devis"

6.1 Conception Frontend

Dans le développement frontend de l'application DevFix, j'ai utilisé le templating de Twig pour optimiser la structure des pages web. J'ai codé le header et le footer dans base.html.twig (**Annexe 3**), ce qui a permis d'obtenir un chargement de contenu dynamique et rapide. Cette méthode garantit que le contenu principal varie selon la page, tout en maintenant une uniformité dans la navigation et la présentation générale du site.

Choix du type d'appareil, cette section est dans le template : homepage

```
{% for typeAppareil in typeAppareils %}
    <div class="col-md-4 col-sm-4 col-xl-3 col-lg-4 col-6 device_category">
        <a href="/device-brand/{{ typeAppareil.idTypeAppareil }}" class="card align-items-center">
            <div class="img inner mx-auto">
                <img class="main_img" src="{{ asset('user/images/categories/' ~ typeAppareil.imgTypeAppareil) }}"
                    alt="{{ typeAppareil.libTypeAppareil }}">
                <img class="main_hover_img" src="{{ asset('user/images/categories/hover_' ~ typeAppareil.imgTypeAppareil) }}"
                    alt="{{ typeAppareil.libTypeAppareil }}">
                <h4>{{ typeAppareil.libTypeAppareil }}</h4>
            </div>
        </a>
    </div>
{% endfor %}
```

Choix de la marque, cette section est dans le template : device-brand

```
{% if brands %}
    <ul class="clients-grid grid-6 bottommargin-sm clearfix brand_category_selection center-grid device-brands">
        {% for brand in brands %}
            <li>
                <a href="{{ path('app_device-model', {'idMarque': brand.idMarque}) }}" class="d-flex align-items-center">
                    <div class="inner mx-auto">
                        <img src="{{ asset('user/images/brand/' ~ brand.logoMarque) }}" alt="{{ brand.libMarque }}">
                    </div>
                </a>
            </li>
        {% endfor %}
    </ul>
{% else %}
    <div class="center" role="alert">
        <p>Pas de marque disponible pour cette appareil</p>
        <div class="list-inline">
            <a href="{{ url('app_homepage') }}#product-categories" class="btn btn-lg mr-3 btn-outline-light rounded-pill ">
                recommencer
            </a>
        </div>
    </div>
{% endif %}
```

Choix du modéle, cette section est dans le template : device-model

```
{% if modeles %}
    {% for modele in modeles %}
        <article class="portfolio-item pf-media pf-icons">
            <div class="inner">
                <div class="portfolio-image">
                    <a href="{{ path('app_device-repair', {'idModele': modele.idModele}) }}">
                        <img src="{{ asset('user/images/mobile/' ~ modele.imgModele) }}" alt="{{ modele.libModele }}">
                    </a>
                    <div class="portfolio-overlay">
                        <a href="{{ path('app_device-repair', {'idModele': modele.idModele}) }}" class="center-icon">
                            <i class="icon-line-ellipsis"></i>
                        </a>
                    </div>
                </div>
                <div class="portfolio-desc">
                    <span>{{ modele.libModele }}</span>
                </div>
            </div>
        </article>
    {% endfor %}
{% else %}
    <div class="center" role="alert">
        <p>Pas de modèles disponible pour cette marque</p>
        <div class="list-inline">
            <a href="{{ url('app_homepage') }}#product-categories" class="btn btn-lg mr-3 btn-outline-light rounded-pill ">
                recommencer
            </a>
        </div>
    </div>
{% endif %}
```

Choix du modéle, cette section est dans le template : device-repair

```
{% if tarifs %}
    <form action="{{ path('app_devis') }}" method="post">
        {% for tarif in tarifs %}
            <label for="{{ tarif.idTarif }}" id="{{ tarif.idPanne.libPanne }}"
                class="radios font-body ls0 nott rounded-0 radio_btn d-flex align-items-center btn btn-outline-dark  image_have"
                data-input-type="radio">
                <div class="inner">
                    <div class="images">
                        <img src="{{ asset('user/images/' ~ tarif.idPanne.imgPanne) }}" alt="{{ tarif.idPanne.libPanne }}"
                            width="30px" id="{{ tarif.idPanne.libPanne }}">
                    </div>
                    <input type="checkbox" name="tarifIds[]" id="{{ tarif.idTarif }}"
                        value="{{ tarif.idTarif }}" autocomplete="off" data-default="0">
                    <h5 class="mb-0">{{ tarif.idPanne.libPanne }}</h5>
                    <div class="item_price text-primary h5 mb-0">
                        <strong>{{ tarif.montant|number_format(2, ',', ' ') }} €</strong>
                    </div>
                </div>
            </label>
        {% endfor %}
        <div class="radio_btn d-flex align-items-center btn suivant-button-container">
            <div class="inner">
                <button type="submit" class="button button-3d nomargin sell-this-device" id="get-price-btn" role="button">
                    Suivant
                    <i class="icon-arrow-right"></i>
                </button>
            </div>
        </div>
    </form>
{% else %}
```

Dans les captures d'écran, on observe comment Twig permet de dynamiser l'affichage des éléments en parcourant les résultats de la base de données à l'aide d'une boucle `for`. J'utilise également des conditions `if` pour vérifier la présence de marques, modèles, et types de réparations. Par exemple, si aucun modèle en ligne n'est disponible pour la marque Sony, un message s'affiche pour informer le client de cette absence. Cette méthode garantit que l'interface utilisateur reste informative et réactive aux sélections de l'utilisateur, améliorant ainsi l'expérience utilisateur en fournissant des feedbacks instantanés sur la disponibilité des options.

6.2 Validation des Données Utilisateur

6.2.1 Recherches

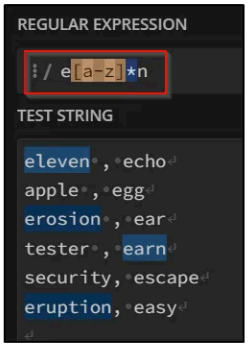
J'ai entrepris des recherches sur Internet pour identifier les meilleures pratiques en matière de validation et de contrôle des données utilisateur. L'objectif était de comprendre les points critiques où l'application nécessite une sécurité et un contrôle solides pour prévenir les vulnérabilités et garantir une expérience utilisateur sécurisée.

Ces recherches m'ont permis de me familiariser avec une variété de techniques et de stratégies de validation des données, allant des expressions régulières (regex) pour la validation côté client à l'importance des validations côté serveur pour une double couche de sécurité. J'ai également exploré des concepts avancés tels que la prévention des injections SQL, la protection contre les attaques XSS (Cross-Site Scripting) et CSRF (Cross-Site Request Forgery).

En appliquant les connaissances acquises lors de ces recherches et ma formation, j'ai pu mettre en œuvre un ensemble complet de mesures de sécurité, assurant ainsi que chaque point d'entrée et interaction avec l'application soit sécurisé.

Regex is an effective text-matching tool that enables us to locate and modify strings in accordance with the precise rules we establish. Numerous computer languages and text processing tools employ regex to aid in processing tasks like extracting data, modifying sections of strings, and looking for certain patterns in text.

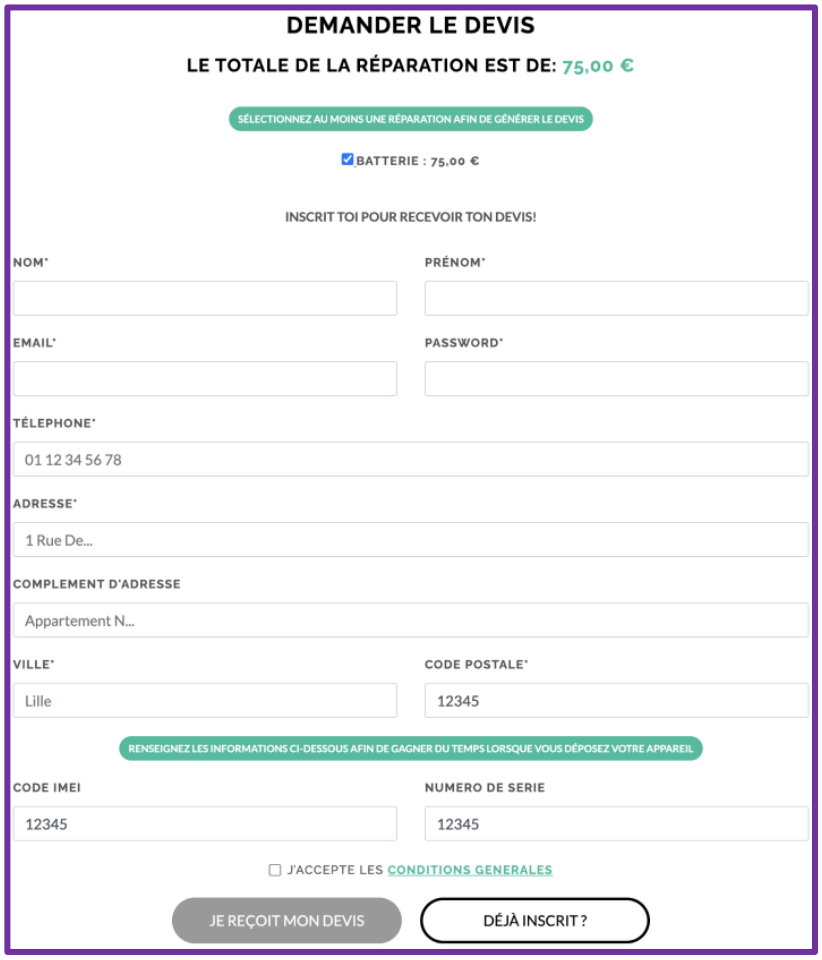
Imagine if we were looking through a lengthy paper for terms that begin with the letter “e” and end with the letter “n.” We can automate the process by using a regular expression like **e[a-z]*n** rather than manually going through each word. In this instance:



Regex that identifies the words that starts with ‘e’ and ends with ‘n’

6.2.2 En frontend

J'ai accordé une attention particulière à la validation des données utilisateur pour assurer la sécurité et la fiabilité des informations saisies. Pour cela, j'ai intégré des patterns directement dans les balises HTML **input**, en spécifiant des expressions régulières qui correspondent aux formats attendus pour les données. Chaque champ **input** dispose également d'un attribut **title** qui affiche un message d'erreur en cas de saisie non conforme, offrant ainsi un retour immédiat à l'utilisateur sur la validité de son entrée.



Les formulaires d'inscription et de connexion sont conçus pour inclure des champs spécifiques, tels que **email** et **password**, utilisant les types d'input HTML correspondants pour bénéficier des validations natives du navigateur, en plus du contrôle de saisie **pattern**. Ces formulaires sont également sécurisés grâce à l'ajout d'un jeton CSRF (**_csrf_token**), inséré via un champ caché, pour protéger contre les attaques par cross-site request forgery.

Cela garantit que chaque soumission de formulaire est authentique et provient bien de l'interface utilisateur de l'application.

```
<div class="col-md-6 start-flex bottom-space">
    <label for="inputPassword4" class="form-label">Prénom*</label>
    <input name="prenom" required type="text" class="form-control" id="inputPrenom"
        pattern="[a-zA-Z ]+"title="Le prénom doit contenir uniquement des lettres et des espaces.">
</div>
<div class="col-md-6 start-flex bottom-space">
    <label for="inputEmail4" class="form-label">Email*</label>
    <input name="email" required type="email" class="form-control" id="inputEmail_sign_in">
</div>
<div class="col-md-6 start-flex bottom-space">
    <label for="inputPassword4" class="form-label">Password*</label>
    <input name="mdp" required type="password" class="form-control" id="inputPassword_sign_in"
        pattern="(?=.*[a-zA-Z])(?=.*\d)(?=.*[\W_]).{8,}"
        title="Le mot de passe doit contenir au moins 8 caractères, incluant au moins un chiffre et un caractère spécial.">
</div>
```

Un exemple de pattern que j'ai appliqué au champ mot de passe exige que les mots de passe contiennent au moins 8 caractères, incluent un chiffre et un caractère spécial.

En JavaScript, j'ai implémenté des contrôles additionnels pour vérifier que les champs ne restent pas vides, que l'utilisateur ait accepté les conditions générales (à travers un input de type checkbox) et qu'au moins une option de réparation soit cochée. Si ces conditions ne sont pas remplies, le bouton de validation demeure désactivé.

```
// PREVIENT L'ENVOI DU FORMULAIRE SI AUCUNE PANNE N'EST SELECTIONNEE
const noLoggedSection = document.querySelector('.no-logged-in-form');
noLoggedSection.addEventListener('submit', function(e) {
    const checkboxes = noLoggedSection.querySelectorAll('input[type="checkbox"][name="tarifIds[]"]');
    const isAnyChecked = Array.from(checkboxes).some(checkbox => checkbox.checked);

    if (!isAnyChecked) {
        e.preventDefault();
        alert('Veuillez sélectionner au moins une panne.');
    }
});

// VALIDATION ET DÉSACTIVATION DU BOUTON LOG IN
const noLoggedInForm = document.querySelector('.no-logged-in-form');
const forms = noLoggedInForm.querySelectorAll('form');

forms.forEach(form => {
    function validateForm() {
        const button = form.querySelector('button[type="submit"]');

        if (form.id === 'inscriptionForm') {
            const conditionCheckBox = form.querySelector('.form-check-input'); // condition generales
            const inputsPrice = form.querySelectorAll('input[type="checkbox"].price-input'); // checkbox prix devis
            const inputs =
form.querySelectorAll('input[type="text"]:not(#inputAddress2):not(#code_imei_log_in):not(#num_serie_log_in):not(#code_imei_sign_in):not
(#num_serie_sign_in), input[type="number"], input[type="email"], input[type="password"]');

            const areInputsFilled = Array.from(inputs).every(input => input.value.trim() !== '');
            const isConditionChecked = conditionCheckBox.checked;
            const isAtLeastOneChecked = Array.from(inputsPrice).some(input => input.checked);

            button.disabled = !areInputsFilled || !isConditionChecked || !isAtLeastOneChecked;
        } else {
            const inputs = form.querySelectorAll('input[type="email"], input[type="password"]');
            const areInputsFilled = Array.from(inputs).every(input => input.value.trim() !== '');

            button.disabled = !areInputsFilled;
        }
    }

    const allInputs = form.querySelectorAll('input');
    allInputs.forEach(input => {
        input.addEventListener('change', validateForm);
        input.addEventListener('input', validateForm);
    });

    validateForm();
});
```

De plus, le total du devis est dynamiquement mis à jour en fonction des options de réparation cochées, offrant à l'utilisateur une estimation en temps réel du coût de la réparation envisagée.

```
// MAJ TOTALE
const devisForms = document.querySelectorAll('form');
devisForms.forEach(noLoggedForm => {
    let checkboxes = noLoggedForm.querySelectorAll('input[type="checkbox"][name="tarifIds[]"]');
    function updateTotal() {
        let total = 0;
        checkboxes.forEach(function(checkbox) {
            if (checkbox.checked) {
                let montantText = checkbox.nextElementSibling.textContent;
                let montant = parseFloat(montantText.match(/\d+(,\d+)?/)[0].replace(',', '.'));
                total += montant;
            }
        });
        document.getElementById('total').textContent = total.toFixed(2).replace('.', ',') + ' €';
    }
    checkboxes.forEach(function(checkbox) {
        checkbox.addEventListener('change', updateTotal);
    });
    updateTotal();
});
```

Cette approche multidimensionnelle de la validation des données, renforce la robustesse de l'application en prévenant les erreurs de saisie et en améliorant l'expérience utilisateur par des feedbacks immédiats et pertinents.

6.2.3 En Backend

Dans le backend de l'application, une distinction importante est faite entre les contrôles effectués dans **InscriptionController** et ceux dans **DevisController**, reflétant les différentes étapes du parcours utilisateur. Dans **InscriptionController**, une validation exhaustive des entrées est réalisée pour s'assurer de la conformité des données saisies.

Par exemple pour les noms, prénoms et villes, j'utilise des expressions régulières pour n'accepter que les caractères alphabétiques et les espaces, assurant ainsi que ces champs contiennent des données valides et pertinentes. Les adresses email sont vérifiées via **filter_var** avec le filtre **FILTER_VALIDATE_EMAIL**, garantissant que l'adresse saisie respecte le format standard d'un email. En cas de doublon d'email dans la base de données, une erreur est retournée pour éviter les inscriptions multiples avec la même adresse.

Contrôle fait dans InscriptionController

```php
$validationErrors = [];

if (!preg_match('/^[a-zA-Z ]+$/', $nom)) {
    $validationErrors[] = 'Le nom contient des caractères invalides.';
}

if (!preg_match('/^[a-zA-Z ]+$/', $prenom)) {
    $validationErrors[] = 'Le prénom contient des caractères invalides.';
}

if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
    $validationErrors[] = 'L\'adresse email est invalide.';
}

if ($this->utilisateurRepository->findOneBy(['email' => $email])) {
    $validationErrors[] = 'Cet email est déjà utilisé.';
}

if (!preg_match('/^(?=.*[a-zA-Z])(?=.*\d)(?=.*[\W_]).{8,}$/', $mdp)) {
    $validationErrors[] =
      'Le mot de passe doit contenir au moins 8 caractères, incluant au moins un chiffre et un caractère spécial.';
}

if (!preg_match('/^\d{10}$/', $numero)) {
    $validationErrors[] = 'Le numéro de téléphone doit contenir 10 chiffres.';
}

if (!preg_match('/^[a-zA-Z ]+$/', $nomVille)) {
    $validationErrors[] = 'La ville contient des caractères invalides.';
}

if (!preg_match('/^\d{5}$/', $cp)) {
    $validationErrors[] = 'Le code postal doit contenir 5 chiffres.';
}

if ($nom === null || $prenom === null || $email === null || $mdp === null ||
    $numero === null || $adresse === null || $nomVille === null || $cp === null) {
    $validationErrors[] = 'Tous les champs requis doivent être renseignés.';
}

if (empty($tarifIds)) {
    $validationErrors[] = 'Veuillez sélectionner au moins une panne.';
}

if (count($validationErrors) > 0) {
    foreach ($validationErrors as $error) {
        $this->addFlash('error', $error);
    }
    return $this->redirectToRoute('app_devis_redirect');
}
```

En revanche, dans **DevisController**, la validation se concentre principalement sur la sélection des options de réparation, identifiées par les **tarifIds**. Cette différence s'explique par le fait que **DevisController** intervient dans les cas où l'utilisateur effectue un log-in plutôt qu'une inscription. Le contrôle d'authentification lors du log-in est géré automatiquement par Symfony, qui fournit un cadre sécurisé pour vérifier les identifiants de l'utilisateur sans nécessiter de validation manuelle supplémentaire dans le contrôleur.

Contrôle fait dans DevisController

```php
if (empty($tarifIds)) {
    $this->addFlash('error', 'Veuillez sélectionner au moins une panne.');
return $this->redirectToRoute('app_devis');
```

Si une ou plusieurs de ces validations échouent, des messages d'erreur sont générés et renvoyés à l'utilisateur, l'invitant à corriger les informations saisies. La remontée de ces erreurs vers l'utilisateur est gérée grâce à Twig, avec un bloc spécifique qui affiche les messages d'erreur accumulés pendant la validation :

```
{% for message in app.flashes('error') %}
    <div class="alert alert-danger col-12 buttons-container">{{ message }}</div>
{% endfor %}
```

Cette approche méthodique de validation en backend joue un rôle crucial dans la maintenance d'un système fiable et sécurisé, où les données utilisateurs sont traitées avec le plus grand soin.

6.2.4 Essais

Des tests dédiés à la vérification de la sécurité ont été effectués. Ces tests, réalisés dans un environnement contrôlé, ont consisté à simuler des tentatives d'entrée de données non conformes, en utilisant notamment la désactivation des validations regex via la console de développement et la tentative de soumission forcée de formulaires.

Le processus a été rigoureusement conçu pour tester la réactivité des mécanismes de sécurité face à diverses tentatives de contournement ou d'injection de données malveillantes.

L'objectif était de s'assurer que, même en cas de manipulation directe du code client via des outils de développement, les validations côté serveur restent infaillibles et capables de rejeter toute donnée suspecte ou non conforme.

Les résultats de ces tests ont été positifs, confirmant l'efficacité des mesures de sécurité implémentées dans l'application. Chaque tentative de soumission non conforme a été correctement interceptée par les validations back-end, qui ont renvoyé des messages d'erreur appropriés, empêchant ainsi toute progression non autorisée ou potentiellement dangereuse dans l'application.

6.3 Gestion Backend creation d'un devis

La gestion backend a joué un rôle crucial en assurant la cohérence entre les interactions des utilisateurs et la logique métier de l'application. J'ai pris en charge la programmation des contrôleurs, utilisant l'ORM Doctrine pour établir une communication fluide avec la base de données.

Ce processus a impliqué la conception de requêtes sur mesure pour récupérer les informations correspondant aux choix des utilisateurs. Ces données, une fois recueillies, ont été intégrées dans les templates Twig, comme mentionné précédemment, assurant que le contenu présenté reste dynamique et à jour.

6.3.1 Sécurité

La sécurité contre les injections SQL est assurée par Doctrine. L'utilisation de la méthode `find()` pour récupérer des entités par leur identifiant en est un exemple concret. Doctrine encapsule automatiquement la valeur de l'identifiant dans une requête préparée, établissant ainsi une défense robuste contre les injections SQL. Ce processus isole les données des instructions SQL, prévenant toute exécution malveillante de code injecté, et protège l'application de vulnérabilités potentielles. Une seconde couche de sécurité est apportée par le typage strict en base de données et la définition de la longueur maximale pour chaque attribut.

6.3.2 Utilisation de Doctrine pour les Requêtes

L'utilisation de Doctrine pour interroger la base de données a été essentielle. Par exemple le contrôleur **HomepageController** gère cette interaction en récupérant tous les types d'appareils disponibles depuis la base de données. Le code associé extrait ces informations à l'aide de la méthode **findAll** du **TypeAppareilRepository**.

```php
<?php

namespace App\Controller\UserApp;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;
use App\Entity\TypeAppareil;
use Doctrine\ORM\EntityManagerInterface;

class HomepageController extends AbstractController
{
    #[Route('/homepage', name: 'app_homepage')]
    public function index(EntityManagerInterface $entityManager): Response
    {
        $typeAppareils = $entityManager->getRepository(TypeAppareil::class)->findAll();

        return $this->render('user_app/homepage.html.twig', [
            'typeAppareils' => $typeAppareils,
        ]);
    }
}
```

Cela permettra d'afficher tous les types d'appareils dans le template Twig avec :

```twig
{% for typeAppareil in typeAppareils %}
    {# affichage dynamique des appareils #}
{% endfor %}
```

Un deuxième exemple est la récupération dynamique des marques en fonction du type d'appareil sélectionné par l'utilisateur. Cette fonctionnalité implique la création de requêtes personnalisées.

```twig
{% for typeAppareil in typeAppareils %}
    <div class="col-md-4 col-sm-4 col-xl-3 col-lg-4 col-6 device_category">
        <a href="/device-brand/{{ typeAppareil.idTypeAppareil }}" class="card align-items-center">
            <div class="img inner mx-auto"
                <img class="main_img" src="{{ asset('user/images/categories/' ~ typeAppareil.imgTypeAppareil) }}"
                    alt="{{ typeAppareil.libTypeAppareil }}">
                <img class="main_hover_img" src="{{ asset('user/images/categories/hover_' ~ typeAppareil.imgTypeAppareil) }}"
                    alt="{{ typeAppareil.libTypeAppareil }}">
                <h4>{{ typeAppareil.libTypeAppareil }}</h4>
            </div>
        </a>
    </div>
{% endfor %}
```

Au clic du type d'appareil, l'utilisateur est redirigé vers le template du choix des marques, comme argument on prend **l'idTypeAppareil** afin d'effectuer une requête sur cette id et afficher les marques qui sont lié à ce **type d'appareil**. Pour ce faire j'ai créé une requête personnalisée **findMarquesByTypeAppareil** :

DeviceBrandController, qui permet l'affichage des marques dans le template **device-brand.html.twig** :

```php
<?php

namespace App\Controller\UserApp;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;
use App\Repository\TypeAppareilRepository;

class DeviceBrandController extends AbstractController
{
    #[Route('/device-brand/{idTypeAppareil}', name: 'app_device-brand')]
    public function index(TypeAppareilRepository $typeAppareilRepository, $idTypeAppareil): Response
    {
        $marques = $typeAppareilRepository->findMarquesByTypeAppareil($idTypeAppareil);

        return $this->render('user_app/device-brand.html.twig', [
            'controller_name' => 'DeviceBrandController',
            'brands' => $marques,
        ]);
    }
}
```

TypeAppareilRepository, qui permet d'effectuer la requête qui est appellé dans **DeviceBrandController** :

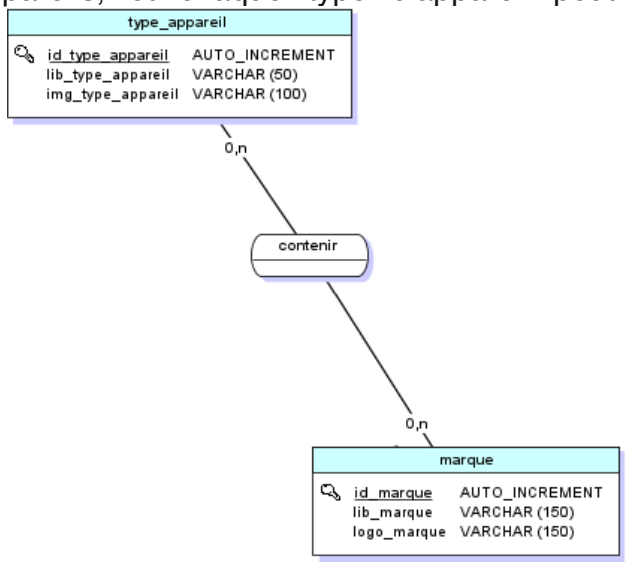
```php
public function findMarquesByTypeAppareil($idTypeAppareil)
{
    $qb = $this->getEntityManager()->createQueryBuilder();
    $qb->select('m')
        ->from('App\Entity\Marque', 'm')
        ->join('m.idTypeAppareils', 't')
        ->where('t.idTypeAppareil = :idTypeAppareil')
        ->setParameter('idTypeAppareil', $idTypeAppareil);

        return $qb->getQuery()->getResult();
}
```

Cette requête permet de chercher les **marques** qui ont dans la même ligne **l'idTypeAppareil** sélectionné par l'utilisateur, dans la table **contenir :**

	id_type_appareil	id_marque
Éditer Copier Supprimer	1	1
Éditer Copier Supprimer	1	2
Éditer Copier Supprimer	1	3

La liaison pour la table **contenir** est établie grâce à Doctrine en utilisant une association **ManyToMany** entre les entités **Marque** et **TypeAppareil**. Cela signifie que chaque marque peut être associée à plusieurs types d'appareils, et chaque type d'appareil peut correspondre à plusieurs marques.

Dans l'ORM, cette relation est gérée par une table de jointure qui possède une clé primaire composite, composée de la clé étrangère de la table **marque** et de la clé étrangère de la table **type_appareil**. Cette structure de données permet une flexibilité dans la gestion des relations et facilite les requêtes croisées entre ces deux entités essentielles de l'application.




```php
/**
* @var \Doctrine\Common\Collections\Collection
*
* @ORM\ManyToMany(targetEntity="TypeAppareil", inversedBy="idMarque")
* @ORM\JoinTable(name="contenir",
*   joinColumns={
*  @ORM\JoinColumn(name="id_marque", referencedColumnName="id_marque")
*   },
*   inverseJoinColumns={
*  @ORM\JoinColumn(name="id_type_appareil", referencedColumnName="id_type_appareil")
*   }
* )
*/
private $idTypeAppareils = array();


/**
* @return Collection<int, TypeAppareil>
*/
public function getidTypeAppareils(): Collection
{
    return $this->idTypeAppareils;
}

public function addidTypeAppareil(TypeAppareil $idTypeAppareil): static
{
    if (!$this->idTypeAppareils->contains($idTypeAppareil)) {
        $this->idTypeAppareils->add($idTypeAppareil);
    }
    return $this;
}

public function removeidTypeAppareil(TypeAppareil $idTypeAppareil): static
{
    $this->idTypeAppareils->removeElement($idTypeAppareil);
    return $this;
}
```

6.3.3 Sélection des tarifs

La fonctionnalité de sélection des tarifs représente un aspect clé de l'interaction utilisateur, permettant une estimation personnalisée des coûts de réparation. Ce processus commence dans le template **device-repair.html.twig**, où j'ai implémenté une boucle for pour afficher dynamiquement chaque tarif disponible.

```twig
{% for tarif in tarifs %}
    <label for="{{ tarif.idTarif }}" id="{{ tarif.idPanne.libPanne }}"
      class="radios font-body ls0 nott rounded-0 radio_btn d-flex align-items-center btn btn-outline-dark  image_have"
      data-input-type="radio">
        <div class="inner">
            <div class="images">
                <img src="{{ asset('user/images/' ~ tarif.idPanne.imgPanne) }}" alt="{{ tarif.idPanne.libPanne }}"
                    width="30px" id="{{ tarif.idPanne.libPanne }}">
            </div>
            <input type="checkbox" name="tarifIds[]" id="{{ tarif.idTarif }}" value="{{ tarif.idTarif }}"
                autocomplete="off" data-default="0">
            <h5 class="mb-0">{{ tarif.idPanne.libPanne }}</h5>
            <div class="item_price text-primary h5 mb-0">
                <strong>{{ tarif.montant|number_format(2, ',', ' ') }}
                    €</strong>
            </div>
        </div>
    </label>
{% endfor %}
```

Les utilisateurs peuvent sélectionner les réparations souhaitées via des inputs de type checkbox, dont les identifiants (**idTarif**) sont accumulés dans le tableau **tarifIds[]** pour traitement ultérieur.

Dans **DevisController**, les identifiants de tarif sont récupérés à partir de la requête POST. Une validation est effectuée pour s'assurer de la légitimité des identifiants, puis une requête est lancée pour chaque **idTarif** valide afin d'extraire les détails tarifaires correspondants de la base de données. Ces informations sont ensuite stockées dans la session et passées au template **devis.html.twig** pour affichage.

```php
#[Route('/devis', name: 'app_devis', methods: ['POST'])]
public function showDevisForm(Request $request, EntityManagerInterface $entityManager, SessionInterface $session):
Response {
    $postData = $request->request->all();
    $tarifIds = $postData['tarifIds'] ?? [];

    $session->set('tarifIds', $tarifIds);

    $tarifs = [];
    foreach ($tarifIds as $tarifId) {
        if (preg_match('/^\d{1,2}$/', $tarifId)) {
            $tarif = $entityManager->getRepository(Tarif::class)->find($tarifId);
            if ($tarif) {
                $tarifs[] = $tarif;
            }
        } else {
            throw new \InvalidArgumentException('Identifiant de tarif invalide.');
        }
    }

    return $this->render('user_app/devis.html.twig', [
        'tarifs' => $tarifs,
    ]);
}
```

Le template **devis.html.twig** utilise ensuite ces données pour afficher dynamiquement la liste des pannes sélectionnées et leurs coûts associés, permettant aux utilisateurs de visualiser le résumé de leur devis.

Chaque tarif sélectionné est marqué comme checked, assurant que les choix de l'utilisateur sont conservés et présentés clairement, l'utilisateur pourra aussi décider de désélectionner une réparation s'il a changé d'avis, tant que au moins une réparation est checked pour la création du devis.

```
{% for tarif in tarifs %}
    <li class="liste-des-pannes">
        <input type="checkbox" name="tarifIds[]" id="tarif{{ tarif.idTarif }}" value="{{ tarif.idTarif }}" checked>
        <label for="tarif{{ tarif.idTarif }}">
            {{ tarif.idPanne.libPanne }} : {{ tarif.montant|number_format(2, ',', ' ') }} €
        </label>
    </li>
{% endfor %
```

Cette implémentation démontre l'intégration fluide entre le frontend et le backend, où le choix des utilisateurs influence directement la génération du devis, offrant une expérience utilisateur personnalisée et précise quant au coût des réparations envisagées.

6.3.4 Instanciation du devis

Dans le contrôleur **DevisController**, une fonction est dédiée à l'instantiation des devis. Ce processus démarre dès que l'utilisateur soumet ses choix de réparations. Le système, via le code dans **DevisController**, gère d'abord la validation pour s'assurer qu'au moins une panne est sélectionnée. Si cette condition n'est pas remplie, un message d'erreur est affiché et l'utilisateur est redirigé pour effectuer une sélection valide (comme on as vu auparavant).

Une fois les sélections validées, le contrôleur procède à la création d'un nouveau numéro de devis, qui est généré en incrémentant le dernier numéro existant dans la base de données.

Chaque tarif identifié par son identifiant unique dans le tableau **tarifIds**, grâce à cela je peux trouver avec le model lié à un **tarif** en utilisent **$tarif->getModele()**. En aient les infos : **$user** , **$modele**, **$codeImei** et **$numSerie** (ses deux dernier attributs sont réentrée par l'utilisateur l'lorsque de la demande du devis, ces informations ne sont pas obligatoires pour une demande du devis), je peux vérifier si dans la table appareil existe un appareil avec ces infos, dans le cas contraire, j'instancie un nouveau appareil dans **AppareilRepository.**

Pour chaque **tarif**, une nouvelle **reparation** est instanciée dans **ReparationRepository** et liée à l'appareil concerné. Toutes les informations pertinentes sont capturées, y compris les observations liées à la panne et les dates de la demande de réparation.

Enfin, un **devis** est instancié dans **DevisRepository**, regroupant toutes les informations : le numéro de devis unique, le prix total TTC basé sur le **tarif** de la panne. Le devis est également doté d'une date prévue de restitution pour l'appareil et d'un statut initial indiquant que le devis n'a pas « été encore accepté.

Cette séquence d'actions est soigneusement exécutée dans une **transaction**, garantissant que toutes les modifications sont enregistrées de manière atomique dans la base de données. Ainsi, si toutes les étapes sont validées sans erreurs, les données sont persistées, et l'utilisateur est ensuite redirigé vers son tableau de bord où il peut voir son nouveau devis, concluant ainsi une interaction fluide et sécurisée avec l'application.

DevisController.php

```php
<?php

#[Route('/create-devis', name: 'app_create_devis', methods: ['POST'])]
public function createDevis(
    Request $request, EntityManagerInterface $entityManager, AppareilRepository $appareilRepository,
    ReparationRepository $reparationRepository, DevisRepository $devisRepository): Response
{
    $entityManager->getConnection()->beginTransaction();
    try {
        $user = $this->getUser();

        $codeImei = $request->request->get('codeImei', 'AJOUTER IMEI PAR ADMIN'); //regex
        $numSerie = $request->request->get('numSerie', 'AJOUTER NUMERO SERIE PAR ADMIN');

        $postData = $request->request->all();
        $tarifIds = $postData['tarifIds'] ?? [];

        if (empty($tarifIds)) {
            $this->addFlash('error', 'Veuillez sélectionner au moins une panne.');
            $entityManager->getConnection()->rollBack();
            return $this->redirectToRoute('app_devis');
        }

        $lastDevis = $entityManager->getRepository(Devis::class)->findOneBy([], ['idDevis' => 'DESC']);
        $numDevis = $lastDevis ? 'DEV' . str_pad((substr($lastDevis->getNumDevis(), 3) + 1), 3, '0', STR_PAD_LEFT) : 'DEV001';

        $tarif = $entityManager->getRepository(Tarif::class)->find($tarifIds[0]);
        $modele = $tarif->getIdModele();

        $appareil = $appareilRepository->findOneBy([
            'idUtilisateur' => $user,
            'idModele' => $modele,
            'codeImei' => $codeImei,
            'numSerie' => $numSerie,
        ]);

        if (!$appareil) {
            $appareil = $appareilRepository->createAppareil($user, $modele, $codeImei, $numSerie);
        }

        foreach ($tarifIds as $tarifId) {
            $tarif = $entityManager->getRepository(Tarif::class)->find($tarifId);

            if ($tarif && $appareil) {
                $reparation = $reparationRepository->createReparation($tarif, $appareil, $user);
                $devis = $devisRepository->createDevis($numDevis, $tarif, $reparation, $user);
            }
        }

        $entityManager->flush();
        $entityManager->getConnection()->commit();
        return $this->redirectToRoute('app_user-dashboard');

    } catch (\Exception $e) {
        $entityManager->getConnection()->rollBack();
        throw $e;
    }
}
```

AppareilRepository.php

```php
public function createAppareil($user, $modele, $codeImei, $numSerie)
{
    $appareil = new Appareil();
    $appareil->setCodeImei($codeImei)
        ->setNumSerie($numSerie)
        ->setDateCreationAppareil(new \DateTime())
        ->setIdUtilisateur($user)
        ->setIdModele($modele);

        $typeAppareil = $this->findOneBy(['idModele' => $modele]);
        if ($typeAppareil) {
        $appareil->setIdTypeAppareil($typeAppareil->getIdTypeAppareil());
    }

    $this->_em->persist($appareil);

    return $appareil;
}
```

ReparationRepository.php

```php
public function createReparation($tarif, $appareil, $user)
{
    $reparation = new Reparation();
    $reparation->setObservation($tarif->getIdPanne()->getLibPanne())
        ->setDateDemande(new \DateTime())
        ->setDateMajDemande(new \DateTime())
        ->setIdPanne($tarif->getIdPanne())
        ->setIdAppareil($appareil)
        ->setIdUtilisateur($user);

    $this->_em->persist($reparation);

    return $reparation;
}
```

DevisRepository.php

```php
public function createDevis($numDevis, $tarif, $reparation, $user)
{
    $devis = new Devis();
    $devis->setNumDevis($numDevis)
        ->setDateDevis(new \DateTime())
        ->setPrixTtc($tarif->getMontant())
        ->setCommentaireDevis($tarif->getIdPanne()->getLibPanne())
        ->setDateRestitution(new \DateTime('+1 week'))
        ->setStatut(0)
        ->setDateMajDevis(new \DateTime())
        ->setIdReparation($reparation)
        ->setIdUtilisateur($user);

    $this->_em->persist($devis);

    return $devis;
}
```

6.4 Retours et Améliorations

Lors de la démonstration finale du troisième sprint, nous avons reçu des retours de la part du client, suggérant des pistes d'amélioration pour l'application. En réponse à ces suggestions, plusieurs modifications ont été envisagées pour renforcer la sécurité et améliorer l'expérience utilisateur.

Premièrement, l'intégration d'un captcha a été prévue pour renforcer la sécurité lors des processus d'inscription et de soumission de formulaires de contacte, dans le but de limiter les risques de soumissions automatisées et malveillantes.

Deuxièmement, une refonte visuelle du dashboard utilisateur est envisagée pour améliorer sa clarté et son ergonomie. Cette refonte impliquera des modifications dans la disposition de certains éléments, avec l'objectif de rendre l'interface plus intuitive et accessible, facilitant ainsi la navigation et l'utilisation des fonctionnalités par les utilisateurs.

Troisièmement, une autre amélioration importante à considérer est l'ajout d'un pop-up demandant à l'utilisateur d'accepter l'utilisation des cookies. Cette mesure s'aligne avec les réglementations en matière de confidentialité des données, telles que le RGPD, et permet de garantir une expérience utilisateur transparente et conforme aux normes de protection de la vie privée. En intégrant cette fonctionnalité, l'application renforce sa conformité réglementaire tout en démontrant un engagement envers la protection des données personnelles de ses utilisateurs.

Enfin, l'envoi d'un email de confirmation lors de l'inscription est à mettre en place comme une mesure supplémentaire de vérification et d'engagement. Cette démarche vise non seulement à confirmer l'adresse email saisie lors de l'inscription, mais aussi à offrir une expérience utilisateur plus sécurisée et personnalisée.

7 Conclusion

7.1 Synthèse des accomplissements

Le développement a été caractérisé par une succession de réussites, renforçant non seulement la sécurité et l'efficacité de l'application, mais améliorant également l'expérience utilisateur et mes compétences. Ces efforts ont abouti à la création d'une plateforme robuste, intuitive et sécurisée, apte à répondre aux divers besoins des utilisateurs concernant les devis pour réparations électroniques.

Mon user story sur la création de devis a détaillé le parcours utilisateur depuis la sélection du type d'appareil jusqu'à l'obtention d'un devis personnalisé dans son espace client. Ce processus a soulevé des défis techniques importants, notamment l'intégration d'un système de validation avancé à la fois en frontend et en backend, ainsi que le développement d'une interface utilisateur dynamique qui réagit en temps réel aux choix de l'utilisateur.

Les principales contraintes et difficultés rencontrées ont été de concilier une expérience utilisateur fluide avec les nécessités de sécurité. Ces défis se sont toutefois transformés en opportunités d'apprentissage et d'évolution, me permettant d'affiner mes compétences et d'apporter des améliorations significatives à l'application.

7.2 Perspectives pour le projet

Malgré les progrès accomplis, le chemin vers la finalisation et le déploiement de l'application DevFix est encore long. Actuellement, le projet n'a pas atteint sa complétude et n'a pas été déployé. Parmi les évolutions prévues, l'introduction d'une fonction permettant de télécharger les devis en format PDF est prioritaire. Cette amélioration vise à enrichir l'expérience utilisateur tout en s'assurant que les documents générés respectent les normes officielles et professionnelles.

En plus de cette capacité, nous envisageons également de convertir les devis en factures, offrant ainsi une fluidité optimale du processus de gestion des commandes pour les utilisateurs et les administrateurs du système. Cette fonctionnalité permettra de simplifier la transition d'une estimation à une transaction conclue, soulignant notre engagement à fournir des solutions complètes et intégrées.

8 Annexes

8.1 Annexe 1 :

Script de création SQL

```
DROP DATABASE IF EXISTS `devfix`;
CREATE DATABASE `devfix`;

use `devfix`;

CREATE TABLE IF NOT EXISTS type_panne(
        id_panne  Int  Auto_increment  NOT NULL ,
        lib_panne Varchar (150) NOT NULL,
        img_panne   Varchar (40)
    ,CONSTRAINT type_panne_PK PRIMARY KEY (id_panne)
)ENGINE=InnoDB;

CREATE TABLE type_appareil(
        id_type_appareil  Int  Auto_increment  NOT NULL ,
        lib_type_appareil Varchar (50) NOT NULL ,
        img_type_appareil Varchar (100)
    ,CONSTRAINT type_appareil_PK PRIMARY KEY (id_type_appareil)
)ENGINE=InnoDB;

CREATE TABLE marque(
        id_marque   Int  Auto_increment  NOT NULL ,
        lib_marque  Varchar (150) NOT NULL ,
        logo_marque Varchar (150)
    ,CONSTRAINT marque_PK PRIMARY KEY (id_marque)
)ENGINE=InnoDB;

CREATE TABLE contenir(
        id_type_appareil  Int NOT NULL ,
        id_marque Int NOT NULL
    ,CONSTRAINT contenir_PK PRIMARY KEY (id_type_appareil,id_marque)
    ,CONSTRAINT contenir_type_appareil_FK FOREIGN KEY (id_type_appareil)
REFERENCES type_appareil(id_type_appareil)
    ,CONSTRAINT contenir_marque0_FK FOREIGN KEY (id_marque) REFERENCES
marque(id_marque)
)ENGINE=InnoDB;

CREATE TABLE modele(
        id_modele  Int  Auto_increment  NOT NULL ,
        lib_modele Varchar (150) NOT NULL ,
        img_modele Varchar (40) ,
        id_marque  Int NOT NULL
    ,CONSTRAINT modele_PK PRIMARY KEY (id_modele)
    ,CONSTRAINT modele_marque_FK FOREIGN KEY (id_marque) REFERENCES
marque(id_marque)
)ENGINE=InnoDB;

CREATE TABLE pieces(
        id_pieces              Int  Auto_increment  NOT NULL ,
        lib_pieces             Varchar (150) NOT NULL ,
        ref_fabricant          Varchar (150) NOT NULL ,
        stock                  Int NOT NULL ,
        img_piece              Varchar (50) NOT NULL ,
        delai_livraison_pieces Int NOT NULL ,
        prix_pieces_ttc        DECIMAL (15,3)  NOT NULL ,
        date_creation_pieces   Datetime NOT NULL ,
        date_maj_pieces        Datetime NOT NULL ,
        id_modele              Int NOT NULL
    ,CONSTRAINT pieces_PK PRIMARY KEY (id_pieces)
    ,CONSTRAINT pieces_modele_FK FOREIGN KEY (id_modele) REFERENCES
modele(id_modele)
)ENGINE=InnoDB;

CREATE TABLE role(
        id_role     Int  Auto_increment  NOT NULL ,
        code_role   Varchar (50) NOT NULL ,
        libelle_role Varchar (50) NOT NULL
    ,CONSTRAINT role_PK PRIMARY KEY (id_role)
)ENGINE=InnoDB;
```

```
CREATE TABLE notes(
        id_notes           Int  Auto_increment  NOT NULL ,
        titre_notes        Varchar (100) NOT NULL ,
        commentaires       Text NOT NULL ,
        date_creation_note Datetime NOT NULL ,
        date_maj_note      Datetime NOT NULL ,
        id_modele          Int NOT NULL
    ,CONSTRAINT notes_PK PRIMARY KEY (id_notes)
    ,CONSTRAINT notes_modele_FK FOREIGN KEY (id_modele) REFERENCES
modele(id_modele)
)ENGINE=InnoDB;

CREATE TABLE ville(
        id_ville    Int  Auto_increment  NOT NULL ,
        nom_ville   Varchar (50) NOT NULL ,
        code_insee  Char (5) NOT NULL ,
        code_postal Char (5) NOT NULL
    ,CONSTRAINT ville_PK PRIMARY KEY (id_ville)
)ENGINE=InnoDB;

CREATE TABLE utilisateur(
        id_utilisateur            Int  Auto_increment  NOT NULL ,
        nom                       Varchar (100) NOT NULL ,
        prenom                    Varchar (100) NOT NULL ,
        email                     Varchar (150) NOT NULL ,
        mdp                       Varchar (150) NOT NULL ,
        numero                    Varchar (20) NOT NULL ,
        adresse                   Varchar (255) ,
        complement_adresse        Varchar (50) ,
        date_creation_utilisateur Datetime NOT NULL ,
        date_maj_utilisateur      Datetime NOT NULL ,
        id_role                   Int NOT NULL ,
        id_ville                  Int NOT NULL
    ,CONSTRAINT utilisateur_PK PRIMARY KEY (id_utilisateur)
    ,CONSTRAINT utilisateur_role_FK FOREIGN KEY (id_role) REFERENCES role(id_role)
    ,CONSTRAINT utilisateur_ville0_FK FOREIGN KEY (id_ville) REFERENCES
ville(id_ville)
)ENGINE=InnoDB;

CREATE TABLE appareil(
        id_appareil           Int  Auto_increment  NOT NULL ,
        code_imei             Varchar (100) ,
        num_serie             Varchar (100) NOT NULL ,
        date_creation_appareil Datetime NOT NULL ,
        id_utilisateur        Int NOT NULL ,
        id_modele             Int NOT NULL ,
        id_type_appareil      Int NOT NULL
    ,CONSTRAINT appareil_PK PRIMARY KEY (id_appareil)
    ,CONSTRAINT appareil_utilisateur_FK FOREIGN KEY (id_utilisateur) REFERENCES
utilisateur(id_utilisateur)
    ,CONSTRAINT appareil_modele0_FK FOREIGN KEY (id_modele) REFERENCES
modele(id_modele)
    ,CONSTRAINT appareil_type_appareil1_FK FOREIGN KEY (id_type_appareil)
REFERENCES type_appareil(id_type_appareil)
)ENGINE=InnoDB;

CREATE TABLE reparation(
        id_reparation    Int  Auto_increment  NOT NULL ,
        observation      Text NOT NULL ,
        date_demande     Datetime NOT NULL ,
        date_maj_demande Datetime NOT NULL ,
        id_panne         Int NOT NULL ,
        id_appareil      Int NOT NULL ,
        id_utilisateur   Int NOT NULL
    ,CONSTRAINT reparation_PK PRIMARY KEY (id_reparation)
    ,CONSTRAINT reparation_type_panne_FK FOREIGN KEY (id_panne) REFERENCES
type_panne(id_panne)
    ,CONSTRAINT reparation_Appareil0_FK FOREIGN KEY (id_appareil) REFERENCES
appareil(id_appareil)
    ,CONSTRAINT reparation_utilisateur2_FK FOREIGN KEY (id_utilisateur) REFERENCES
utilisateur(id_utilisateur)
)ENGINE=InnoDB;
```

```
CREATE TABLE tarif(
        id_tarif       Int  Auto_increment  NOT NULL ,
        montant        DECIMAL (15,3)  NOT NULL ,
        date_creation  Datetime NOT NULL ,
        date_maj_tarif Datetime NOT NULL ,
        id_panne       Int NOT NULL ,
        id_modele      Int NOT NULL
    ,CONSTRAINT tarif_PK PRIMARY KEY (id_tarif)
    ,CONSTRAINT tarif_type_panne_FK FOREIGN KEY (id_panne) REFERENCES
type_panne(id_panne)
    ,CONSTRAINT tarif_modele0_FK FOREIGN KEY (id_modele) REFERENCES
modele(id_modele)
)ENGINE=InnoDB;

CREATE TABLE devis(
        id_devis          Int  Auto_increment  NOT NULL ,
        num_devis         Varchar (50) NOT NULL ,
        date_devis        Datetime NOT NULL ,
        prix_ttc          DECIMAL (15,3)  NOT NULL ,
        commentaire_devis Text ,
        date_restitution  Date NOT NULL ,
        statut        Bool NOT NULL ,
        date_maj_devis    Datetime NOT NULL ,
        id_reparation     Int NOT NULL ,
        id_utilisateur    Int NOT NULL
    ,CONSTRAINT devis_PK PRIMARY KEY (id_devis)
    ,CONSTRAINT devis_reparation_FK FOREIGN KEY (id_reparation) REFERENCES
reparation(id_reparation)
    ,CONSTRAINT devis_utilisateur0_FK FOREIGN KEY (id_utilisateur) REFERENCES
utilisateur(id_utilisateur)
)ENGINE=InnoDB;

CREATE TABLE alerte(
        id_alerte      Int  Auto_increment  NOT NULL ,
        titre_alerte   Varchar (100) NOT NULL ,
        contenu_alerte Text NOT NULL ,
        date_alerte    Datetime NOT NULL ,
        id_utilisateur Int NOT NULL ,
        id_devis       Int NOT NULL
    ,CONSTRAINT alerte_PK PRIMARY KEY (id_alerte)
    ,CONSTRAINT alerte_utilisateur_FK FOREIGN KEY (id_utilisateur) REFERENCES
utilisateur(id_utilisateur)
    ,CONSTRAINT alerte_devis0_FK FOREIGN KEY (id_devis) REFERENCES devis(id_devis)
)ENGINE=InnoDB;

CREATE TABLE parametre(
    id_parametre            INT AUTO_INCREMENT PRIMARY KEY,
    nom_entreprise          VARCHAR(50) NOT NULL,
    siren                   VARCHAR(50) NOT NULL,
    code_tva                VARCHAR(50) NOT NULL,
    adresse_entreprise      VARCHAR(50) NOT NULL,
    email_entreprise        VARCHAR(50) NOT NULL,
    telephone_entreprise    VARCHAR(50) NOT NULL,
    couleur_primaire_entreprise VARCHAR(30) NOT NULL,
    logo_entreprise         VARCHAR(30) NOT NULL
) ENGINE=InnoDB;

CREATE TABLE composer(
        id_devis  Int NOT NULL ,
        id_pieces Int NOT NULL
    ,CONSTRAINT composer_PK PRIMARY KEY (id_devis,id_pieces)
    ,CONSTRAINT composer_devis_FK FOREIGN KEY (id_devis) REFERENCES
devis(id_devis)
    ,CONSTRAINT composer_pieces0_FK FOREIGN KEY (id_pieces) REFERENCES
pieces(id_pieces)
)ENGINE=InnoDB;
```

8.2 Annexe 2

Script d'initialisation SQL

```sql
INSERT INTO type_panne (lib_panne, img_panne) VALUES
('Écran cassé', 'ecran.png'),
('Dommages par l''eau', 'dommages_eau.png'),
('Batterie', 'batterie.png'),
('Problème de charge', 'chargement.png'),
('Déblocage / Logiciel', 'deblocage.png'),
('Caméra défectueuse', 'camera.png'),
('Dommages à l''arriere', 'arriere.png');


INSERT INTO role (code_role, libelle_role) VALUES
('admin', 'Administrateur'),
('technicien', 'Technicien'),
('client', 'Client');

INSERT INTO type_appareil (lib_type_appareil, img_type_appareil) VALUES
('Smartphone', 'smartphone.png'),
('Tablette', 'tablette.png'),
('smartwatch', 'smartwatch.png'),
('Ordinateur', 'ordinateur.png'),
('Console', 'console.png');

INSERT INTO marque (lib_marque, logo_marque) VALUES
('Samsung', 'samsung.png'),
('Apple', 'apple.png'),
('Sony', 'sony.png');

INSERT INTO contenir (id_type_appareil, id_marque) VALUES
(1, 1),
(1, 2),
(1, 3);

INSERT INTO modele (lib_modele, img_modele, id_marque) VALUES
('Samsung Galaxy S10', 'galaxy-s10.png', 1),
('Samsung Galaxy S20 Ultra 5G', 'galaxy-s20-ultra-5g.png', 1),
('iPhone 11 Pro', 'iphone-11-pro.png', 2),
('iPhone XS', 'iphone-xs.png', 2);

INSERT INTO pieces (lib_pieces, ref_fabricant, stock, img_piece,
delai_livraison_pieces, prix_pieces_ttc, date_creation_pieces, date_maj_pieces,
id_modele) VALUES
('Batterie', 'BAT123', 100, 'batterie.png', 7, 29.99, NOW(), NOW(), 1),
('Écran', 'SCR456', 50, 'ecran.png', 5, 99.99, NOW(), NOW(), 2),
('Caméra', 'CAM789', 30, 'camera.png', 3, 49.99, NOW(), NOW(), 3);

INSERT INTO notes (titre_notes, commentaires, date_creation_note, date_maj_note,
id_modele) VALUES
('Remarques sur le Galaxy S20', 'Très bon état général', NOW(), NOW(), 1),
('Problème de charge sur l''iPhone 12', 'Nécessite un remplacement de la
batterie', NOW(), NOW(), 2),
('Écran cassé sur le Xperia 5 II', 'Demande de devis pour réparation', NOW(),
NOW(), 3);

INSERT INTO ville (nom_ville, code_insee, code_postal) VALUES
('Paris', '75000', '75000'),
('Lyon', '69000', '69000'),
('Marseille', '13000', '13000');
```

```sql
INSERT INTO utilisateur (nom, prenom, email, mdp, numero, adresse,
complement_adresse, date_creation_utilisateur, date_maj_utilisateur, id_role,
id_ville) VALUES
('Dupont', 'Jean', 'jean.dupont@example.com',
'$argon2i$v=19$m=65536,t=4,p=1$Q3Z2Z2dmbHdCVXRFV1Jvdw$2Y2gWQ1y2vszOUhumFQKJgAlcciD
2V2hmnHvMmomDRg', '0123456789', '1 Rue de la Paix', NULL, NOW(), NOW(), 1, 1),
('Doe', 'Jane', 'jane.doe@example.com',
'$argon2i$v=19$m=65536,t=4,p=1$Q3Z2Z2dmbHdCVXRFV1Jvdw$2Y2gWQ1y2vszOUhumFQKJgAlcciD
2V2hmnHvMmomDRg', '0987654321', '15 Avenue des Lilas', 'Appartement 3B', NOW(),
NOW(), 3, 2),
('Smith', 'John', 'john.smith@example.com',
'$argon2i$v=19$m=65536,t=4,p=1$Q3Z2Z2dmbHdCVXRFV1Jvdw$2Y2gWQ1y2vszOUhumFQKJgAlcciD
2V2hmnHvMmomDRg', '1234567890', '25 Boulevard Voltaire', NULL, NOW(), NOW(), 2,
3);

INSERT INTO appareil (code_imei, num_serie, date_creation_appareil,
id_utilisateur, id_modele, id_type_appareil) VALUES
('123456789012345', 'ABCD1234', NOW(), 1, 1, 1),
('987654321098765', 'EFGH5678', NOW(), 2, 2, 1),
('456789012345678', 'IJKL9101', NOW(), 3, 3, 1);

-- Insertion de données factices dans la table reparation
INSERT INTO reparation (observation, date_demande, date_maj_demande, id_panne,
id_appareil, id_utilisateur) VALUES
('Écran cassé', NOW(), NOW(), 1, 3, 2),
('Dommages par l''eau', NOW(), NOW(), 2, 3, 2),
('Batterie', NOW(), NOW(), 3, 3, 3);

INSERT INTO tarif (montant, date_creation, date_maj_tarif, id_panne, id_modele)
VALUES
(100.00, NOW(), NOW(), 1, 3),
(150.00, NOW(), NOW(), 2, 3),
(75.00, NOW(), NOW(), 3, 3),
(80.00, NOW(), NOW(), 4, 3),
(100.00, NOW(), NOW(), 5, 3),
(100.00, NOW(), NOW(), 6, 3),
(50.00, NOW(), NOW(), 7, 3);

INSERT INTO devis (num_devis, date_devis, prix_ttc, commentaire_devis,
date_restitution, statut, date_maj_devis, id_reparation, id_utilisateur) VALUES
('DEV001', NOW(), 150.00, 'Remplacement de l''écran nécessaire', '2024-02-15', 0,
NOW(), 1, 2),
('DEV002', NOW(), 80.00, 'Remplacement de l''écran arriere', '2024-02-15', 0,
NOW(), 1, 2),
('DEV003', NOW(), 75.00, 'Remplacement de la batterie', '2024-02-20', 0, NOW(), 2,
3),
('DEV004', NOW(), 100.00, 'Réparation de la caméra', '2024-02-25', 0, NOW(), 3,
1);

INSERT INTO alerte (titre_alerte, contenu_alerte, date_alerte, id_utilisateur,
id_devis) VALUES
('Alerte batterie faible', 'La batterie doit être remplacée', NOW(), 2, 1),
('Alerte devis expiré', 'Le devis DEV002 doit être validé avant le 20 février',
NOW(), 3, 2),
('Alerte réparation terminée', 'La réparation du Xperia 5 II est terminée', NOW(),
1, 3);

INSERT INTO parametre (nom_entreprise, siren, code_tva, adresse_entreprise,
email_entreprise, telephone_entreprise, couleur_primaire_entreprise,
logo_entreprise) VALUES
('DevFix', '978789956', 'FR42978789956', '203 Rue Pierre de Roubaix, 59100
Roubaix', 'contact@devfix.com', '0979301018', '#336699', 'logo.png');

INSERT INTO composer (id_devis, id_pieces) VALUES
(1, 2),
(2, 1),
(3, 3);
```

8.3 Annexe 3

```
<div id="wrapper" class="clearfix window">
    <header id="header" class="transparent_header_bg">
        <div id="header-wrap">
            <div id="primary-menu-trigger">
                <i class="icon-reorder"></i>
            </div>
            <div class="container clearfix container-header-width">
                <div class="row header_full_width">
                    <div
                        id="top-header-logo" class="col-9 col-md-3">
                        <!-- Logo -->
                        <div id="logo">
                            <a href="{{ url('app_homepage') }}" class="header_logo">
                                <img src="{{ asset('user/images/' ~ parametre.logoEntreprise) }}"
                                    alt="{{ parametre.nomEntreprise }}" width="" height="100">
                            </a>
                            <a hhref="homepage" class="sticky_logo">
                                <img src="{{ asset('user/images/' ~ parametre.logoEntreprise) }}"
                                    alt="{{ parametre.nomEntreprise }}" width="" height="80">
                            </a>
                        </div>
                    </div>

                    <nav id="primary-menu">
                        <ul class="main_menu_section">
                            <li class="shop-li">
                                <a href="https://shopify.com" class="shop header_link">
                                    <div>Boutique en ligne</div>
                                </a>
                            </li>
                            <li class="">
                                <a href="{{ url('app_contact') }}" class="contact-link header_link">
                                    <div>Contact</div>
                                </a>
                            </li>
                        </ul>
                    </nav>
                </div>
            </div>
        </div>
    </header>

    {% block body %}
        {# Le bloc qui affiche le 'main' autres pages #}
    {% endblock %}

    <footer id="footer">
        <div class="container">
            <div class="footer-widgets-wrap pb-3 clearfix">
                <div class="widget widget_links app_landing_widget_link clearfix">
                    <h4>Legales</h4>
                    <ul>
                        <li><a class=""href="{{ url('app_legal-mentions') }}">Mentions légales</a></li>
                        <li><a class=""href="{{ url('app_privacy-policy') }}">Politique de confidentialité</a></li>
                        <li><a class=""href="{{ url('app_terms-and-conditions') }}">Conditions générales</a></li>
                        <li><a class=""href="{{ url('app_cookies-policy') }}">Politique relative aux cookies</a></li>
                    </ul>
                </div>
                <div class="widget widget_links app_landing_widget_link clearfix">
                    <h4>DevFIx</h4>
                    <ul>
                        <li><a class=""href="{{ url('app_contact') }}">Contact Nous</a></li>
                        <li><a class=""href="{{ url('app_services') }}">Services</a></li>
                    </ul>
                </div>
                <div class="widget widget_links app_landing_widget_link clearfix">
                    <h4>Social</h4>
                    <div class="footer-social-list">
                        <a href="https://facebook.com" target="_blank" class="global-icon-container">
                            <i class="fa fa-facebook" aria-hidden="true"></i>
                        </a>
                        <a href="https://instagram.com" target="_blank" class="global-icon-container">
                            <i class="fa fa-instagram" aria-hidden="true"></i>`
                        </a>
                        <a href="https://snapchat.com" target="_blank" class="global-icon-container">
                            <i class="fa fa-snapchat-ghost" aria-hidden="true"></i>
                        </a>
                    </div>
                </div>
            </div>
        </div>

        <div id="copyrights" class="nobg border-top footer_bottom">
            <div class="container clearfix">
                <div class="row">
                    <div class="col-md-6">
                        ©
                        {{ "now"|date("Y") }}
                        {{ parametre.nomEntreprise }}. Tous droits réservés.
                    </div>
                </div>
            </div>
        </div>
    </footer>
</div>
```