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

AutoGestion

PDG Projet 2024-2025

06.09.2024

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 de petite taille (1-2 personnes).

L'application permet de scanner les plaques d'immatriculation des véhicules afin de simplifier le processus de recherche de client Le véhicule du client doit être déjà enregistrés. En plus de cela, l'application offre des fonctionnalités avancées pour la gestion des réparations 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 permet d'extraire le numéro de la plaque d'immatriculation d'un véhicule en utilisant la caméra du smartphone. Pour ce faire, elle a accès à la caméra du téléphone et à internet pour pouvoir envoyé des requêtes à une API externe.
- Gérer les clients : L'application permet l'ajout, la modification et la suppression des informations des clients. Elle permet d'associer un ou plusieurs véhicules à un client spécifique.
- Gérer des véhicules : L'application permet l'ajout, la modification et la suppression des véhicules. Elle permet d'associer des réparations à un véhicule spécifique.
- Gérer des réparations : L'application permet d'ajouter, modifier ou supprimer les informations concernant les pannes, les travaux effectués lors des interventions.
  Elle permet 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 affiche les informations du véhicule lié à cette plaque.
  L'application doit aussi proposer une barre de recherche pour permettre la recherche manuelle d'un numéro de plaque.
- Historique des réparations : L'application permet de visualiser l'historique des réparations et interventions effectuées sur chaque véhicule.

2.2. Exigences non-fonctionnelles

- Performance : L'application est réactive avec un temps de réponse rapide pour les opérations courantes (scan, recherche, etc.).
- Fiabilité : L'application est stable et fonctionner de manière fiable sans crashs fréquents. Les opérations de base de données sont robustes pour éviter les pertes ou corruptions de données.
- Utilisabilité : L'application a une interface utilisateur intuitive, facile à naviguer, même pour des utilisateurs ayant peu de compétences techniques. Les formulaires sont clairs, avec des validations pour éviter les erreurs de saisie.
- Portabilité : L'application est compatible avec la majorité des versions d'Android (au moins Android 7.0 et versions ultérieures).
- Maintenance : Le code est bien structuré et documenté pour faciliter les futures mises à jour et la correction des bugs. Le pipeline CI/CD mis en place permet des déploiements fréquents et sûrs.

3. Architecture

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. La création de l'interface se fait avec le framework Jetpack compose.
- Reconnaissance de plaque d'immatriculation : Pour réaliser cette fonctionnalité, nous avons finalement choisi d'utiliser Plate Recognizer, une API de reconnaissance automatique de plaques d'immatriculation. La communication se fait au travers du framework Retrofit2.
- 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.

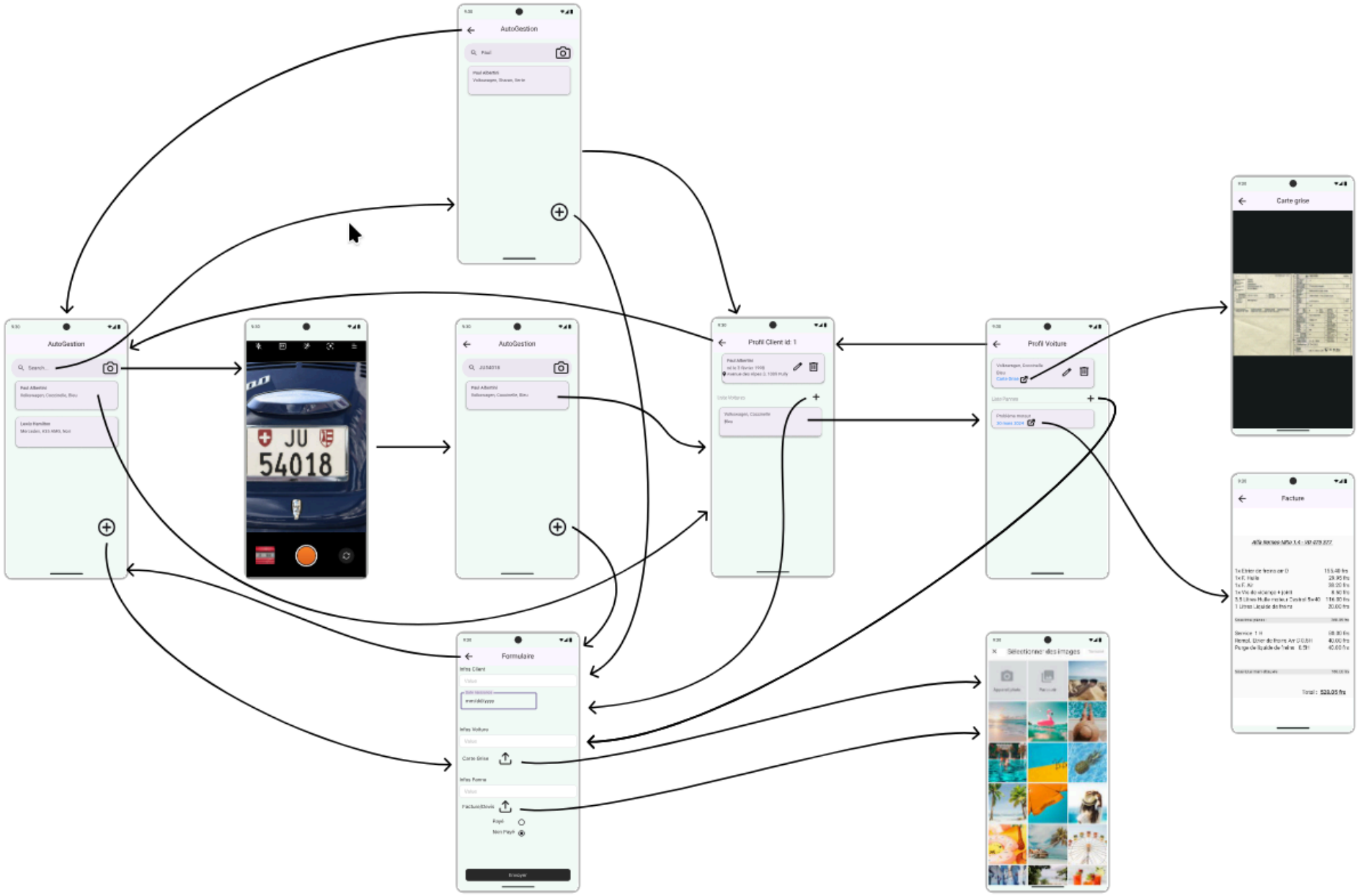
4. Landing page

Nous avons crée une landing page servant de vitrine à notre application et qui présente les fonctionnalités et avantages de notre projet. Elle est accessible via ce lien.

5. 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.

Le mockup ne représente pas forcement le produit finale, mais une idée générale de l'agencement de notre application. Il a été réalisé avant que nous ne sachions comment faire notre application, c'est pourquoi le produit finale ne ressemble pas exactement au mockup.


AutoGestion
AutoGestion
AutoGestion
JU
54018
Profil Client id: 1
Profil Voiture
Carte grise
Facture
Formulaire
Sélectionnez des images

6. 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.
- Plate recognizer : Plate Recognizer, quant à lui, est une API spécialisé dans la reconnaissance automatique de plaques offrant jusqu'à 2500 requêtes par mois, ce qui est suffisant pour l'usage prévu 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.

7. Processus de travail

7.1. 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 a couvert une semaine, avec un Daily Scrum de 10 à 15 minutes chaque matin pour planifier nos tâches quotidiennes. Nous avons organisé 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 s'est déroulé en plusieurs étapes selon leur priorité. Chacune de ces étapes a été dédié à l'implémentation d'une ou plusieurs fonctionnalités (scan OCR, gestion des véhicules, gestion des clients, etc.).
- Tests : Certaines fonctionnalités ont été soumises à des tests unitaires pour s'assurer que la fonctionnalité est correcte. Nous avons aussi inclus des tests de 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 a été déployée dans des conditions réelles dans le garage du mécanicien. Par la suite, nous réaliserons de potentielles modifications si nécessaire pour rendre l'application la plus optimale.

7.2. Organisation du kanban

Le kanban lié à Scrum a été 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.

7.3. 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 ».

Une protection sur les merges de la branch main à été mise en place. Chaque merge doit avoir une PR approuvé par une autre personne. Une autre empêche de push directement sur main TODO : completé

7.4. 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.

8. Déploiement

Pour lancer notre projet sur votre téléphone :

Nous utilisons plusieurs frameworks dans notre application, assurez-vous de bien comprendre leur fonctionnement : Android Room, Jetpack Compose, et Retrofit (Autre ?? a ajouter).

1. Installer Android Studio sur votre ordinateur.
2. Cloner le dépôt du projet.
3. Compiler-le pour vérifier que tout fonctionne correctement.
4. Sur le téléphone, activer le mode développeur.
5. Activer ensuite le débogage USB.

Et voilà ! À chaque fois que vous compilerez votre application, elle sera automatiquement installée sur votre téléphone pour que vous puissiez la tester en mode live.

9. Contributions

Voici les étapes pour contribuer à notre projet :

1. Créer une nouvelle issue : Sur le dépôt GitHub du projet, ouvrir une nouvelle issue inexistante.
2. Forker le projet : Créer un fork.
3. Cloner le dépôt : Cloner le fork localement avec la commande suivante :
   `git clone https://github.com/Username/AutoGestionApp.git`

4. Créer une nouvelle branche : Créer une nouvelle branche pour réaliser le code :
   `git checkout -b nouvelle-fonctionnalite`
5. Ajouter des modifications : Faire les modifications nécessaires dans le code et ajouter-les à la zone de staging :
   `git add .`
6. Commiter les modifications : Commiter les modifications avec un message explicatif :
   `git commit -m "Ajout de la nouvelle fonctionnalité"`
7. Pousser les modifications : Pousser les modifications vers le dépôt distant :
   `git push origin nouvelle-fonctionnalite`
8. Créer une pull request : Après avoir poussé les modifications, sur GitHub, ouvrir une nouvelle pull request pour que les modifications soient examinées et fusionnées dans le dépôt principal, si les modifications sont acceptées.

10. Point d'amélioration

1. Nous n'avons pas eu le temps d'implémenter correctement la release de l'apk Android. Pour ce faire, nous aurions dû mettre en place un système de signature automatique de notre apk dans les GitHub action. Sans quoi, l'application ne peut pas être lancée sur un téléphone. Pour contourner les sécurités Android, le .apk fournit est la version debug du l'application, ce qui ne requière pas de signature.
2. La plaque de la voiture ne peut pas être complétée automatiquement dans le formulaire de création.
3. Malheureusement, avec le manque de temps, nous aurions voulu mettre en place la possibilité de transférer les données en cas de changement de téléphone.
4. Factorisation du code. Étant donnée que c'est la premier fois que nous faisions un projet Android, nous n'avons pas pu appliquer la factorisation du code de manière efficace et propres.