

Projet d'examen

Développement d'applications mobiles

Daniel Schreurs

2024-2025

Introduction

Ce travail s'inscrit dans le cadre du cours intitulé « Développement d'applications mobiles ». Son objectif principal est de vous offrir l'opportunité de concevoir et créer une application mobile personnelle en utilisant la technologie Flutter. Cette démarche vise à consolider les notions enseignées durant le cours tout en vous permettant de répondre à un besoin spécifique.




De plus, ce projet peut également devenir un moyen de vous démarquer auprès de futurs employeurs, en démontrant vos compétences en développement d'applications mobiles et en mettant en avant votre capacité à concevoir des solutions innovantes et fonctionnelles.

Vous devez dans le cadre de ce projet réfléchir à un besoin pour un utilisateur final afin de répondre à celui-ci au travers d'une application mobile. Vous êtes libre de choisir ce besoin. Nous vous offrons une certaine liberté pour mettre à profit votre créativité et les connaissances que vous avez acquises.

Ce projet met aussi en avant l'approche agile. C'est-à-dire qu'il est attendu que vous fournissiez un travail régulier pendant l'année afin que nous puissions vous orienter vers des pistes d'améliorations.

Documentation

Votre dépôt doit obligatoirement contenir un fichier `readme.md`. Il s'agit de vendre votre application tout en démontrant votre démarche rigoureuse et efficace. Vous vous adressez potentiellement à un futur employeur et donc une personne qui n'aura pas nécessairement l'occasion de compiler votre projet. Votre `readme.md` contiendra donc au moins :

-  • Une présentation des principaux dossiers de votre dépôt. Quelles sont les différentes ressources qu'il contient à la racine ? Par exemple, les maquettes, vos inspirations, etc. Si vous avez fait des efforts quant à l'organisation de vos fichiers dans le dossier `lib` expliquez les ici.
 - Mikail • Une présentation de votre application. Ce dernier répond à un besoin, présentez-le. Ne faites aucune hypothèse sur le niveau de connaissances de votre lecteur. Vous vous adressez ici à un internaute quelconque qui découvre votre dépôt. Évitez un jargon technique dans cette partie de votre présentation.
 - Luca • Une brève étude de l'existant. L'idée étant de savoir si d'autres ont déjà couvert le besoin auquel vous essayez de répondre. Ce qui est demandé ici, au-delà d'une brève description, ce sont les points forts et les points faibles de ces différentes applications. Il peut être intéressant de faire un tableau pour mettre en regard les avantages et les inconvénients. Enfin, mettez des captures d'écran des applications afin que l'on comprenne mieux de quoi on parle.
 - Mikail • Parlez de votre public cible. À qui s'adresse votre application et surtout *comment* prenez-vous en compte ce public-là ?
 - Une présentation des différentes fonctionnalités de votre application au travers de récits utilisateurs (user story). Soit une description courte et simple d'un besoin ou d'une attente exprimée par un utilisateur. Chacun de ces récits suivra la syntaxe "En tant que <qui>, je veux <quoi> afin de <pourquoi>" :
 - Le *qui* indique le rôle/statut de l'utilisateur à ce moment-là. Par exemple « membre premium » ou « utilisateur non identifié ». Pour mieux illustrer la diversité des besoins, on peut également utiliser le concept de persona, c'est-à-dire une personne fictive et représentative à laquelle on peut s'identifier pour mieux comprendre ses attentes. L'identification et la description des personas se fait alors avant de commencer l'écriture des récits utilisateurs. Par exemple, "Odile est une enseignante qui utilise pour la première fois le système".
 - Le *quoi* décrit succinctement la fonctionnalité ou le comportement attendu. Le but du récit n'est pas d'en fournir une explication exhaustive.
 - Le *pourquoi* permet d'identifier l'intérêt de la fonctionnalité et d'en justifier le développement. Il permet également de mieux évaluer la priorité des fonctionnalités.
- Pour chacune de ces fonctionnalités, présentées par un récit utilisateur, vous présenterez les maquettes qui s'y rapportent.
-  • Un état d'avancement pour chaque fonctionnalité de votre application. Ceci doit évidemment être mis à jour régulièrement. Dès lors que vous aurez terminé de programmer une fonctionnalité, ajoutez dans le document `readme.md` un `.gif` qui l'illustre. Vous pouvez vous servir de [GIF Brewery](#)¹ ou de [Gyazo](#)².
-  • Enfin, nous vous demandons d'ajouter une section dédiée aux développeurs, dans laquelle vous expliquez les étapes nécessaires pour compiler l'application. Cette documentation doit être simple et surtout efficace. La valeur ajoutée réside dans le fait de mentionner les dépendances vers d'autres packages, comme les DTO, ainsi que de préciser les versions minimales requises. Pensez également à indiquer les fichiers contenant d'éventuelles clés API.

¹Si vous êtes sous macOS.

²Si vous êtes sous Windows.

Maquette

Pour les étudiants en infographie³ il est obligatoire de se concentrer pleinement sur l'expérience utilisateur et l'identité graphique de votre application. Il est donc nécessaire de produire des maquettes avant l'implémentation. Vous devez :

- Renseigner de manière claire vos [Moodboard](#).
- Fournir des maquettes : dans vos choix de conception, placez la personne au premier plan, afin de prendre en compte *ses besoins* avec *ses incapacités* permanentes, temporaires, contextuelles, ou changeantes—c'est à dire nous tous.⁴ Toujours dans cette optique, si vous devez concevoir le design de votre application, optez pour une maquette avec une taille d'écran plus petite. Souvent, ce sont ces petits écrans qui posent des problèmes par la suite.

Application

La partie qui vous demandera le plus de temps sera la conception de l'application en Flutter. Pour cela, il est nécessaire d'être à l'aise avec les projets réalisés en cours, car ils contiennent la plupart des éléments dont vous aurez besoin. Vous veillerez à :

- Gérer l'identification⁵ ainsi que l'authentification de votre utilisateur.
- Créer un écran d'accueil qui *accueille* votre utilisateur. Pensez à sa première utilisation : comment faire pour qu'il *comprenne* et *adopte* votre application ? Soignez l'[onboarding](#). Ne placez pas de gros boutons de navigation au centre de cet écran⁶. Présentez plutôt une sélection des ressources les plus pertinentes pour lui.
- Créer un écran de détail qui permet d'afficher plus d'informations sur une ressource⁷, *et de la modifier*. Par exemple, sur l'écran d'accueil, on peut lister les dernières courses, et lorsqu'on clique sur l'une d'elles, on affiche les détails avec la possibilité de modifier les informations.
- Mettre à jour des informations⁸. Gérez la persistance des données avec Firebase ou une API existante. **Attention : il est indispensable de respecter l'architecture vue en classe, sans quoi la réussite ne sera pas possible.**
- Exploiter les fonctionnalités offertes par l'environnement des périphériques mobiles, par exemple les notifications, la caméra, l'accéléromètre, etc.
- Ajouter une dimension de *gamification* dans l'application. Par exemple, intégrer des récompenses ou des badges pour encourager l'utilisateur à atteindre certains objectifs ou à utiliser l'application de manière régulière.
- Mettre en place des stratégies de secours lorsque l'utilisateur n'a pas de connexion internet. Veillez à prévoir des écrans d'erreurs et de chargement pour éviter que l'application ne semble buggée ou non réactive.
- Anticiper la maintenabilité du projet, surtout en s'assurant que le projet fonctionne avec les dernières versions stables.

³Pour les autres vous n'êtes pas obligé de les fournir.

⁴Il faut donc relire les [principes de conceptions inclusives](#).

⁵Je ne souhaite pas avoir de formulaire où il faut retaper 2 fois son mot de passe. En revanche, je souhaite pouvoir montrer/cacher mon mot de passe.

⁶Il existe au moins deux possibilités pour la navigation : soit en bas de l'écran, soit via un menu latéral.

⁷En dehors de l'écran qui affiche le profil

⁸Le package [Shared preferences](#) permet uniquement de sauvegarder les préférences de l'utilisateur et ne remplace pas une base de données.

Quelques critères de qualité

La qualité du code est essentielle. Ignorer les bonnes pratiques rend le code incompréhensible et empêche l'évolution. C'est pourquoi il est crucial que vous vous engagiez à produire un code propre et bien structuré. Respecter ces standards ne vous rapportera pas de points supplémentaires, mais ne pas les suivre vous en fait perdre. De plus, il est nécessaire d'adopter l'architecture vue en classe, avec 3 projets distincts dans le dossier de votre code.

- Exploitez au maximum les possibilités des Widgets et factorisez votre code pour éviter les fichiers à rallonge. Divisez votre application en Widgets paramétrables. Pensez à réduire le couplage.
- Séparez les considérations graphiques des considérations algorithmiques. Nous évaluons aussi la maintenabilité de votre code. Accordez une attention particulière à l'organisation des fichiers et maintenez une cohérence dans votre approche. De plus, il est strictement interdit d'utiliser des [Magic numbers](#).
- Même s'il s'agit d'un projet scolaire, veillez à rester attentif à la gestion des clés d'accès d'une API, voire même de vos informations d'identification (credentials). Essayez de les regrouper au même endroit et de documenter cette faille de sécurité temporaire.
- Soignez votre historique de *commits*. Il est attendu qu'on puisse, à partir des vos message de commit, comprendre ce qui s'est passé lors du développement et suivre l'histoire de votre projet. Le projet doit se trouver *dès le début* sur la bonne organisation, celle de l'année en cours.
- Pensez à soumettre tous les fichiers nécessaires à la compilation. Nous devons pouvoir compiler votre application sur nos machines.
- Soyez attentifs aux indications que votre IDE vous fournit depuis l'onglet *Dart analysis*. Votre code ne peut pas contenir d'avertissements ou d'erreurs sans quoi vous perdrez des points.
- Pensez aux petits écrans, vous devez au plus tard pour la remise compiler votre application dans le simulateur en prenant une petite taille d'écran afin de corriger les problèmes de débordement.

Évaluation

La remise de votre projet se fera en session, la veille avant minuit. Le projet doit obligatoirement être déposé sur GitHub, dans l'organisation de l'année en cours. Pour ce faire, il faut accepter le devoir via le lien disponible sur Moodle.

Il s'agit ici d'une évaluation orale qui porte sur :

1. d'une part, *l'achèvement du projet*, la *qualité du code* et le *respect des contraintes*
2. d'autre part, votre capacité à comprendre, adapter et expliquer vos choix ainsi que votre code avec vos propres mots.

Plagiat et utilisation de code externe

Dans le domaine du développement, il est courant de consulter des ressources en ligne ou d'utiliser de petits extraits de code pour résoudre des problèmes spécifiques. Cependant, il est essentiel que tout code emprunté ou inspiré d'une source externe soit dûment cité et expliqué dans votre documentation. Il est strictement interdit de soumettre du code intégralement copié sans attribution et sans une compréhension approfondie de son fonctionnement. Vous devez vous assurer que votre contribution soit majoritairement la vôtre et reflète votre propre travail. L'utilisation non déclarée de code tiers sera considérée comme du plagiat et sanctionnée en conséquence.