Projet d'examen

Développement d'applications mobiles - TP

Daniel Schreurs et Cyril Soldani

2023–2024

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.
- 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.
- 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 regarde les avantages et les inconvénients. Enfin, mettez des captures d'écran des applications afin que l'on comprenne mieux de quoi on parle.
- 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 prémium » 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 pour les dev où vous expliquez ce qu'il faut faire pour pouvoir compiler l'application. Cette documentation doit être simple et surtout efficace.

[1] Si vous êtes sous macOS.
[2] Si vous êtes sous Windows.

Maquette

Pour les étudiants en infographie[3] 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 ce faire, il est nécessaire d'être à l'aise avec les projets réalisés en cours qui contiennent la plupart des parties dont vous avez besoin. Vous veillerez :

- À gérer l'identification 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 au centre de cet écran des gros boutons pour naviguer dans l'application[5]. Présentez plutôt une sélection des ressources les plus pertinentes pour lui.
- Créer un écran de détail qui permet d'afficher plus d'information sur une ressource.
- Mettre à jour des informations [6]. Gérez la persistance des données avec Firebase ou une API existante.
- Exploiter les possibilités du temps réel que vous offre l'environnement des périphériques mobiles. Par exemple, les notifications.
- Mettre en place des stratégies de secours quand l'utilisateur n'a pas de connexion internet. Veillez à prévoir des écrans d'erreurs et de chargement pour éviter l'aspect d'une application buggée qui ne réagit pas.

[3] Pour les autres vous n'êtes pas obligé de les fournir.
[4] Il faut donc relire les principes de conceptions inclusives.
[5] Il existe pour cela au moins 2 possibilités : soit une navigation dans le bas de l'écran et/ou un menu latéral.
[6] Le package Shared preferences, permet uniquement d'enregistrer les préférences de l'utilisateur et ne remplace pas une base de données.

Quelques critères de qualité

La qualité du code constitue un pilier fondamental de l'ingénierie logicielle. Ne pas suivre les bonnes pratiques rend la compréhension du code ainsi que sa future évolution laborieuse. C'est précisément en raison de ces considérations que nous insistons sur la nécessité pour vous de vous engager dans la discipline visant à produire un code de haute qualité. Il est important de noter que la conformité aux critères de qualité ne se traduit pas par une acquisition directe de points supplémentaires, mais négliger ces critères pourrait entraîner des déductions de points.

- 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'historique des modifications.
- 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.

Échéance et évaluation

La remise de votre projet se fera en session d'examen quelques jours avant la date de l'examen, elle vous sera communiquée en classe. Il s'agit ici d'une évaluation intégrée qui combine une partie théorique et pratique. Cette évaluation se déroulera donc en deux étapes. Tout d'abord, vous présenterez oralement votre projet, avec une évaluation basée principalement sur l'*achèvement du projet*, la *qualité du code*, le *respect des contraintes* et votre aptitude à expliquer et comprendre votre code. Cette première partie comptera pour 80% de votre cote finale.

Ensuite, dans un second temps, vous devrez effectuer une ou plusieurs petites manipulations en temps réel sur votre projet afin de démontrer vos compétences et de confirmer l'authenticité de votre travail. Cette partie compte donc pour les 20% restant. Nous calculerons une moyenne géométrique pondérée de ces deux parties pour obtenir votre cote finale.[7]

$$\text{Cote finale} = \text{Partie théorique}^{0.8} \times \text{Partie pratique}^{0.2}$$

[7] Pour les étudiants en informatique qui réalisent leur projet en groupe, il est important de noter que la cote finale sera individuelle.