

L'application Crazy Wolf

Application smartphone pour la gestion des
horaires et des services du personnel du
restaurant Crazy Wolf

TRAVAIL DE BACHELOR

DANIEL SANZ

Décembre 2020

Supervisé par :

Prof. Dr. Jacques Pasquier

et

Pascal Gremaud

Software Engineering Group

Table des matières

1	Introduction	2
1.1	Motivations et objectifs	2
1.2	Structure du rapport	2
1.3	Table des symboles	2
2	Domaine métier	4
2.1	Analyse	4
2.2	Problématique	4
2.3	Use case	4
3	L'application	5
3.1	Authentification & écran d'accueil - Scénario I	5
3.2	Mise en bourse d'un service - Scénario II	6
3.3	Postuler pour un service - Scénario III	7
3.4	Valider un échange - Scénario IV	8
3.5	Ajouter un service - Scénario V	10
3.6	Ajouter un serveur - Scénario VI	12
3.7	Doublages - Scénario VII	13
4	Eléments de programmation	14
4.1	Choix des technologies	14
4.2	Flutter	15
4.2.1	Dart	15
4.2.2	Architecture	16
4.2.3	Les widgets	16
4.2.4	Flux de données	19
4.3	La base de données	20
4.3.1	La méthodologie REST	20
4.3.2	REST appliqué à Dart	20
4.4	Notifications	20
4.5	Sécurité	20
4.5.1	Authentification	20
4.5.2	Firebase rules	20
5	Le client	21
5.1	Introduction	21
5.2	Echanges	21
5.3	Problèmes	21
6	Conclusion	22

Introduction

Qu'est que c'est?. C'est une phrase français avant le lorem ipsum. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

1.1 Motivation et Objectifs

Qu'est que c'est?. C'est une phrase français avant le lorem ipsum. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

1.2 Structure du rapport

Qu'est que c'est?. C'est une phrase français avant le lorem ipsum. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

1.3 Conventions d'écriture

Qu'est que c'est?. C'est une phrase français avant le lorem ipsum. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper. Qu'est que c'est?. C'est une phrase français avant le lorem ipsum. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus

placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Domaine métier

Qu'est que c'est?. C'est une phrase français avant le lorem ipsum. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

2.1 Analyse

Qu'est que c'est?. C'est une phrase français avant le lorem ipsum. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

2.2 Définition de la problématique

Qu'est que c'est?. C'est une phrase français avant le lorem ipsum. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

2.3 Cas d'utilisation

Qu'est que c'est?. C'est une phrase français avant le lorem ipsum. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Présentation de l'application

Dans ce chapitre, les scénarios les plus communs d'utilisation de l'application Crazy Wolf sont présentés au travers d'images commentées.

3.1 Scénario I

Authentification & écran d'accueil

Ici la serveuse Kim, qui n'a pas de privilèges, c'est-à-dire qu'elle n'est ni manager ni administratrice, se connecte et accède au premier écran possible. Puis elle, clic sur *Mes services*.

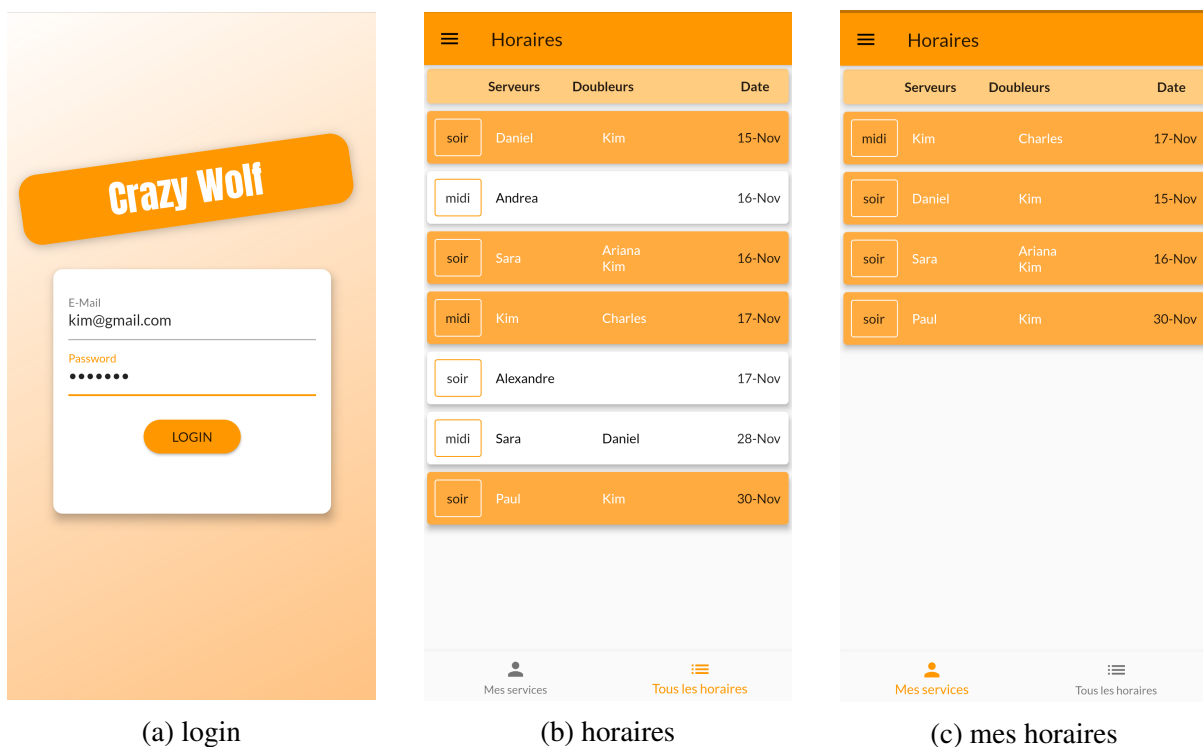


FIGURE 3.1 – scénario I

L'utilisateur doit dans un premier temps se connecter à l'aide d'identifiants déjà existant dans l'écran 3.1a. Une fois l'adresse mail et le mot de passe saisis, l'écran *Horaires* 3.1b s'affiche.

On y voit l'ensemble des horaires de travail. Les services sont définis par

- le type : midi ou soir.
- un ou plusieurs serveurs.
- zéro, un ou plusieurs doubleurs.
- la date

Les services sont ordonnés par date, les jours précédents au moment de la connexion ne sont pas affichés. Les horaires correspondant à la personne authentifiée sont de couleurs orange.

Les services sont affichés sous forme de liste scrollable.

On peut également naviguer à l'aide du menu inférieur à *Mes services* où seuls les horaires du serveur authentifié sont affichés. Comme on le voit dans 3.1c

3.2 Scénario II

Mise en bourse d'un service

Supposons que Kim, la serveuse authentifiée ne puisse pas travailler le 17 novembre. Elle souhaite donc se faire remplacer. Pour se faire, dans l'onglet *Horaires*, elle peut mettre son service en bourse en glissant le service en question sur la gauche.

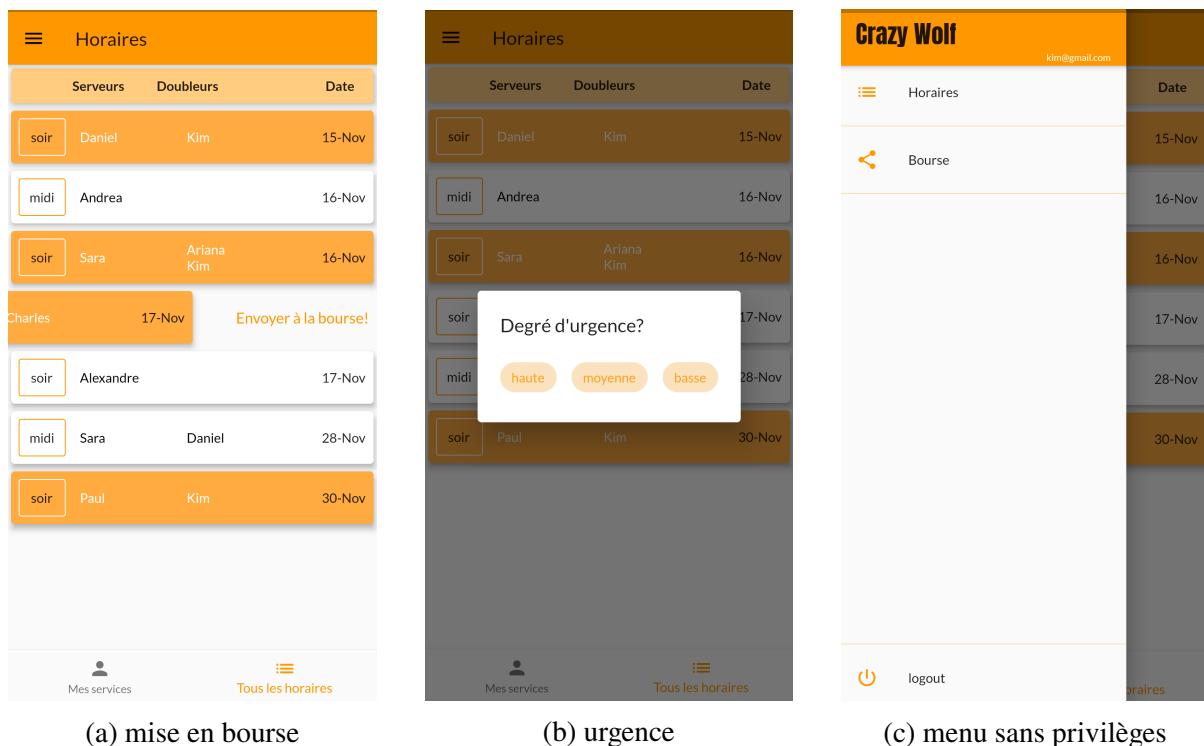


FIGURE 3.2 – scénario II - a

Une fois le glissement effectué un popup est s'affiche 3.2b demandant à quel point le remplacement est urgent. Une fois que l'utilisateur à répondu, le service est mis en bourse. Un snackbar¹ s'affiche pour le notifier que l'action à réussi. Si l'action venait à échouer par exemple à cause d'une déconnexion à internet ou car le service se trouve déjà en bourse, un snackbar avec un message d'erreur pertinent s'afficherait également.

Si l'action est réussie, tous les utilisateurs de l'application sont notifiés 3.3a comme quoi un nouveau service est disponible. La notification informe sur la disponibilité d'un service ainsi que l'urgence requise à y répondre.

Suit à ça, l'utilisatrice peut naviguer à l'aide du menu latéral 3.2c où s'affichent les options de navigation suivante :

- Horaires : Pour aller à l'écran *Horaires* 3.1b
- Logout : Pour se déconnecter, qui renvoie à l'écran 3.1a d'authentification
- Bourse : Pour aller à l'écran *Bourse aux jobs* 3.3b

1. Petite notification informative qui émerge dans la partie inférieur d'un écran

Dans l'onglet *Bourse aux jobs* 3.3b son service est disponible à toute personne souhaitant y postuler. Cet onglet est partagé parmi tous les utilisateurs de l'application.

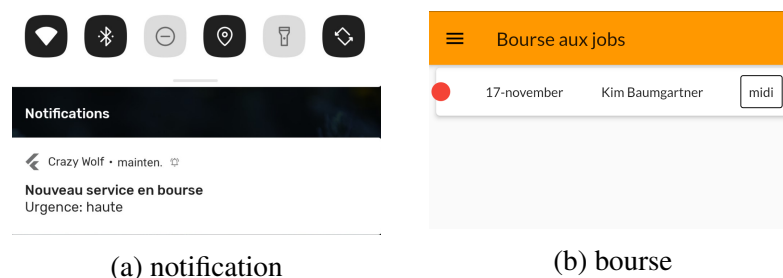


FIGURE 3.3 – scénario II - b

La couleur représente le degré d'urgence. Ainsi la convention suivante est appliquée :

- rouge : implique une urgence élevée.
- jaune : implique une urgence moyenne.
- vert : implique une urgence basse.

L'onglet *Bourse aux jobs* affiche tous les services actuellement en bourse sous forme de liste scrollable. L'utilisateur ayant mis son service en bourse doit patienter à ce que quelqu'un y postule.

Les utilisateurs sans privilèges sont autorisés à mettre un service en bourse uniquement s'ils y travaillent.

3.3 Scénario III

Postuler pour un service

Dans la continuation du scénario précédent, après avoir été notifiés, deux serveurs vont postuler au nouveau service mis en bourse. Leurs parcours étant identiques, nous allons en montrer qu'un. Supposons qu'ils se soient authentifiés comme vu en 3.1a et qu'ils se trouvent dans l'onglet *Bourse aux jobs* 3.3b.

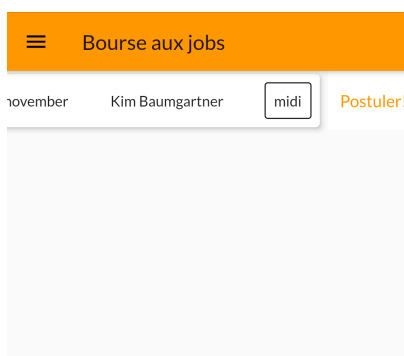


FIGURE 3.4 – postuler

Alors, ils peuvent glisser le service auquel ils souhaitent postuler sur la gauche 3.4. À nouveau, si l'opération réussit, un snackbar apparaît pour l'indiquer. Si au contraire, une erreur se produit, notamment car l'utilisateur a déjà postulé ou une erreur de connexion, un snackbar pertinent s'affiche aussi.

L'utilisateur ayant mis le service en bourse doit procéder à l'acceptation d'un seul postulant. Dans *Bourse aux jobs* 3.3b les éléments de la liste sont clicable. Lorsqu'un utilisateur clic deux options sont possibles :

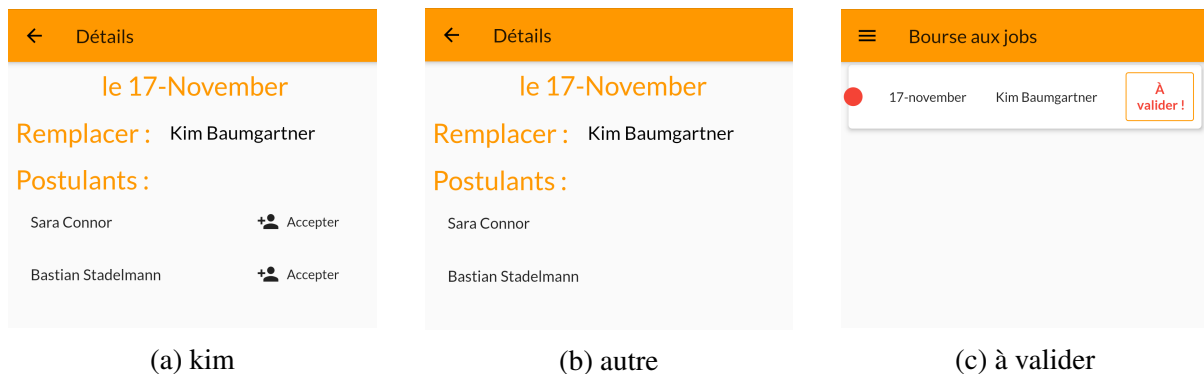


FIGURE 3.5 – scénario III

S'il s'agit de l'utilisateur ayant mis le service en bourse à l'occurrence Kim, l'écran *Détails* tel que 3.5a s'affiche.

S'il s'agit de n'importe quel autre utilisateur alors l'écran *Détails* s'affiche comme dans 3.5b

Ainsi, seul l'utilisateur ayant mis le service en bourse est en mesure d'accepter un ou une remplaçante.

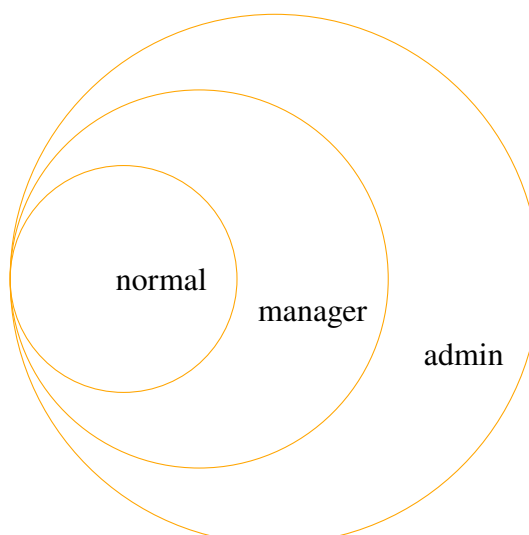
Pour se faire, Kim doit appuyer sur le bouton *accepter* de la personne de son choix. Elle accepte Sara. Tous les utilisateurs ayant des privilèges sont notifiés qu'un service nécessite validation. En attendant, l'élément de la bourse dans *Bourse aux jobs* 3.5c s'affiche comme nécessitant validation.

3.4 Scénario IV

Valider un échange

Toujours dans la continuation de notre exemple, nous allons voir ici la validation d'un échange. En effet, même si un utilisateur a accepté un remplaçant pour son service. Il faut encore que cette transaction soit validée par un utilisateur avec des privilèges.

Il existe trois types d'utilisateurs :



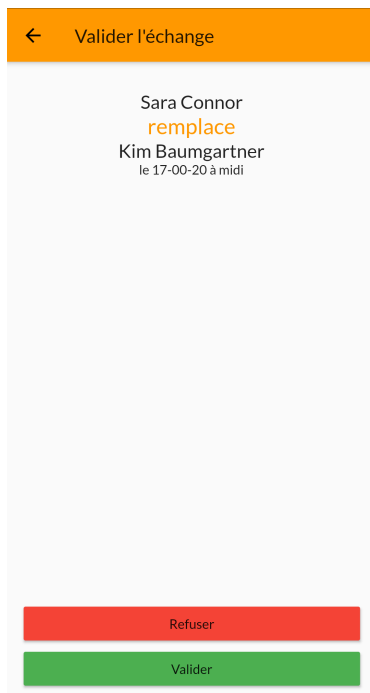
Les utilisateurs autres que *normal* ont des privilèges.

Normal : Peut demander un échange et accepter des postulants.

Manager : Peut valider un échange, créer des services et demander des doubleurs en renfort.

Admin : Peut ajouter de nouveaux serveurs.

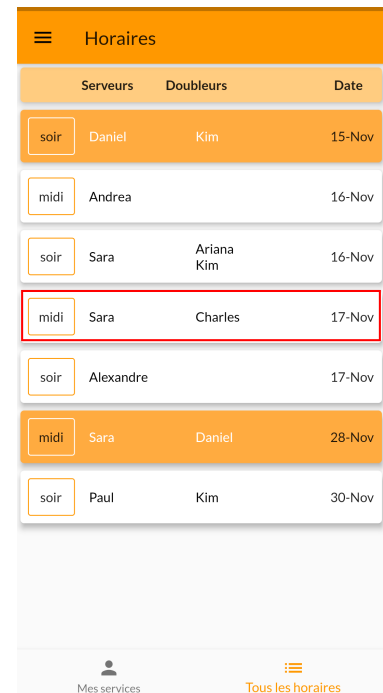
Lorsque l'on clic sur l'élément à valider 3.5c deux options sont possibles suivant si l'utilisateur authentifié 3.6a et *manager / admin* ou bien s'il est *normal* 3.6b.



(a) manager / admin



(b) normal



(c) Daniel : après validation

FIGURE 3.6 – scénario IV

S'il s'agit d'un utilisateur avec privilèges, alors il a l'option de valider l'échange 3.6a.

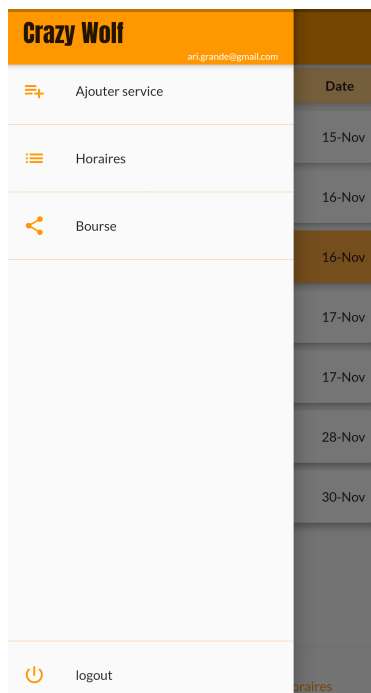
Supposons que Daniel, qui est administrateur valide l'échange. Alors, l'écran *Horaires* est modifié 3.6c pour tout le monde et affiche que c'est bien Sara qui travail le 17 novembre et non plus Kim.

De plus, l'élément dans *Bourse aux jobs* 3.3b est supprimé.

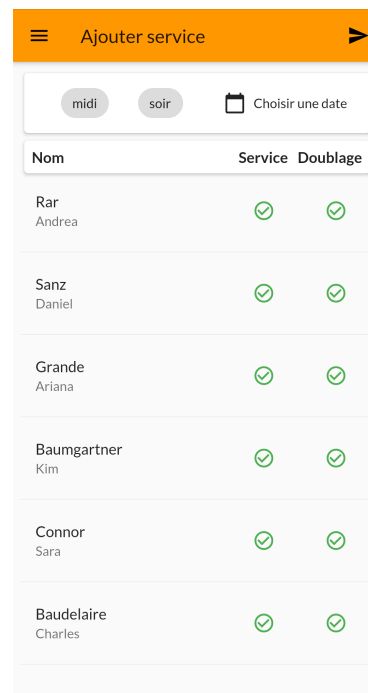
3.5 Scénario V

Ajouter un service

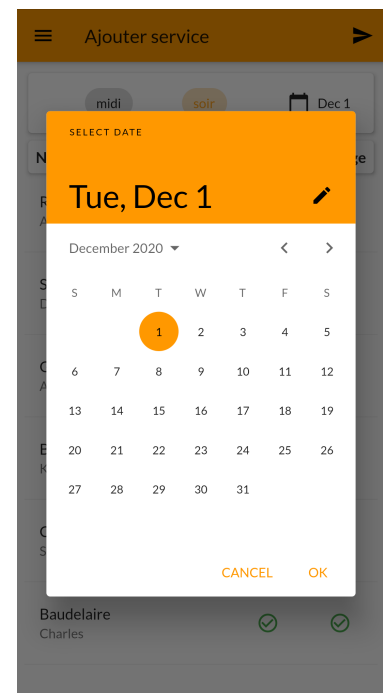
Afin de créer les horaires pour le personnel, un manager ou administrateur doit créer des services. Supposons que c'est Ari, une manager qui le fait. Après s'être authentifiée, elle peut à l'aide du menu latéral 3.7a naviguer à *Ajouter service* 3.7b



(a) menu manager



(b) ajouter service



(c) choix date

FIGURE 3.7 – scénario V - a

Pour créer le service, il faut choisir :

- une date en appuyant sur *Choisir une date* un popup avec le calendrier s'affiche 3.7c.
- un type en appuyant sur *midi* ou *soir* 3.8a.
- une ou plusieurs personnes au *Service* 3.8a.
- zéro, une ou plusieurs personnes au *Doublage* 3.8a.

Dans notre exemple Ari choisit le 1er décembre au soir. Andrea au service et Daniel au doublage.

Un service ne peut pas avoir la même personne au doublage et au service. Un service peut ne pas avoir de doubleurs.

De plus, l'opération ajouter le service aux horaires qui se fait au moyen du bouton envoyer dans le coin supérieur droit de la figure 3.8a retournera une erreur s'il existe un service du même type à la même date.

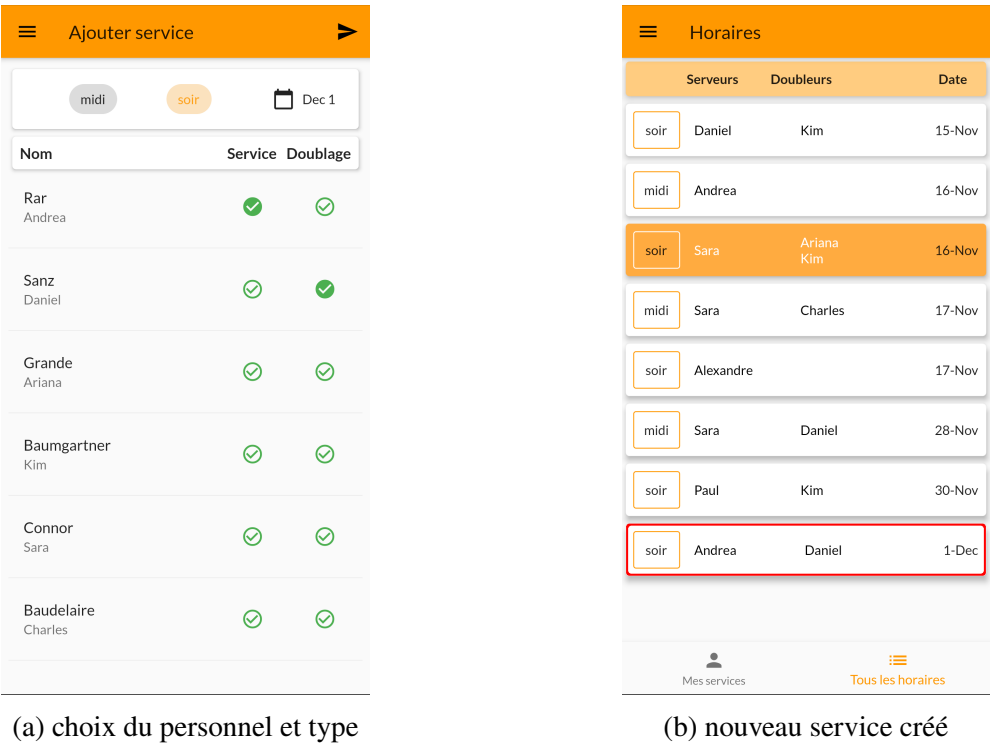


FIGURE 3.8 – scénario V - b

3.6 Scénario VI

Ajouter un serveur

Les utilisateurs ayant les privilèges administrateur ont la capacité d'ajouter de nouveaux serveurs.

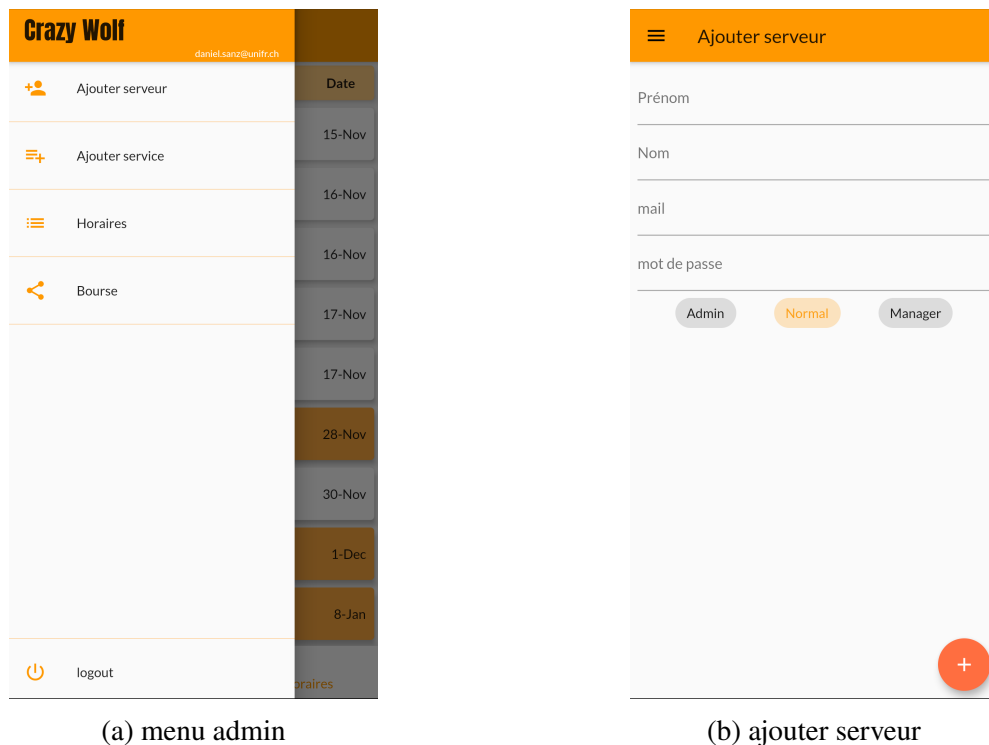


FIGURE 3.9 – scénario VI

Pour se faire, ils doivent après s'être authentifié, naviguer à l'aide du menu latéral 3.9a à l'onglet *Ajouter serveur*

Dans l'onglet *Ajouter serveur* il faut remplir les champs définissant un serveur. I.e.

- Prénom
- Nom
- Adresse mail
- Mot de passe
- Niveau de privilèges : administrateur, normal, manager.

Le niveau de privilège le plus courant étant *normal* il est déjà pré-sélectionné.

Le mot de passe doit être strictement plus long que 6 caractères.

Utiliser la même adresse mail pour deux serveurs différents est interdit. Cependant, deux serveurs peuvent théoriquement avoir le même prénom et le même nom de famille.

Un fois tous les champs remplis, l'administrateur peut ajouter le serveur dans le système en appuyant sur le bouton + dans le coin inférieur gauche de l'écran *Ajouter serveur* 3.9b

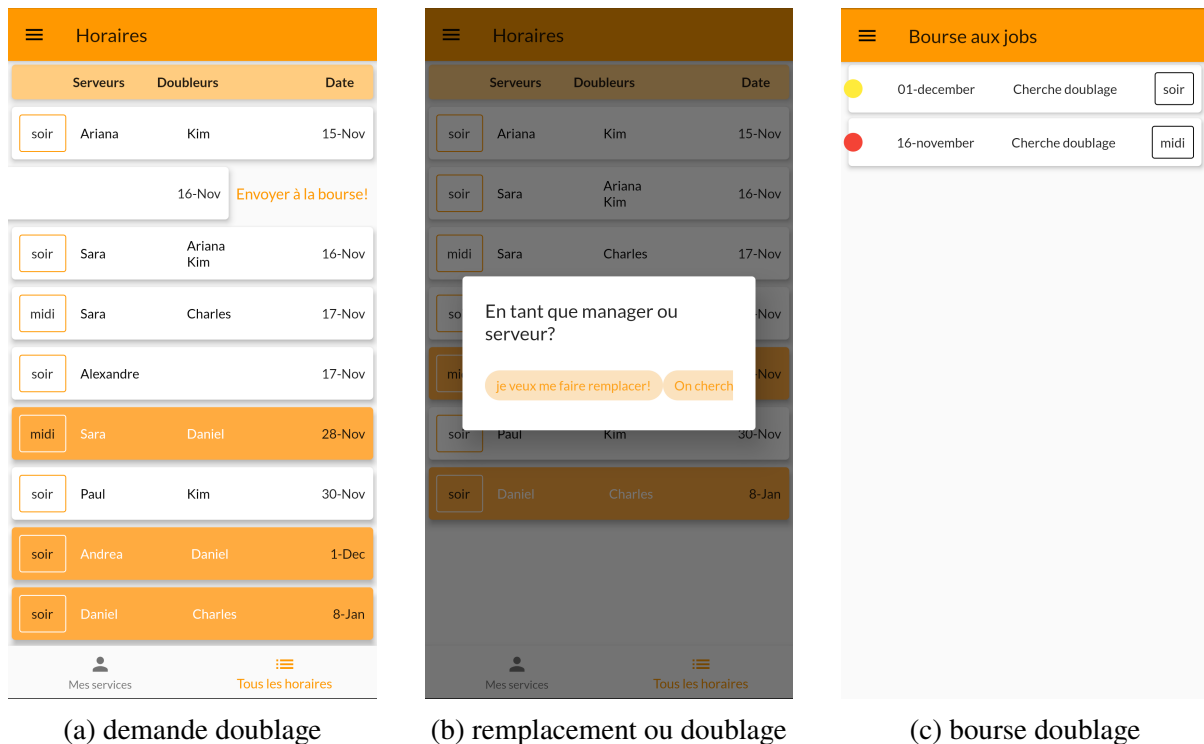
Ce sont les identifiants nécessaires pour s'authentifier dans l'écran 3.1a.

3.7 Scénario VII

Doublages

Ici on va voir comment un administrateur ou un manager peut demander du personnel supplémentaire pour un service. Ces personnes supplémentaires sont dites doubleurs et font un doublage.

La démarche est similaire à celle d'un échange.



(a) demande doublage

(b) remplacement ou doublage

(c) bourse doublage

FIGURE 3.10 – scénario VII

Dans un premier temps un utilisateur avec privilèges fait glisser un service sur la gauche 3.10a. Deux options sont possibles suivant si l'utilisateur travaille dans ce service ou pas.

- S'il y travaille alors un popup apparaît 3.10b pour demander si c'est bien une demande de doublage ou bien se faire remplacer
- S'il n'y travaille pas alors la demande de doublage est directement envoyée en bourse.

Dans cet exemple, deux services sont mis en bourse. Un où l'administrateur n'y travaille pas 3.10a. Comme pour les échange, le degré d'urgence 3.2b est demandé. On peut voir le service du 16 novembre en bourse. Et l'autre où l'administrateur travaille et où il choisit 3.10b qu'il cherche un doublage.

À la différence du scénario III 3.5, la condition pour pouvoir accepter des postulants n'est plus d'être l'utilisateur ayant mis le service en bourse mais d'avoir un niveau de privilèges supérieur à *normal*.

Eléments de programmation

Dans ce chapitre sont présentés les technologies, paradigmes et techniques liés à l'implémentation informatique de l'application. Notamment, lesquelles ont été utilisées, comment elles fonctionnent et, une à une, comment cela s'est traduit dans l'implémentation final de l'application. Les exemples de codes sont tirés de l'application.

4.1 Choix des technologies

Le développement d'applications pour smartphones est depuis un peu plus d'une décennie en pleine effervescence. Il existe, par conséquent, une multitude de frameworks, services, langages, méthodologies et paradigmes liés à leur développement.

Ces technologies aux noms exotiques, aux logos plus brillants les uns que les autres et aux conférences accrocheuses qui leur sont dédiées, font que leurs différences relèvent plus d'une stratégie marketing visant les informaticiens, réussie que des attributs intrinsèques de la technologie en question.

L'application étant d'une complexité modérée et ne demandant pas de ressources importantes comme tel pourrait être le cas pour une messagerie instantanée à grand échelle ou une application utilisant abondamment un domaine spécifique de connaissance comme le machine learning, le traitement d'images, les jeux vidéo, etc. Exclu *de facto* le choix d'une technologie basée exclusivement sur les performances ou sur le développement natif.

N'ayant jamais fait cela auparavant, il n'y a aucune préférence de ma part pour telle ou telle technologie.

Ces constats donnent lieu aux critères de sélection suivants :

- Développement cross-plateforme
- Simplicité
- Apprentissage d'un langage plutôt qu'une multitude
- Vaste documentation et ressources d'apprentissage

Le premier critère est celui qui réduit le plus la liste des possibilités. En effet, les frameworks permettant le développement d'applications pour Android et IOS ne se comptent pas en grand nombre. Il existe ¹ :

- Xamarin - Microsoft
- React Native - Facebook
- Flutter - Google
- Adobe PhoneGap - Adobe
- Ionic - MIT

Le choix parmi ces possibilités découle essentiellement de l'arbitraire. Toutefois, Ionic a été exclu car il est nécessaire de maîtriser HTML5 et par conséquent CSS mais encore Angular JS. Ce qui contredit le 3ème critère.

React native a été exclu pour des raisons similaires. I.e. l'apprentissage de divers langages.

Finalement, à la suite d'un cours de Academind d'une durée de 40 heures portant sur les aspects les plus basiques du développement jusqu'au déploiement de l'application en passant par le routage, la gestion de requêtes http, la connexion à tout un écosystème de bases de données, l'utilisation de caméra et géolocalisation, et même sur comment changer le logo de l'application, le choix s'est porté sur Flutter.

1. Liste non exhaustive

Flutter est un framework créé par Google. Ce dernier offre, pour les applications, le service Firebase qui englobe :

- Cloud Firestore
- Real time database
- Functions
- Machine learning
- Cloud messaging
- ...

Firebase s'intègre, par conception, particulièrement bien et facilement à Flutter. Même s'ils sont indépendants, leurs utilisation conjointe forme un seul écosystème plus facile à appréhender. Ainsi pour la base de données, le choix a été la Real time database. Car cette dernière fournit une API REST.

De plus, toujours dans cet écosystème, *Cloud messaging* est utilisé pour l'envoi des notifications et *Functions* pour effectuer des actions côté serveur lorsque la base de données subit des modifications.

4.2 Flutter

Flutter est un *Software Development Kit* (SDK) développé par Google permettant de concevoir des applications pour plusieurs plateformes. Notamment, Android et IOS avec un seul code source.

Flutter est aussi et surtout un framework. Il se caractérise par le fait qu'il va principalement dessiner des éléments à l'écran en se basant sur les pixels. Ainsi, il n'utilise pas, de base, de bibliothèques natives. Par exemple, pour dessiner un bouton, il ne va pas faire appel aux primitives bouton dans Android ou IOS mais va dessiner pixel par pixel l'objet souhaité.

4.2.1 Dart

Dart est le langage de programmation avec lequel certains éléments du framework ont été développés mais il s'agit surtout du langage dans lequel on implémente une application en Flutter.

C'est un langage orienté objet avec une syntaxe de type C à l'image d'autres langages comme Java ou C++. Sa syntaxe est proche du Java.

Il partage certaines fonctionnalités propres aux langages fonctionnels. Notamment, l'inférence de type ou encore certaines fonctionnalités comme les *map*, *functors* ...

En Dart, l'allocation de mémoire est gérée automatiquement et sa libération est faite par un garbage collector.

Voici un exemple minimal d'un *Hello world* en Dart :

```
1 void main() => print('Hello, World!');
```

Listing 1 – Hello world

4.2.2 Architecture

Pour mieux comprendre les différents éléments de programmation qui vont suivre, il est intéressant de s'attarder un peu sur l'architecture interne du framework. Flutter a une architecture en couches (*layers*).

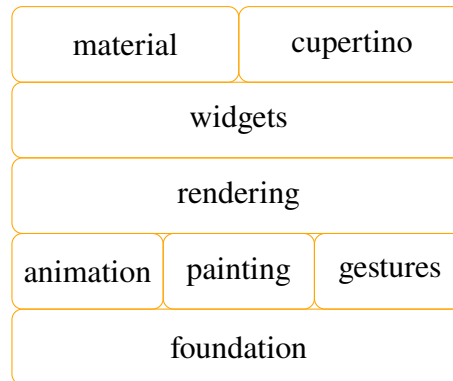


FIGURE 4.1 – Layers du Framework Flutter

Le *layer foundation* offre les classes de bases du framework sans avoir à tenir compte de l'*engine* écrit en C++ qui se trouve en dessous. Il offre aussi, au moyen de *animation*, *painting* et *gestures* les bases pour l'animation, le dessin et la détection de mouvement sur l'écran.

Le *layer rendering* offre une abstraction permettant de gérer l'affichage des éléments. Il va construire des objets (dérivés des widgets principalement) organisés sous forme d'arbre dynamique qui peuvent être affichés à l'écran.

Le *layer widgets* est une abstraction au-dessus de *rendering* qui respecte le pattern de conception composite. Ainsi, chaque objet dans le *layer rendering* lui correspond une classe dans le *layer widgets*.

Le *layer material* et *cupertino* sont un ensemble de classes prédéfinies et personnalisables qui implémentent tout un catalogue de widgets avec respectivement le style material design d'Android et le design propre à iOS.

4.2.3 Les widgets

Les widgets dans Flutter sont les éléments avec lesquels le programmeur a le plus de contact. Ils sont en effet, le composant principal d'une application Flutter. Voici un exemple :

```

1      Text(
2          "Crazy Wolf",
3          style: TextStyle(fontSize: 26, fontFamily: "Anton"),
4      ),
  
```

Listing 2 – Widget Text

Comme le code le sous-entend, les widgets peuvent s'encapsuler les uns dans les autres. Ainsi, Le widget *Text* accepte dans son constructeur une chaîne de caractères "Crazy Wolf" et un style qui est lui-même un widget *TextStyle*. À son tour, il accepte, entre autres, une taille et une police d'écriture en argument.

Cette encapsulation n'est pas un choix arbitraire mais trahi, comme vu auparavant, l'architecture du framework. En effet, Le *layer rendering* est un arbre dynamique de widgets (pas seulement) et le *layer widgets* respecte le pattern composite. Ainsi, il est naturel que les widgets se composent.

On en déduit que Flutter est un framework déclaratif plutôt qu'impératif. En effet, plutôt que de modifier des instances d'objets existants afin de varier l'affichage, Flutter construit de nouvelles instances de widgets selon les besoins.

La philosophie sous-jacente du framework est de composer de petits widgets aux fonctionnalités de bases afin de créer une fonctionnalité complexe.

Le terme widget englobe une multitude de verbes. Un widget peut ne pas avoir de représentation en tant que tel mais peut offrir un positionnement, une transformation, une taille, interaction avec l'utilisateur, animation, ...

Par exemple, le widget *Center* centre un autre widget en argument au centre de l'écran, dans la mesure de l'espace restant disponible.

```
Center(child: Text(...)),
```

Listing 3 – Widget Center

Les widget sont des classes immutables, c'est-à-dire qu'une fois l'objet instancié, il n'est plus possible d'en modifier le contenu. Il existe essentiellement deux types de widgets dans Flutter : ceux qui ont un état : *StatefulWidget* et ceux qui n'ont pas : les *StatelessWidget*.

StatelessWidget

Ce sont les widgets dont les propriétés ne varient pas dans le temps. Par exemple :

```
import 'package:flutter/material.dart';
class SplashScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(body: Center(child: CircularProgressIndicator(),));}}

```

Listing 4 – Exemple *stateless*

Ici, *SplashScreen* hérite de *StatelessWidget*. Afin, de déterminer la représentation visuelle d'un widget, il faut *override* la méthode *build()* de *StatelessWidget*. Ainsi, *SplashScreen* retourne un *Scaffold* qui est un écran d'affichage basique respectant le style material design. À son tour, *Scaffold* a une barre de progression circulaire (le *CircularProgressIndicator*) au centre (*Center*) de l'écran.

Cette classe représente l'écran affiché à chaque fois qu'une requête à la base de données se fait. Elle permet à l'utilisateur de savoir qu'une action est en cours et qu'il doit patienter.

StatefulWidget

Ce sont les widgets dont les propriétés peuvent varier dans le temps. Ces widgets ont un état, or ils sont immutables. C'est pourquoi ils sauvegardent l'état dans une classe séparée sous-classe de *State*.

Quand l'état change à l'aide de l'appel de fonction *setState()*, l'interface est mise-à-jour. En guise d'exemple voici une partie du code source de l'écran *Horaires* 3.1b. Notamment, le petit menu inférieur permettant d'afficher tous les horaires où seulement ceux de l'utilisateur authentifié.

```

1 class ScreenScheduleView extends StatefulWidget {
2   @override
3   _ScreenScheduleViewState createState() => _ScreenScheduleViewState();
4 }
5 class _ScreenScheduleViewState extends State<ScreenScheduleView> {
6   var _selectedIndex = 0;
7
8   void _onItemTapped(int index) {
9     setState(() {_selectedIndex = index;});
10  }

```

Listing 5 – Screen Horaires

La classe *ScheduleView* hérite d'un *StatefulWidget*. Dans la classe d'état il y a l'attribut : *selectedIndex*. L'indice sélectionné a pour valeur 0 à la construction de l'instance. Si la fonction *onItemTapped(int index)* est appelée avec un entier en argument alors, à l'aide de la fonction *setState* l'état est modifié et la fonction *build()* de *ScreenScheduleViewSate* est appelée. Ainis le *widget* est reconstruit en fonction de son nouvel état.

```

1 @override
2 Widget build(BuildContext context) {
3   return Scaffold(
4     bottomNavigationBar: BottomNavigationBar(
5       onTap: _onItemTapped,
6       currentIndex: _selectedIndex,
7       items: const [
8         BottomNavigationBarItem(
9           icon: Icon(Icons.person),
10          title: Text("Mes services")),
11         BottomNavigationBarItem(
12           icon: Icon(Icons.list),
13           title: Text("Tous les horaires"))]),

```

Listing 6 – BottomNavigation

Scaffold accepte un menu de navigation au bas de la page. Celui-ci est un widget : *BottomNavigationBar*. Ce widget prend en construction :

- une fonction qui est appelée lorsque une *tap* a été détecté sur l'un des deux icons définis dans la liste *Items*.
- un index qui définit lequel des éléments de la liste est actuellement sélectionné.
- une liste de *widgets* de type *BottomNavigationBarItem*.

Le premier élément de la liste i.e. "Mes services" a pour index 0. Et "Tous les horaires" 1. Par défaut, *selectedIndex* vaut 0. Dans un premier temps c'est "Mes services" qui est affiché.

Lorsque l'utilisateur clic su l'icône, *BottomNavigationBar* va appeler la fonction *onItemTapped* définie dans le code 5 avec l'index de l'icône en argument. *onItemTapped* va changer l'état du widget et par extension la valeur de *selectedIndex*. L'état ayant changé la fonction *build* est appelée et donc *BottomNavigationBar* redessiné avec un *currentIndex* potentiellement différent. Traduisant ainsi un affichage différent et l'icône sélectionné apparaîtra en orange.

4.2.4 Flux de données

Dans la partie précédente on constate que la gestion de l'état des widgets peut vite devenir encombrante. En effet, dans l'exemple, l'utilisation d'un *StatefulWidget* est naturelle, l'information étant un index n'est ni complexe ni lourde. Or comment, partager des données complexes entre les différents widgets ?

En général on fait la distinction entre l'état éphémère et l'état de l'application.

- L'état éphémère est celui qui peut entièrement être défini au sein d'un widget. Il n'a pas ou peu besoin d'être accédé à l'extérieur du widget et il ne varie pas de façon complexe.
- L'état de l'application est celui qui à l'inverse, peu se modifier de façon complexe, doit être préservé lorsque l'application est fermée, contient de vastes informations et doit être partagé parmi divers widgets. Comme par exemple, la liste des serveurs, la liste des services, les données d'authentification ...

Il existe de nombreuses techniques pour la gestion du flux de données dans les applications et en informatique en générale. Pour cette application, 3 façons ont été utilisées.

Pour transmettre les données d'un widget à ses enfants soit la transmission s'est faite par constructeur qui est classique dans les langages orientés objets soit à l'aide des patterns Provider et Consumer.

Pour transmettre les données d'un enfant à un parent, les callback ont été utilisés.

Par constructeur

Il s'agit à la fois de la méthode la plus répandue comme la plus simple. Voici un exemple :

```

1 class HoraireSubItem extends StatelessWidget {
2     const HoraireSubItem({//...,
3         @required this.isUserIn,}): super(key: key);
4     //...
5     final bool isUserIn;
6
7     @override
8     Widget build(BuildContext context) {
9         return Card(
10             color: isUserIn ? Colors.orangeAccent : Colors.white,
11             //...

```

Listing 7 – par constructeur

Cette classe 7 est celle qui se charge de dessiner un élément de la liste des horaires 3.1b. Son widget parent est *HoraireItem* qui s'occupe, entre autres, de fournir les informations suivantes à son fils : un service, les

serveurs correspondants, les doubleur correspondants et si le l'utilisateur authentifié travail dans le service en question, alors une variable booléenne de valeur *vraie* est transmise.

L'enfant, *HorairesSubItem* utilise cette variable pour afficher la *Card(...)*² en orange quand l'utilisateur authentifié y travaille et en blanc quand il n'y travaille point. Ainsi, c'est bien une instance différente de la carte qui va être créée pour changer la couleur de d'arrière fond et non une modification d'une instance existante.

Provider & Consumer

Théoriquement, rien n'empêche de transmettre toutes les informations par constructeur. Toutefois, l'arborescence des widgets devient rapidement profonde et dans la situation où une certaine information est nécessaire au niveau n mais aussi au niveau $n + i, i > 1$ il serait nécessaire de transmettre l'information dans tous les widgets intermédiaires alors qu'ils n'en ont pas besoin.

4.3 La base de données

4.3.1 La méthodologie REST

4.3.2 REST appliqué à Dart

4.4 Notifications

4.5 Sécurité

4.5.1 Authentification

4.5.2 Firebase rules

2. C'est un widget dans le catalogue material design

Coordination avec le client

5.1 Introduction

Qu'est que c'est?. C'est une phrase français avant le lorem ipsum. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

5.2 Echanges

Qu'est que c'est?. C'est une phrase français avant le lorem ipsum. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

5.3 Problèmes

Qu'est que c'est?. C'est une phrase français avant le lorem ipsum. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Conclusion

Qu'est que c'est?. C'est une phrase français avant le lorem ipsum. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper. Qu'est que c'est?. C'est une phrase français avant le lorem ipsum. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.