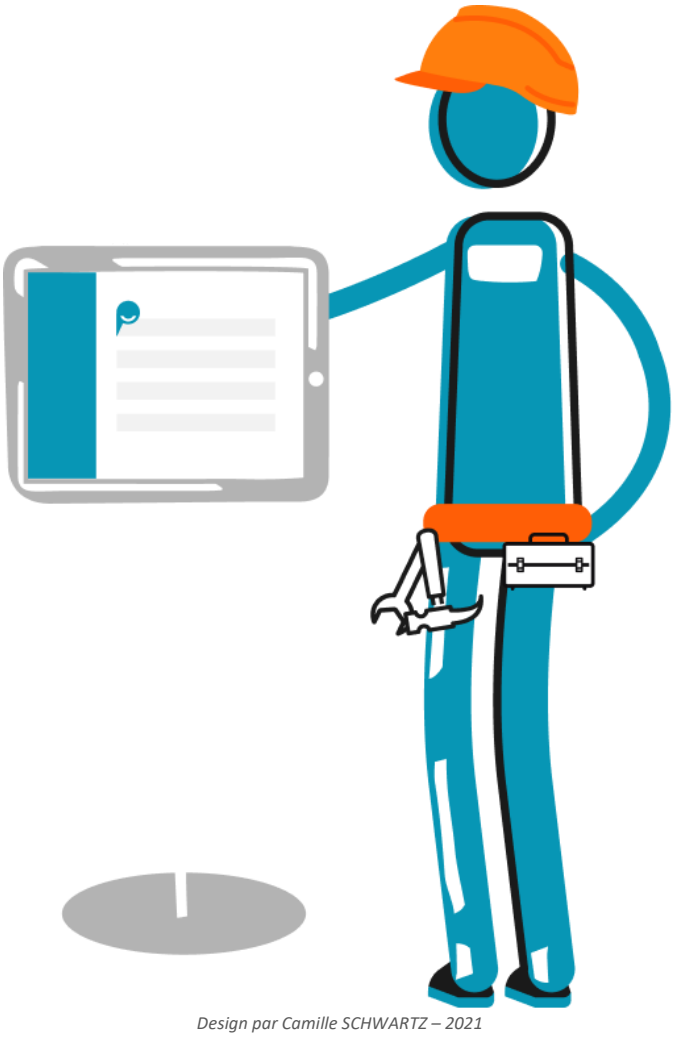
Dotspot Free

DOSSIER DE PROJET

François BANITZ

Titre professionnel développeur web et web mobile

25/10/2021



Design par Camille SCHWARTZ – 2021

InterFacile
CODÉVELOPPONS VOTRE FUTUR

WEBFORCE
BE THE CHANGE

I. Résumé

Dostpot free permet de créer et de visualiser des procédures industrielles. Celles-ci ont pour vocation de guider l'utilisateur pour l'utilisation ou l'installation d'une machine.

Dans ce projet, il s'agit de développer une application attractive, accessible au plus grand nombre et facile à maintenir.

Nous avons comparé et testé plusieurs solutions pour la base de données (Firebase, Floor et Hive) et pour l'interface graphique (Bubble, FlutterFlow et Flutter). Ces solutions (sauf bubble), sont intégrées à Dart, un langage compilé, multiplateforme, open-source, développé par Google.

Hive s'est montré plus polyvalent et flexible que les autres. Il fonctionne sur toutes les plateformes (Android, iOS, web, Windows, Linux et Mac). Il est plus rapide en lecture et écriture, puisque c'est une base de données NoSQL.

Flutter permet plus de personnalisation graphique par rapport à ses concurrents NoCode (FlutterFlow et Bubble).

Nous avons donc utilisé Flutter avec Hive avec un résultat final conforme aux attentes.

Table des matières

II. Contexte du projet

1. L'entreprise InterFacile et ses objectifs

a. Avant 2021

Interfacile est une société par actions simplifiée unipersonnelle (SASU) qui a pour vocation de proposer des outils logiciels pour des entreprises industrielles. Gregory JAQUET en est le président.

Dotspot est l'application phare d'InterFacile. Elle a pour vocation de permettre aux acteurs de terrain d'une entreprise industrielle de produire et gérer facilement des documents opérationnels, en un seul endroit, notamment des procédures techniques et modes opératoires. Cela permet de capitaliser les connaissances métier, ce qui est particulièrement utile lors de l'arrivée d'un nouvel ouvrier ou après l'arrêt prolongé de l'utilisation d'une machine.

Le client principal d'InterFacile pour Dotspot a pour activité d'assembler des machines pour la production textile et est reconnu internationalement.

La première version de Dotspot a été développée pour InterFacile par une entreprise informatique externe. Elle utilisait Angular, un framework front-end permettant une construction « single page » d'applications web et, par la même, de fluidifier l'expérience utilisateur en supprimant le chargement d'une page à chaque action. Elle fonctionne également avec Ionic, un framework qui utilise un autre framework web, Angular, React, Vue ou Javascript et pouvant les transpiler en application Android ou iOS.

Le Backend de l'application a été réalisé en Java.

b. A partir de 2021

En 2021, InterFacile a souhaité devenir autonome pour la maintenance et l'évolution de cette application et donc de se séparer de l'équipe externe.

Interfacile a alors recruté des alternants, Camille SCHWARTZ pour le design et moi-même, François BANITZ, pour le développement.

Cela a permis à l'entreprise de se diversifier, en proposant la conception d'applications spécifiques, codéveloppées avec des industriels.

Depuis peu, InterFacile a fait évoluer son système économique et propose à présent une offre dite « codéveloppement ». Cette offre donne accès à ces avantages :

- L'accès à Dotspot
- Le codéveloppement d'une application répondant à un besoin spécifique de l'entreprise
- Le panel d'applications déjà développé avec les avec les autres entreprises clientes
- La possibilité de faire évoluer ces dernières, ainsi que Dotspot pour s'approcher au plus près de ses besoins.

Le codéveloppement est le fait de développer un projet en collaboration directe avec le client en l'intégrant directement et régulièrement à la discussion initiale et lors de l'avancement du projet. Cela permet une meilleure communication des réels besoins, de mettre un point

d'honneur sur la confiance et la solidarité et de responsabiliser chaque acteur car il est directement force de proposition.

2. Genèse du projet

Pour qu'Interfacile devienne autonome, il était nécessaire de changer la technologie sur laquelle Dotspot est basée. Angular et Ionic, utilisant une grande diversité de langages, requièrent en effet des compétences poussées en développement et rendent l'application difficile à mettre à jour. De plus, Java, le langage utilisé pour le back-end, est difficile à maintenir à cause de la difficulté à recruter des développeurs Java selon le responsable de l'entreprise.

Grégory Jaquet, a pris en compte les différentes technologies disponibles pour cette refonte. Préférant les outils no-code (sans ligne de code) pour leur capacité à effectuer un développement rapide et ne nécessitant pas ou peu de compétences de programmation, son choix s'est d'abord tourné vers Bubble.io.

Cependant, Bubble présentait des inconvénients notables. Les applications créées sur cette plateforme y étaient hébergées sans possibilité de télécharger le code, reposant à nouveau le problème de l'autonomie. De plus, les services de la plateforme commençaient à gagner en popularité et devenaient de plus en plus cher. Enfin, Bubble, comme beaucoup d'outils no-code manquait de fonctionnalités et n'était pas adapté à un projet tel que Dotspot.

Le choix de technologie n'ayant pas encore été décidé, j'ai suivi une formation de l'outil Bubble pour être opérationnel dès mon premier projet : le développement d'une application spécifique pour notre client principal. Leur besoin était de pouvoir faire l'inventaire de leur visserie, pour la gestion des stocks.

Le client utilise toujours cette application à ce jour, elle fera prochainement l'objet d'une refonte sous Flutter.

Plus tard, le choix fut finalement porté sur un développement hybride avec Flutter, un Software Development Kit (SDK), dont nous parlerons dans la partie Technologies utilisées, et FlutterFlow.

3. L'Equipe

Gregory Jaquet est le CEO d'InterFacile. C'est le project owner. Il s'est occupé de la gestion du projet, de la plupart des décisions et de la gestion de l'équipe.

Camille Schwartz est la designeuse d'InterFacile. Elle a conçu le design, les maquettes, les logos et les illustrations.

Je suis le développeur du projet. J'ai réalisé le développement no-code de l'interface, le développement du fonctionnement Back-end et la mise en place de la base de données.

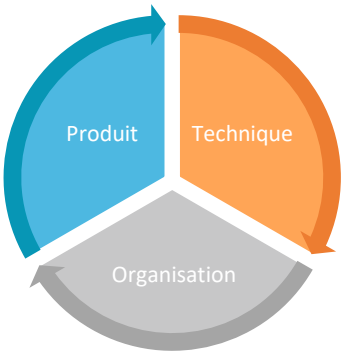
4. Méthode agile

La méthode Agile repose sur quatre valeurs et douze principes[1]

[1] Source : https://agilemanifesto.org/iso/fr/manifesto.html

Selon le Manifeste pour le développement Agile de logiciels, rédigé en 2001 par un collectif d'experts en innovation informatique, quatre valeurs doivent primer dans la conduite d'un projet :

- Les individus et leurs interactions plus que les processus et les outils
- Des logiciels opérationnels plus qu'une documentation exhaustive
- La collaboration avec les clients plus que la négociation contractuelle
- L'adaptation au changement plus que le suivi d'un plan



InterFacile partage toutes ces valeurs et les met en pratique dans la réalisation de ces projets et dans ses interactions avec ces clients. Ainsi le tableau ci-dessous montre, à travers quelques exemples non exhaustifs, qu'InterFacile est une entreprise Agile.

Principe de la méthode agile	Ce qu'InterFacile a mis en place
Les utilisateurs ou leurs représentants et les développeurs doivent travailler ensemble quotidiennement tout au long du projet.	InterFacile travaille en codéveloppement avec ses clients. Cela prend en compte des réunions régulières pour se mettre d'accord sur l'avancée du projet et des tests réalisés au fur et à mesure par le client. Celui-ci est pleinement acteur du projet. Les changements et améliorations en cours de projet sont les bienvenus.
Accueillir chaleureusement les changements de besoins, même tardifs dans le développement. Les processus agiles tirent parti du changement pour renforcer l'avantage concurrentiel du client	
Un logiciel opérationnel est la principale mesure d'avancement.	Les démonstrations régulières de l'applications sont le support principal des réunions.
La simplicité – c'est-à-dire l'art de minimiser la quantité de travail inutile – est essentielle.	La recherche de simplicité est une ligne directrice majeure de l'entreprise. Exemple : choix du NoCode quand c'est possible.
Réalisez les projets avec des personnes motivées. Fournissez-leur l'environnement et le soutien dont ils ont besoin et faites-leur confiance pour atteindre les objectifs fixés.	Le chef d'entreprise fait confiance à ses collaborateurs, bien qu'ils soient peu expérimentés. Il s'appuie sur leur motivation et stimule leur prise d'initiatives et propositions. Il leur laisse une grande autonomie, tout en étant très présent au quotidien.

Interfacile utilise des outils de SCRUM, l'un des premiers frameworks de la méthode agile. Son créateur a participé à la rédaction du Manifeste pour le développement Agile de logiciels.

III. Cahier des charges

1. Origine du projet

Le présent projet, Dotspot Free s'inscrit dans une volonté de refonte. Il marque une étape intermédiaire entre l'ancienne et la nouvelle version de Dotspot.

Dotspot fonctionne seulement en ligne, puisqu'il repose sur une base de données distante.

Dotspot Free est une version locale de Dotspot et sera mise gratuitement à la disposition du public.

L'application enregistre uniquement les données directement dans le stockage local de l'appareil et par conséquent, il ne sera pas possible de les partager dans sa première version.

Dotspot 2.0 sera disponible en ligne et hors-ligne. La version hors-ligne restera gratuite.

2. Définition de la cible et objectifs

Dotspot Free s'adresse aux industriels mais aussi aux bricoleurs amateurs, qui auraient besoin d'un outil pour structurer leurs projets et standardiser leur façon de travailler. Il leur permettra de mettre en place des procédures pour rafraîchir leurs acquis.

Pour l'entreprise, Dotspot Free permettra de faire connaitre la future application Dotspot, qui sera plus riche en termes de fonctionnalités, et InterFacile de manière générale.

3. Périmètre

A sa sortie, Dotspot Free sera disponible en français et donc visera une clientèle francophone mais il est prévu plus tard qu'il soit également disponible en anglais.

Dotspot Free doit être disponible sur Android et iOS.

4. Design

L'application devra respecter la charte graphique d'Interfacile, tout en respectant au maximum celle des plateformes sur lesquelles elle est disponible.

Charte graphique d'Interfacile:

Les couleurs principales :

- Bleu (#4DB8E1)
- Orange (#FFA556)

Les logos :

InterFacile
CODÉVELOPPONS VOTRE FUTUR

dotspot

Certaines illustrations stylisées :

L'application doit être uniquement en dark mode (mode sombre) pour prévenir la fatigue oculaire.

Le projet devra suivre la maquette figurant en **ANNEXE 1**.

5. Description fonctionnelle des besoins

Fonction principale : Créer et visualiser des procédures industrielles.

Sous-fonctions :

- Ajouter un document (procédure) avec un titre et éventuellement une photo.
- Visualiser l'ensemble des documents créés.
- Modifier le titre et la photo d'un document
- Supprimer un document et les éléments qui lui sont liés
- Visualiser le contenu d'un document et la liste des éléments qu'il contient.
- Ajouter, dans un document, un élément contenant du texte et éventuellement une photo et dont la couleur et le symbole peuvent être choisis selon une typologie codifiée.
- Modifier un élément dans un document
- Supprimer un élément d'un document

Fonctions annexes :

- Réorganiser les éléments dans un document
- Afficher les dates de création des documents et éléments

6. Délai

Le développement de la première version de l'application doit commencer septembre 2020. Sa livraison est prévue avant le 4 octobre 2021 pour une conférence dans le cadre du Master spécialisé de l'ECAM (Ecole Catholique d'Arts et Métiers) de Strasbourg.

IV. Technologies utilisées

1. Front-end

Pour faire l'intégration initiale, j'ai utilisé FlutterFlow. Ensuite j'ai exporté et modifié le code pour faire les modifications plus fines.

2. Back-end

Pour le back-end, j'ai utilisé flutter, un Software Development Kit (SDK).
Flutter utilise Dart, un langage unique pour l'interface graphique et le back-end.

3. Base de données

Pour la base de données j'ai utilisé Hive, un package de Dart qui permet de stocker des données en local dans une base de données NoSQL

V. Front-end

1. Design

Un extrait des maquettes réalisées par Camille SCHWARTZ est disponible en **ANNEXE 1**

2. FlutterFlow

FlutterFlow est une plateforme no-code permettant d'exporter le code sous Flutter et que nous utilisons pour la partie front-end.

J'ai utilisé FlutterFlow pour l'intégration initiale puis j'ai exporté le code pour développer le back-end directement en code.

Voici ci-dessous, une capture d'écran de l'interface de FlutterFlow :

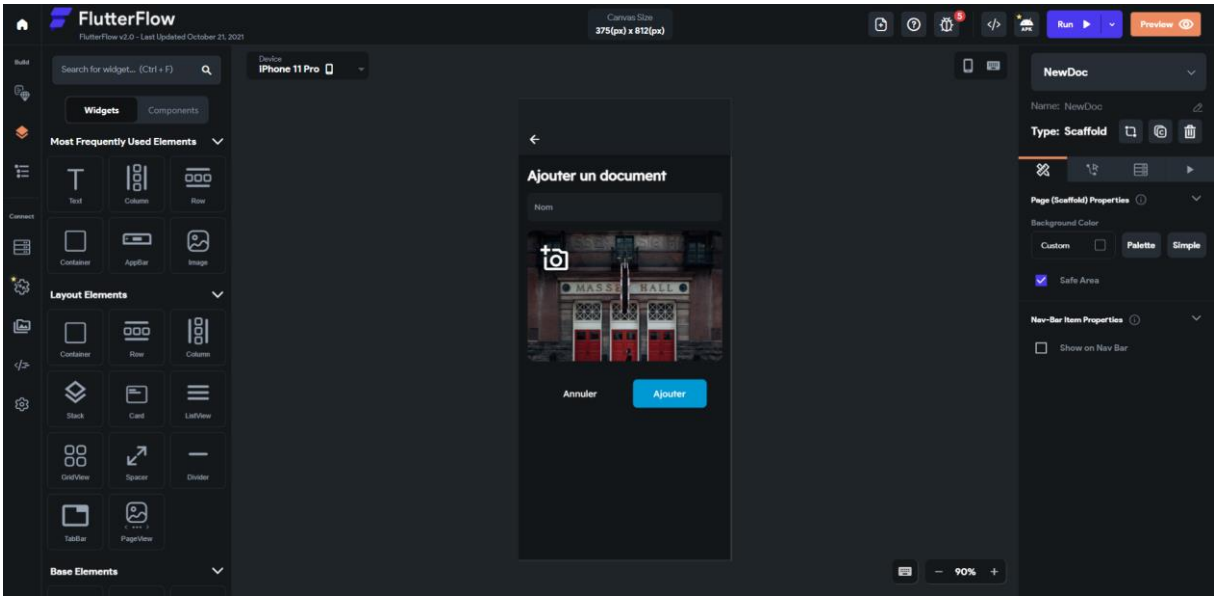


Figure 1 : Capture d'écran de l'interface de FlutterFlow

Plus tard nous avons eu besoin de retravailler le front-end pour répondre à de nouveaux besoins. Pour ce faire, j'ai codé directement sans passer par FlutterFlow en utilisant Flutter.

3. Flutter

J'ai utilisé flutter pour le front end et le back-end. Flutter utilise Dart, un langage unique pour l'interface graphique et le back-end, permettant une mise en place, une révision et une mise à jour des composants simplifiées.

Il permet de compiler le code vers des plateformes mobiles (Android, iOS), web, et maintenant en bêta, des plateformes de bureau (Windows, Linux et MacOS).

Le code est compilé en langage machine, ce qui permet de meilleures performances.

La mise en page de Flutter fonctionne sous forme de widget, un peu à la manière des balises de l'HTML. Cependant, Flutter est beaucoup moins permissif et doit suivre un certain nombre de règles pour fonctionner.

Par exemple, la plupart des widgets peut contenir un unique widget enfant, tandis que certains peuvent en contenir plusieurs. Dans cette seconde catégorie, on retrouve notamment la Column et la

Row, comparable à « flex-direction » de FlexBox. La Column permet de disposer ses widgets sur l'axe vertical (« Main Axis ») tandis que la Row les disposent sur l'axe horizontal (« Cross Axis »).

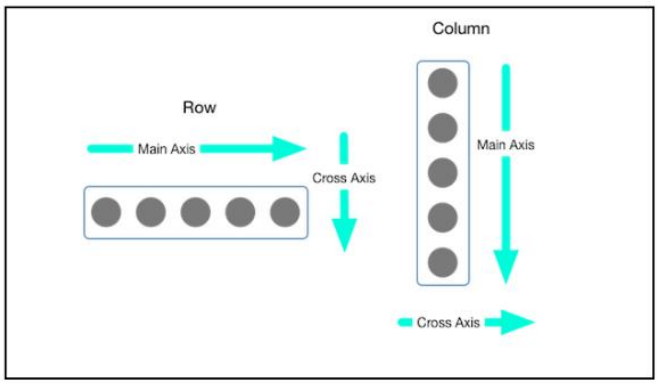


Figure 2 : placement des widgets dans une Column ou une Row[2]

Dans une Row ou Column, comme avec FlexBox on peut positionner les enfants dans la direction du parent, en utilisant la propriété mainAxisAlignement de Flutter.

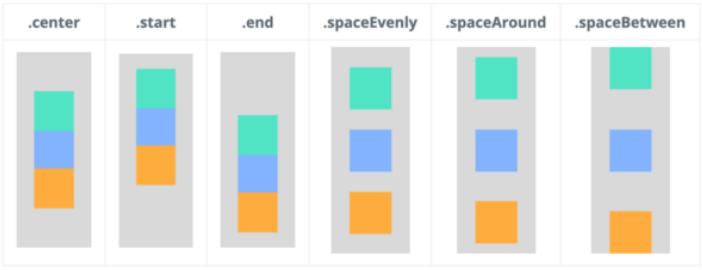


Figure 3 : Propriété mainAxisAlignement dans une Column[3]

Ces fonctionnements spécifiques, différent du développement web classique, ont été un peu déroutants au début mais rapidement surmontés, notamment grâce à mes connaissances antérieures sur FlexBox, acquises lors de ma formation.

[2] Source : arzerin.com
[3] Source : arzerin.com

Voici un exemple d'une application minimale qui utilise les principes de Row et de Column.

Code

```
class MyHomePage extends StatelessWidget {
  final String _title;

  MyHomePage(this._title);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text(_title),),
      body: Column(
        children: [
          Row(
            mainAxisAlignment:  MainAxisAlignment.center,
            children: [
              iconHeart,
              Text(_title),
              iconHeart,
            ],
          ),
          iconAndroid,

          Text("Build for Android"),
        ],
      ),
      floatingActionButton: FloatingActionButton(onPressed: () {},),
    );
  }
}
```

Visuel

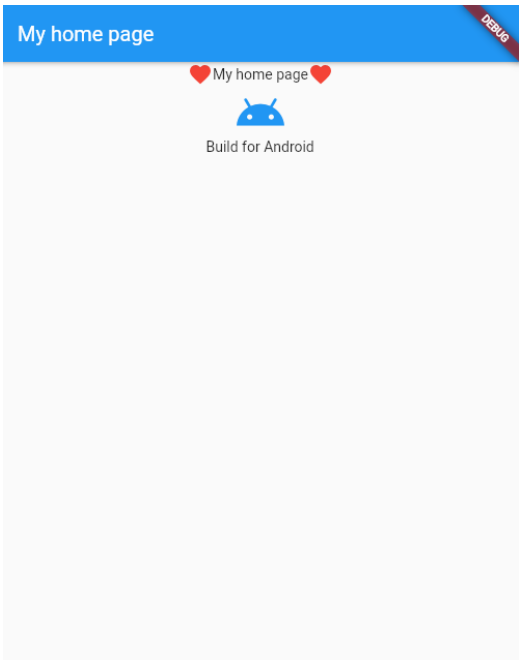


Figure 4 : Exemple de fonctionnement d'une Row et d'une Column à partir d'un exercice que j'ai réalisé

Dans cet exemple, l'application contient une Column avec à l'intérieur une Row, une icône d'androïde et un texte « Build for Android ».

La Row contient une icône de cœur, un texte « My home page » et encore une icône de cœur.

Pour la Row, j'ai utilisé la propriété MainAxisAlignement.center. Ainsi, les éléments sont collés au centre.

VI. Back-end

1. Diagramme initial de la base de données

Le diagramme de la base de données décrit les colonnes des deux tables (Document et Block) ainsi que leur type et à quoi ils servent.
Ce diagramme a été effectué avant le projet, il y a eu donc des évolutions depuis.

Document	Type	Description
uid	ID	Suite de caractères permettant d'identifier le document
blocs	Relation	Permet de connaître les blocs contenus dans le document
owner	Relation	Permet de connaître le propriétaire du document
title	Text	Le titre du document
tags	List of Text	Permet de connaître les tags du document
created_time	Timestamp	La date et l'heure de création du document
photo_url	Image Path	Une image de couverture pour le document

Bloc	Type	Description
uid	ID	Suite de caractères permettant d'identifier le bloc
document	Relation	Permet de connaître le document auquel est lié ce bloc
text	Text	Texte du bloc
video_url	Video Path	Url d'une vidéo liée au bloc
photo_url	Image Path	Url d'une image liée au bloc
audio_url	Audio Path	Url d'un enregistrement audio lié au bloc
warnig	Boolean	Permet d'afficher le bloc comme nécéssitant une attention particulière
info	Boolean	Permet d'afficher le bloc comme porteur d'une information importante
sucess	Boolean	Permet d'afficher le bloc comme une bonne pratique à suivre
danger	Boolean	Permet d'afficher le bloc comme une mauvaise pratique à éviter

Figure 5 : Diagramme de la base de données

2. Technologies de la base de données

Les données de Dotspot Free sont stockées en local dans l'appareil, dans une base de données NoSQL.

Hive permet de gérer celles-ci, nous parlerons de ce package plus en détail au **paragraphe c**. L'accès aux données se fait via un Data Acces Object (DAO) géré par Hive appelé Boxes. Il s'agit d'une classe qui contient toutes les méthodes permettant l'ajout, la lecture, la modification et la suppression de celles-ci.

a. Firebase

FlutterFlow utilise Firebase comme base de données. Il n'y a presque rien à configurer sur Firebase lui-même, tout se fait directement sur FutterFlow.

Comme pour une base de données SQL, on peut définir les champs et leur type pour chaque table, sur l'image ci-dessous, la table « Blocs ».

Contrairement au SQL, le champ permettant d'identifier une entrée n'est pas un id (int) mais un uid (string), qui sera géré automatiquement par Firebase.

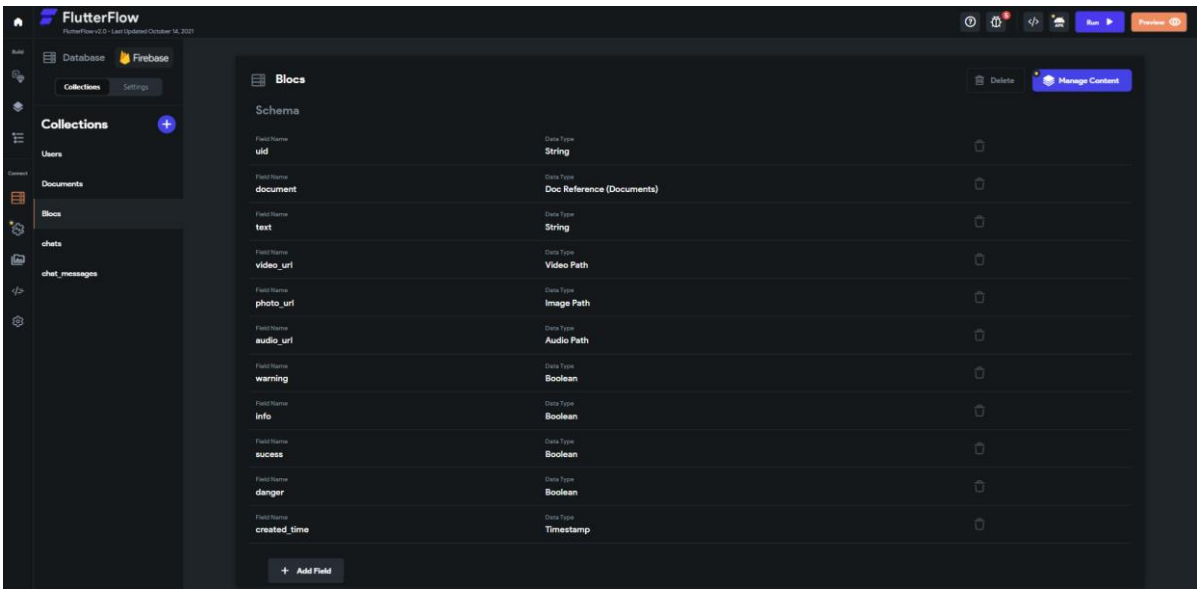


Figure 6 : Capture d'écran de l'interface de FlutterFlow pour la base de données

Cependant, on ne peut pas utiliser une autre base de données que Firebase. Si l'on veut changer de base de données, il faut exporter le code et le modifier.

Pendant le développement de Dotspot Free nous nous sommes rendu compte que FlutterFlow ne serait pas suffisant pour certaines fonctionnalités et qu'il fallait passer sur un développement plus classique, directement en Flutter. Notamment, la base de données sous Firebase était entièrement en ligne, alors que nous voulions, pour cette version, un fonctionnement hors ligne.

b. Floor

Afin de combler les lacunes de Firebase, j'ai modifié la base de données et l'intégralité du back-end pour fonctionner avec **Floor**, un package Dart permettant d'utiliser une base de données stockée en local. Floor permet d'interagir avec une base de données SQLite en fournissant une abstraction objet.

Floor mappe les objets sur la base de données. Cela permet un contrôle total de la base de données en utilisant du SQL.

Voici ci-dessous un exemple d'un DAO (Data Access Object), développé lorsque le projet utilisait encore Floor pour la base de données. Les méthodes du DAO utilisent des annotations qui indiquent le type de la requête SQL. Ainsi, "insertDocument()" utilise l'annotation @insert, qui permet d'insérer une ligne dans la base de données. Mais la plupart utilisent "@Query", une annotation contenant directement des instructions SQL simples.

```dart
import 'package:floor/floor.dart';
// floor documentation :
// https://floor.codes/getting-started/

@dao
abstract class DocumentDao {
  @Query('SELECT * FROM document')
  Future<List<Document>> findAllDocuments();

  @Query('SELECT * FROM document')
  Stream<List<Document>> findAllDocumentsAsStream();

  @Query('SELECT * FROM Document WHERE id = :id')
  Stream<Document?> findDocumentById(int id);

  @insert
  Future<void> insertDocument(Document document);

  @Query('DELETE FROM document WHERE id = :id')
  Future<void> deleteDocument(int id);
}
```

Figure 7.1 : DAO de Floor

Pour utiliser cette classe, on doit d'abord l'instancier puis par exemple appeler la méthode insertDocument :

```dart
documentDao = DocumentDao();
documentDao.insertDocument(new Document());
```

Figure 7.2 : utilisation de la DAO de Floor

c. Hive

Début octobre 2021, nous nous sommes rendu compte que Floor ne répondait pas à nos besoins. En effet, il fonctionne sur Android et iOS mais nous avions aussi prévu pour la suite de pouvoir utiliser Dotspot sur web, ce que ne permet pas Floor. J'ai donc réeffectué une migration, cette fois vers Hive, un autre package de base de données locale NoSQL.

3. Hive

a. Description

Hive est un package et une base de données locale NoSQL. Il fonctionne sur les plateformes Android, iOS, web, Windows, Linux, et Mac.

Comme le montre le benchmark Hive est bien plus rapide que la plupart des autres bases de données et dispose d'un cryptage intégré. Pour ce projet nous n'avons pas utilisé le cryptage.

Hive, comme Floor, est un Object-Relational Mapping (ORM). Un ORM permet de simuler une base de données orientée objet en mappant une classe sur la base de données. Nous donnons un exemple dans le chapitre **Diagramme de classes**.

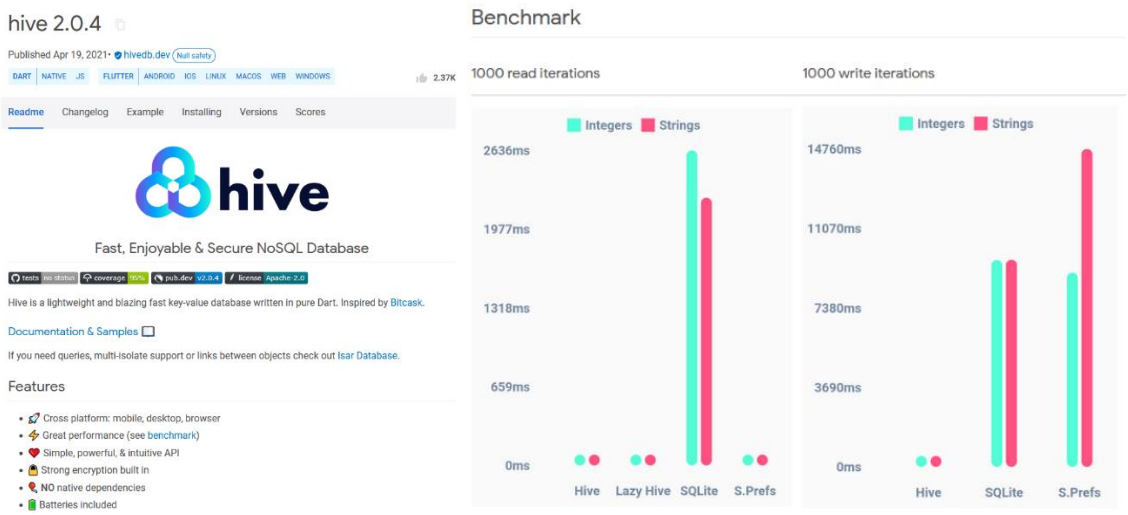


Figure 8: Extraits de la page pub.dev de hive
Source : https://pub.dev/packages/hive

b. NoSQL

Le NoSQL (Not Only SQL) est un type de base de données, dont l'utilisation à grande échelle est relativement récente (début du 21ième siècle). Dans une telle base de données, on s'affranchit du modèle relationnel. Une base de données NoSQL est caractérisée par sa simplicité, sa capacité à s'étendre (scalability) et sa rapidité. Les données sont souvent stockées sous la forme de clé-valeur, clé-colonne ou clé-document.

Hive utilise une base de données clé valeur. Cela signifie qu'il associe une clé unique à une valeur donnée comme un dictionnaire/map. Ce modèle nécessite un développement côté application plus important qu'avec une base de données SQL.

c. Les Box

Hive a un fonctionnement basé sur des « box ». Toutes les données sont stockées dans ces Box. Si l'on veut ouvrir une nouvelle box, on peut par exemple utiliser le code suivant :

```
var box = await Hive.openBox('testBox');
```

Figure 9 : Ouverture d'une Box
Source : https://docs.hivedb.dev/

Après avoir ouvert une box, il est possible d'y insérer n'importe quel type de données

Code

```dart
import 'package:hive/hive.dart';

void main() async {
  var box = await Hive.openBox('person');


  String name = 'David';

  box.put('person', name);

  print('Name: ${box.get('person')}');


  int age = 22;

  box.put('person', age);

  print('Age: ${box.get('person')}');
}
```

Console

```
TERMINAL  PROBLEMS  OUTPUT  DEBUG CONSOLE
Launching box.dart on Chrome in debug m
This app is linked to the debug service
Debug service listening on ws://127.0.0
 Running with sound null safety
Connecting to VM Service at ws://127.0.
Name: David
Age: 22
```

Figure 10 : utilisation d'une box
Source : https://docs.hivedb.dev/

Cependant, bien qu'insérer n'importe quel type de valeur dans une box peut être un avantage certain, pour éviter les crashs il est préférable de lui donner un type de valeur déterminé. Pour ce faire, lorsque l'on va ouvrir la box, on va lui attribuer un type entre chevrons (<>).

Code

```dart
import 'package:hive/hive.dart';

void main() async {
  var box = await Hive.openBox<String>('person');


  String name = 'David';

  box.put('person', name); // OK

  print('Name: ${box.get('person')}');


  int age = 22;

  box.put('person', age); // ERROR

  print('Age: ${box.get('person')}');
}
```

Console

```
TERMINAL  PROBLEMS 1  OUTPUT  DEBUG CONSOLE
Launching box.dart on Chrome in debug mode...
org-dartlang-app:/box.dart:18:18: Error: The argument type 'int' can't be assigned to the parameter type 'String'
  box.put('age', age); // Error
                 ^
Failed to compile application.
Exited (sigterm)
```

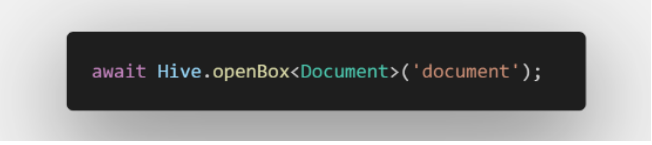
Figure 11 : Utilisation de l'attribution d'un type à une box

Il est aussi possible de lui assigner une classe comme type. Cependant celle-ci doit être formatée d'une certaine manière pour que Hive puisse générer l'ORM correspondant.

```dart
import 'package:hive/hive.dart';

part 'document.g.dart';

@HiveType(typeId: 0)
class Document {
  /// La date de création du [Document]
  @HiveField(0)
  DateTime createdTime = DateTime.now();

  /// Le titre du [Document]
  /// Ne sera pas affiché si égal à `""`
  @HiveField(1)
  late String title;

  /// Le titre du [Document]
  /// Ne sera pas affiché si égal à `""`
  @HiveField(2)
  String photoPath = "";
}
```

Figure 12.1 : Définition de la classe Document avec Hive

Après l'avoir défini, j'ouvre ma box en lui donnant le type « Document ».



```dart
await Hive.openBox<Document>('document');
```

Figure 12.2 : Ouverture d'une Box en lui attribuant la classe Document

d. Le DAO

L'accès aux données se fait via un Data Acces Object (DAO) géré par Hive appelé Boxes. Floor imposait également le DAO. Ce dernier rend l'accès à la base de données indépendant du schéma de la base de données, notamment en rassemblant les méthodes d'accès dans une seule classe. Ces méthodes sont génériques. Cela a permis une migration plus simple de Floor vers Hive.

En **ANNEXE 2**, l'extrait de code de Boxes, le DAO de Hive.

4. Diagramme de classes

Le diagramme de classes donne une vue d'ensemble sur les classes que j'ai codé pour le projet.

La classe Document permet de mapper, grâce à Hive, notre objet Document sur la base de données. C'est le modèle des documents que l'on retrouvera dans l'application.

La classe Block permet également de mapper notre objet Block sur la base de données. C'est le modèle des éléments que l'on retrouvera dans l'application

La classe Boxes est le DAO du projet.

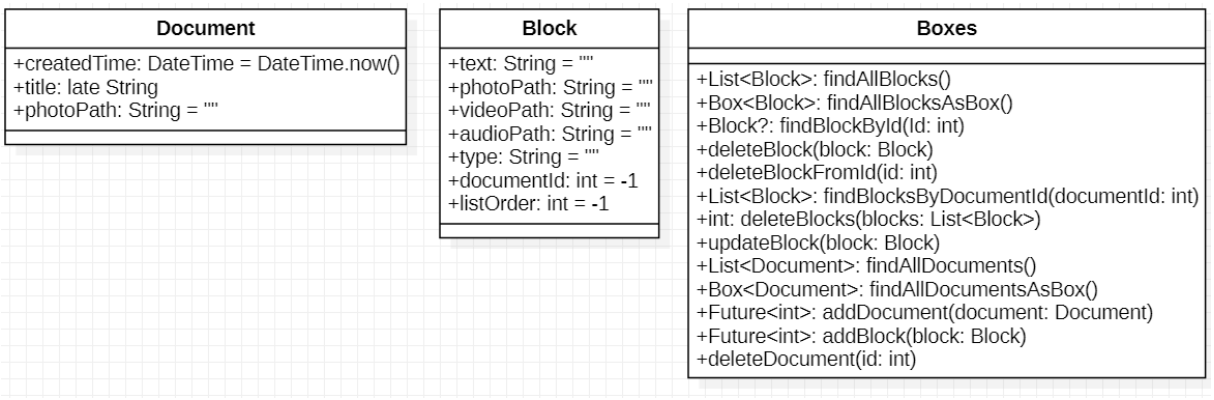


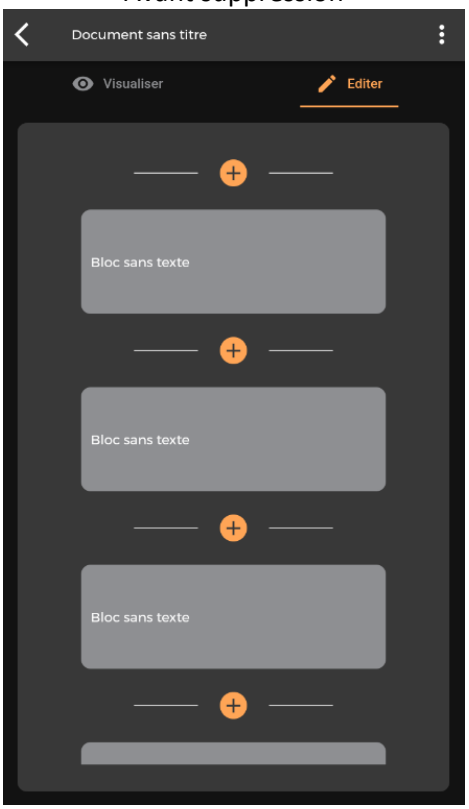
Figure 13 : Diagramme de classes

En ANNEXE 2, l'extrait de code qui décrit Boxes dans le projet.

VII. Tests

Tout au long du développement, j'ai effectué des tests unitaires pour vérifier au fur et à mesure le bon fonctionnement de chaque nouvelle fonctionnalité. Par exemple, lors d'un test de la fonctionnalité « supprimer un Block », j'ai remarqué que les blocks ne se supprimaient pas correctement, laissant derrière eux une trace de leur présence.

Avant suppression



Après suppression

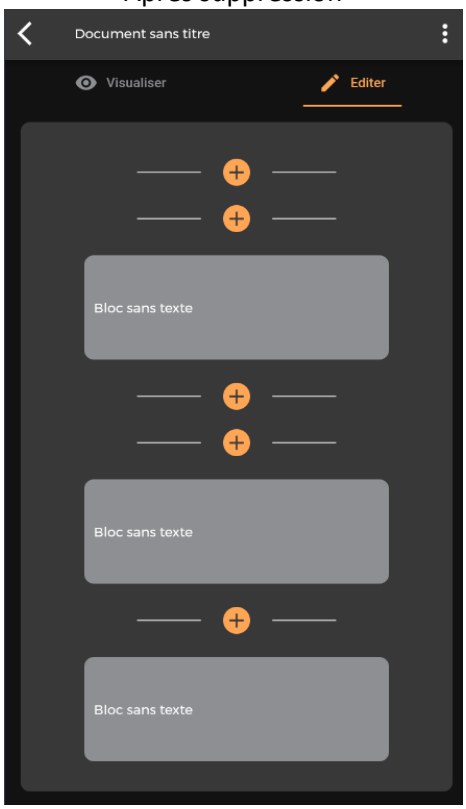


Figure 14 : Captures d'écrans montrant un problème lors de la suppression de Block

J'ai découvert que ce problème venait du fonctionnement de la récupération de données dans Floor. En effet, les blocks étaient chargés depuis la base de données dans une variable « blocks » lors de l'affichage de la page, mais lors de la suppression dans la BDD, ils n'étaient pas supprimés dans la variable « blocks », ce qui causait cette erreur d'affichage.

Pour remédier à ça, j'ai modifié le code pour qu'il supprime en même temps les blocks, non seulement dans la base de données mais aussi dans la variable « blocks ».

```
354  354        child: child,
355  355        direction: DismissDirection.endToStart,
356  356          onDismissed: (_void){
357       -        // blocks.remove(block);
358       -        blockDao.deleteBlock(block.id!);
     357  +        setState(() {
     358  +          blocks.remove(block);
     359  +          blockDao.deleteBlock(block.id!);
     360  +        });
359  361        },
```

Figure 15 : Capture d'écran de GitHub Desktop de la modification effectuée

J’ai ensuite réalisé des tests globaux de l’application pour vérifier la cohérence d’ensemble.

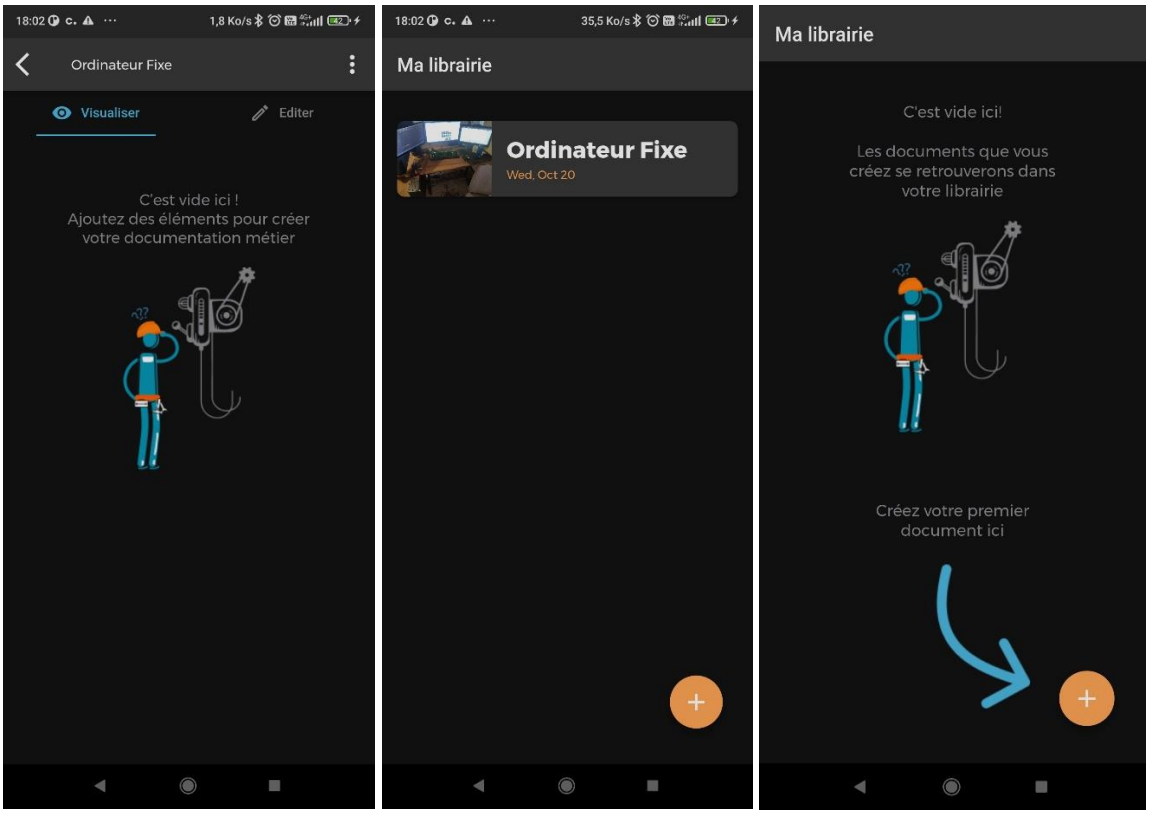
Puis j’ai aussi sollicité les autres membres de l’équipe pour effectuer d’autres tests globaux pour avoir un point de vue externe.

Enfin, nous avons fourni régulièrement une version testable au client qui nous faisait des retours rapides.

Bien que ces tests fonctionnent et permettent de trouver des dysfonctionnements, ils peuvent encore être grandement améliorés pour être plus efficaces. Notamment, flutter propose un package, flutter_test, pour simplifier le processus de tests. Je ne m’en suis rendu compte qu’à la fin du projet.

VIII. Rendu final

Voici à quoi ressemble l'application au 4 octobre 2021:



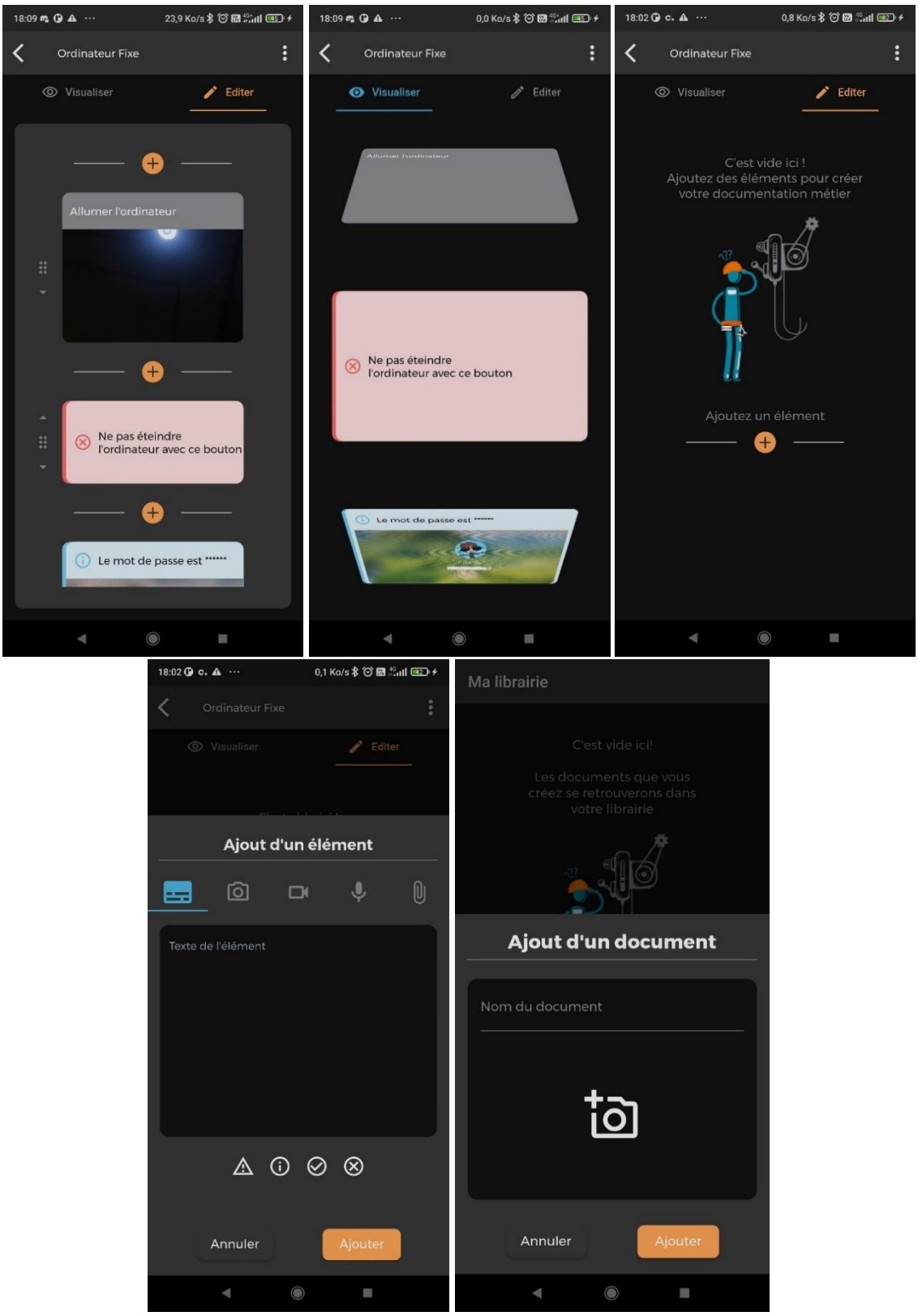


Figure 16 : Captures d'écrans de l'application finale

Le rendu final est proche de ce qui était prévu par la maquette. On note cependant quelques différences :

- Sur la cinquième capture, il a été décidé d'ajouter un affichage spécifique pour les éléments pour mieux visualiser celui qui nous intéresse.
- Certaines couleurs de fond ont été modifiées pour unifier les champs de formulaire.

L'application s'adapte relativement à la taille de l'écran.

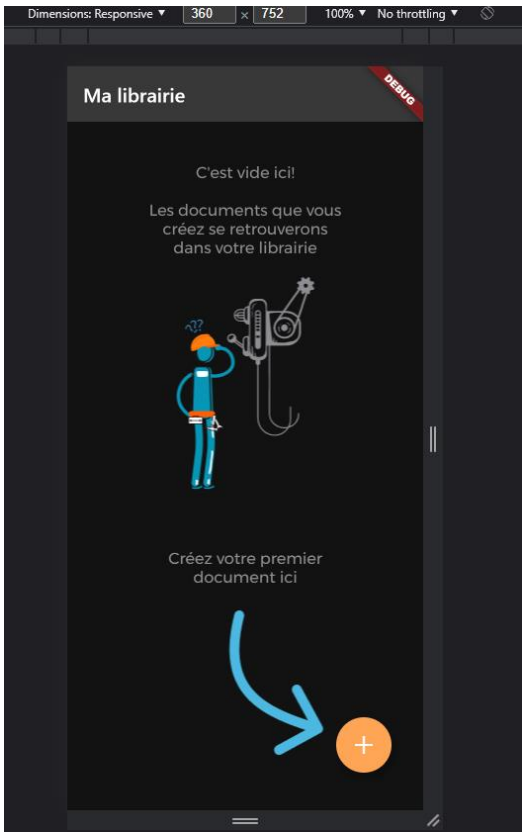


Figure 17.1 : Capture d'écran démontrant le comportement de l'application en fonction de la taille du smartphone

Cependant pour les écrans plus petits en hauteur, l'affichage est un peu moins agréable, mais toujours tout à fait fonctionnel.

Figure 17.2 : Capture d'écran démontrant le comportement de l'application en fonction de la taille du smartphone

IX. Mise à disposition du public

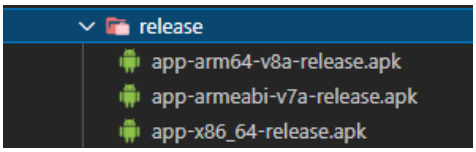
L'application a été mise à disposition d'étudiants lors d'une conférence dans le cadre du Master spécialisé de l'ECAM (Ecole Catholique d'Arts et Métiers) de Strasbourg le 4 octobre 2021.

Elle est cependant seulement disponible sur Android.

Pour la compiler, j'ai utilisé la fonction « build APK » intégrée à Flutter.

```
flutter build apk --split-per-abi
```

L'option --split-per-abi permet de générer l'APK pour les différentes architectures de processeur. Les fichiers générés se présentent ainsi :



Pour installer l'APK sur le téléphone, il suffit de l'ouvrir pour l'installer.

Bientôt, l'application sera disponible sur les stores Play Store et Apple Store.

X. Conclusion et perspectives

1. Conclusion

L'objectif du projet a été pleinement atteint. En effet, Dotspot free est fonctionnel, il correspond au cahier des charges. Ce projet a été l'occasion de mettre en œuvre une organisation d'équipe Agile en interne et avec le client et a fait la démonstration de la capacité d'InterFacile à être autonome dans le développement de ses applications. L'application servira de tremplin dans le développement de Dotspot 2.0 et constitue un Proof Of Concept (POC) de celui-ci.

En ce qui me concerne, ce projet a été une excellente opportunité de découvrir de nombreux outils et langages et de travailler en méthode Agile.

2. Amélioration possible

Au cours de ce projet, les outils utilisés ont beaucoup changé, ce qui l'a ralenti. Pour la suite, nous devrons mieux anticiper le choix des outils, en prenant en compte, en amont, leurs avantages et leurs limites respectifs. Ceci sera facilité par une meilleure connaissance de ces outils, acquise dans ce projet.

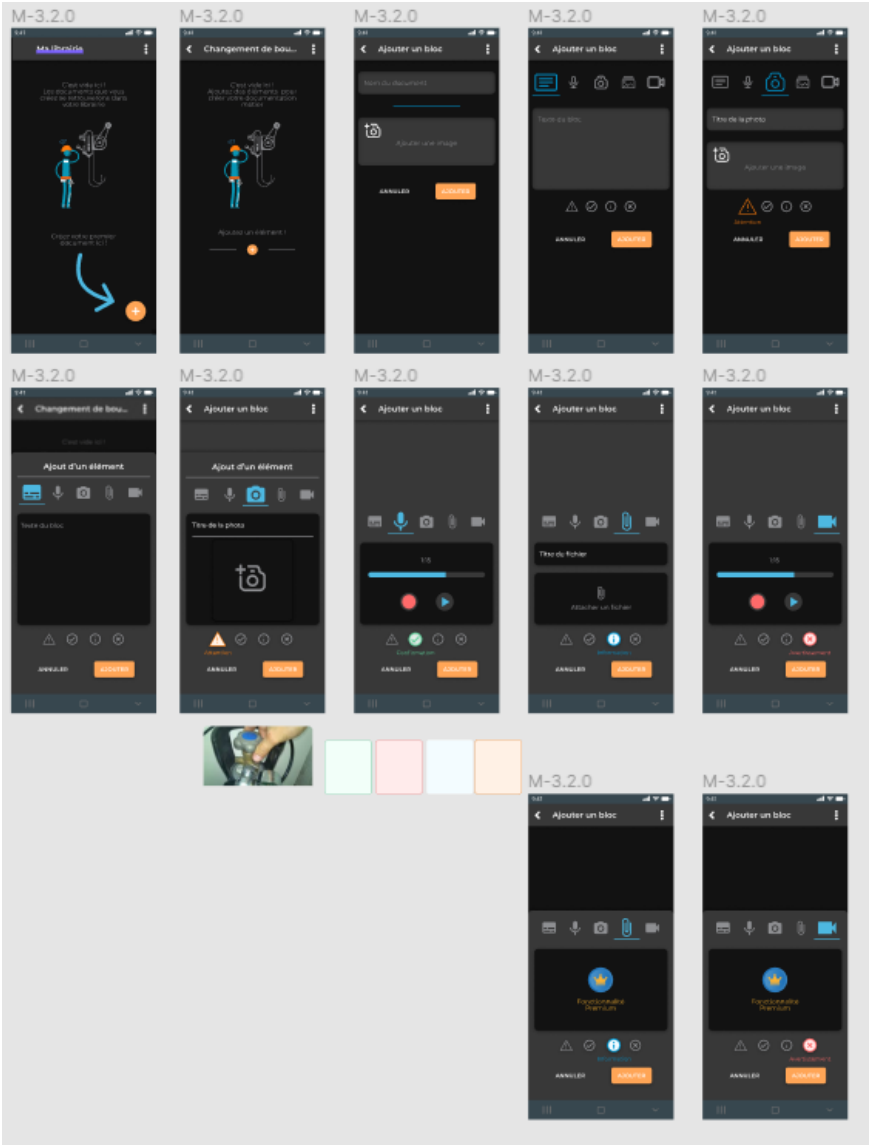
3. Suite du projet

Lorsque nous passerons du développement de Dotspot free à Dotspot 2.0, nous prévoyons de garder Hive mais d'utiliser un stockage hybride local et en ligne. Les données seront stockées en priorité en local et se mettront à jour automatiquement avec le serveur à chaque action, si une connexion internet est détectée. Cela permettra une utilisation hors ligne et une réactivité accrue, tout en partageant les données avec d'autres utilisateur et en évitant une suppression involontaire.

Le développement de Dotspot 2.0 ajoutera aussi un grand nombre de fonctionnalités, notamment la possibilité de l'utiliser sur le web et d'ajouter une bibliothèque d'icônes de matériels nécessaires et d'Equipement de Protection Individuelle (EPI).

ANNEXES

1. ANNEXE 1 : Maquette de l'application Dotspot Free antérieure au développement



2. ANNEXE 2 : Extrait du fichier boxes.dart

```dart
import 'package:dotspot_free/backend/model/block.dart';
import 'package:dotspot_free/backend/model/document.dart';
import 'package:hive/hive.dart';
// hive documentation


class Boxes {
  // # Blocks

  //[...]

  // # Documents

  /// Retourne la liste de tout les [Document]s
  static List<Document> findAllDocuments() {
    return Hive.box<Document>('document').values.toList();
  }

  /// Retourne la liste de tout les [Document]s sous forme de [Box]
  static Box<Document> findAllDocumentsAsBox() {
    return Hive.box<Document>('document');
  }

  /// Supprime le [Document] avec l'[id] founi de la base de données
  static Document? findDocumentById(int id){
    return Hive.box<Document>('document').getAt(id);
  }

  /// Ajoute le [Document] fourni ([Document]) à la base de données
  /// et retourne le nouvel id du document
  static Future<int> addDocument(Document document) async {
    return await Hive.box<Document>('document').add(document);
  }

  /// Supprime le [Document] avec l'[id] founi de la base de données
  static deleteDocument(int id){
    Hive.box<Document>('document').deleteAt(id);
  }
}
```