HEIG VD HAUTE ÉCOLE D'INGÉNIERIE ET DE GESTION DU CANTON DE VAUD

AutoGestion

PDG Projet 2024-2025

26.08.2024

Vitória Oliveira **Camille Koestli** **Guilherme Pinto**

Alexandre Shyshmarov

Table des matières

1. Description du projet

Le projet consiste à développer une application Android dédiée à la gestion des automobiles dans un garage. L'application permet de scanner les plaques d'immatriculation des véhicules afin de simplifier l'ajout de nouveaux véhicules à la base de données du garage ou de rechercher des véhicules déjà enregistrés. En plus de cela, l'application offre des fonctionnalités avancées pour la gestion des réparations, des pièces commandées et des factures associées aux véhicules.

L'objectif principal est de centraliser et de numériser la gestion des données des véhicules et des clients, rendant ainsi le processus plus efficace, rapide et moins sujet aux erreurs humaines.

2. Analyse des besoins

2.1. Exigences fonctionnelles

- Scanner une plaque d'immatriculation : L'application doit permettre d'extraire le numéro de la plaque d'immatriculation d'un véhicule en utilisant la caméra du smartphone ou en analysant une image stockée sur l'appareil. Pour ce faire, elle doit avoir accès à la caméra du téléphone ou aux fichiers d'images stockés.
- Gérer les clients : L'application doit permettre l'ajout, la modification et la suppression des informations des clients. Elle doit permettre d'associer un ou plusieurs véhicules à un client spécifique.
- Gérer des véhicules : L'application doit permettre l'ajout, la modification et la suppression des véhicules. Elle doit permettre d'associer des réparations à un véhicule spécifique.
- Gérer des réparations : L'application doit permettre d'ajouter, modifier ou supprimer les informations concernant les pannes, les travaux effectués, ainsi que les pièces commandées lors des interventions.

  Elle doit permettre d'ajouter ou de consulter les factures liées à une réparation, ainsi que le statut de paiement de celle-ci.
- Rechercher un véhicule : Si la plaque d'immatriculation scannée est déjà présente dans la base de données, l'application doit afficher les informations du véhicule lié à cette plaque. Si le véhicule n'est pas trouvé dans la base de données après le scan, l'application doit proposer un formulaire pour ajouter le véhicule. Ce formulaire doit comporter les champs suivants : numéro de client (et permettre la création d'un nouveau client si nécessaire), numéro de matricule, type de véhicule, marque, modèle, couleur, etc.

  L'application doit aussi proposer une barre de recherche pour permettre la recherche manuelle d'un numéro de plaque.
- Historique des réparations : L'application doit permettre de visualiser l'historique des réparations et interventions effectuées sur chaque véhicule.

2.2. Exigences non-fonctionnelles

- Performance : L'application doit être réactive avec un temps de réponse rapide pour les opérations courantes (scan, recherche, etc.).
- Fiabilité : L'application doit être stable et fonctionner de manière fiable sans crashs fréquents. Les opérations de base de données doivent être robustes pour éviter les pertes ou corruptions de données.
- Utilisabilité : L'application doit avoir une interface utilisateur intuitive, facile à naviguer, même pour des utilisateurs ayant peu de compétences techniques. Les formulaires doivent être clairs, avec des validations pour éviter les erreurs de saisie.

- Portabilité : L'application doit être compatible avec la majorité des versions d'Android (au moins Android 7.0 et versions ultérieures).
- Maintenance : Le code doit être bien structuré et documenté pour faciliter les futures mises à jour et la correction des bugs. Le pipeline CI/CD mis en place doit permettre des déploiements fréquents et sûrs.

3. Architecture préliminaire

L'architecture de l'application est construite et pensée pour faciliter à l'utilisateur l'accès à un maximum de fonctionnalités. L'application est développée en utilisant Android Studio avec le langage Kotlin.

- Interface utilisateur (UI) : L'application est structurée autour de plusieurs pages, chacune dédiée à une fonctionnalité spécifique. Les utilisateurs peuvent naviguer entre ces pages à l'aide de boutons. Les principales fonctionnalités incluent l'accès à la caméra pour scanner et lire les plaques d'immatriculation, un formulaire pour ajouter un client ou un véhicule, un formulaire pour décrire une panne, la recherche de véhicules, ainsi que l'historique des réparations et des factures liées aux réparations.
- Reconnaissance de plaque d'immatriculation : Pour réaliser cette fonctionnalité, nous envisageons deux options principales : OpenCV, une bibliothèque open-source de vision par ordinateur, ou Plate Recognizer, un logiciel de reconnaissance automatique de plaques d'immatriculation.

  OpenCV est largement utilisé pour traiter des images et vidéos, et peut être combiné avec la reconnaissance optique de caractères (OCR) pour extraire les numéros de plaques d'immatriculation à partir d'images capturées.

  Plate Recognizer est une solution dédiée à la reconnaissance de plaques. Elle offre jusqu'à 2500 requêtes par mois, ce qui est suffisant pour l'usage prévu de l'application.
- Base de données : Pour stocker les informations nous utilisons Android Room. Celui-ci nous offre la possibilité de structurer nos données de manière relationnelle, notamment pour lier les clients, véhicules et réparations. De plus, il permet de simplifier le code et éviter ainsi des erreurs comparé à l'utilisation de SQLite directement.

4. Mockup

Dans la section documentation de notre projet sur GitHub, vous pouvez accéder au document PDF contenant les mockups en suivant ce lien. Ce fichier présente les interfaces utilisateur prévues, offrant un aperçu visuel des fonctionnalités de l'application.

5. Description des choix techniques

Pour réaliser la conception et le développement de cette application, nous avons du penser à des solutions robustes, performantes et adaptées aux besoins spécifiques du garage.

- Android studio avec le langage Kotlin : Kotlin est préférable pour son intégration avec les outils Android modernes.
- OpenCV / Plate recognizer : Nous n'avons pas encore arrêté notre choix entre OpenCV et Plate Recognizer pour la reconnaissance des plaques d'immatriculation. OpenCV est une bibliothèque puissante pour le traitement d'images, offrant une grande flexibilité pour les prétraitements nécessaires à l'OCR, notamment dans des conditions de luminosité et d'angle de vue variées. Plate Recognizer, quant à lui, est un logiciel spécialisé dans la reconnaissance

automatique de plaques offrant jusqu'à 2500 requêtes par mois, ce qui est suffisant pour les besoins de l'application.

- Android Room pour la base de données : Android Room a été sélectionné pour sa capacité à offrir des fonctionnalités avancées telle qu'une intégration simplifiée avec d'autres composants Android, ce qui améliore la robustesse et la maintenabilité de l'application.

6. Processus de travail

6.1. Description

Pour le développement de cette application, nous avons choisi de suivre une approche agile, avec des itérations courtes pour permettre de faire des ajustements réguliers.

- Scrum : Chaque sprint couvrira une semaine, avec un Daily Scrum de 10 à 15 minutes chaque matin pour planifier nos tâches quotidiennes. Nous organiserons également une petite rétrospective chaque soir pour recueillir les difficultés rencontrées dans la journée.
- Planification et analyse des besoins : Lors de cette première étape, nous avons identifié les besoins du mécanicien et défini les exigences fonctionnelles et non-fonctionnelles de l'application. Nous avons essayé de chercher à faire une application qui sera axée sur la simplicité d'utilisation.
- Prototypage et conception : Un mockup de l'application a été créé pour visualiser l'interface utilisateur et les flux de navigation.
- Développement : Le développement se déroule en plusieurs étapes selon leur priorité. Chacune de ces étapes sera dédié à l'implémentation d'une ou plusieurs fonctionnalités (scan OCR, gestion des véhicules, gestion des clients, etc.).
- Tests : Chaque fonctionnalité devra être soumise à des tests unitaires et d'intégration pour s'assurer que la fonctionnalité est correcte. Les tests devront aussi inclure des prise de photos réelles pour la lecture de plaques selon différentes conditions de lumière, de vue par exemple.
- Déploiement : Une fois les fonctionnalités principales implémentées et fonctionnelles, l'application pourra être déployée dans des conditions réelles dans le garage du mécanicien puis nous réaliserons de potentielles modifications si nécessaire pour rendre l'application la plus optimale.

6.2. Organisation du kanban

Le kanban lié à Scrum sera mis en place avec GitHub Project, lié à l'organisation AutoGestion2024.

Voici un petit résumé de chaque colonne :
- ToDo : tâches à réaliser avant la date limite du projet.
- In progress : tâches en cours de réalisation.
- Done: tâches totalement terminées sur l'environnement de pré-production et peuvant être déployées.

6.3. Workflow git

- Créer une nouvelle branche et basculer dessus : Création de nouvelles branches pour chaque fonctionnalité et sur laquelle nous travaillons.
- Ajouter des modifications : Après avoir modifié certains fichiers, nous les ajoutons à la zone de staging.
- Commiter et push les modifications : Nous commitons et pushons nos modifications dans la branche actuelle.
- Créer une pull request : Pour transmettre nos modifications, nous créons une pull request.

- Mainteneur : fusionner la pull request : Le mainteneur du dépôt upstream accepte nos modifications et les fusionne dans la branche « main ».

6.4. CI/CD Pipeline

Pour notre pipeline CI/CD, nous utilisons les GitHub Actions. Nous avons configuré un processus automatisé qui se déclenche lors de l'ouverture d'une pull request. Ce processus comprend deux étapes principales : le build de l'application et l'exécution des tests. Cela permet de s'assurer que le code soumis est valide et ne casse pas les fonctionnalités existantes avant qu'il ne soit intégré dans la branche principale.

Une fois la pull request acceptée, les mêmes étapes de build et de tests sont à nouveau exécutées pour vérifier l'intégrité du code final. En plus de ces vérifications, nous générons également une release qui inclut l'APK de notre application. Cette release permet de distribuer l'application de manière contrôlée et facilite son déploiement.