

République Algérienne Démocratique et Populaire  
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique  
Université A. Mira de Béjaïa  
Faculté des Sciences Exactes  
Département d'Informatique



## Mémoire de Fin de Cycle

En vue de l'obtention du diplôme de Licence en Informatique Générale

**Thème**

---

# Intitulé de votre mémoire

---

Réalisé par

M. NOM Prénom	M. NOM Prénom
M <sup>lle</sup> NOM Prénom	M. NOM Prénom
M <sup>lle</sup> NOM Prénom	M <sup>lle</sup> NOM Prénom

Devant le jury composé de

<b>Président :</b>	M. NOM Prénom	Grade de l'enseignant	Université de Béjaïa
<b>Examineur :</b>	M. NOM Prénom	Grade de l'enseignant	Université de Béjaïa
<b>Examinatrice :</b>	M <sup>me</sup> NOM Prénom	Grade de l'enseignante	Université de Béjaïa
<b>Encadrant :</b>	M. NOM Prénom	Grade de l'enseignant	Université de Béjaïa

Promotion 2017 - 2018

---

# Remerciements

---

Texte ...

Texte ...

Texte ...

Texte ...

Texte ...

# Dédicaces

---

Texte ...

*M. NOM Prénom*

Texte ...

*M<sup>lle</sup> NOM Prénom*

# Table des matières

Table des matières	i
Table des figures	iii
Liste des tableaux	iv
Liste des abréviations	v
<b>Introduction Générale</b>	<b>1</b>
<b>1 Contexte et problématique</b>	<b>3</b>
1.1 Introduction . . . . .	3
1.2 Contexte du projet . . . . .	3
1.3 Problématique . . . . .	3
1.4 UML . . . . .	4
1.5 Application mobile . . . . .	4
1.6 Cahier des charges . . . . .	5
1.7 Conclusion . . . . .	6
<b>2 Spécification des besoins et conception</b>	<b>7</b>
2.1 Introduction . . . . .	7
2.2 Spécification et analyse des besoins . . . . .	7
2.2.1 Identification des besoins . . . . .	7
2.2.2 Diagramme de cas d'utilisation . . . . .	8
2.2.3 Maquettes IHM et navigation . . . . .	9
2.2.4 Le modèle du domaine . . . . .	11
2.2.5 Diagramme des classes participantes . . . . .	11
2.3 Conception . . . . .	12
2.3.1 Diagramme d'interactions . . . . .	12
2.3.2 Modèle relationnel . . . . .	14
2.4 Conclusion . . . . .	14
<b>3 Implémentation</b>	<b>15</b>
3.1 Introduction . . . . .	15
3.2 Environnement de développement . . . . .	15

3.2.1	Android Studio . . . . .	15
3.2.2	Git et GitHub . . . . .	15
3.3	Outils de développement . . . . .	15
3.3.1	SDK de Android . . . . .	15
3.3.2	JDK . . . . .	16
3.4	Langage de programmation . . . . .	16
3.5	Persistance des données . . . . .	16
3.6	Librairies utilisées . . . . .	17
3.7	Présentation des interfaces . . . . .	17
3.7.1	Interface d'accueil . . . . .	17
3.7.2	Interface d'ajout d'un calendrier . . . . .	18
3.7.3	Interface d'ajout d'événement . . . . .	19
3.8	Conclusion . . . . .	19

<b>Conclusion générale et perspectives</b>	<b>20</b>
--	-----------

<b>Bibliographie</b>	<b>21</b>
----------------------	-----------

<b>4 Annexe</b>	<b>22</b>
-----------------	-----------

# Table des figures

1.1	Les différents types d'applications. . . . .	5
2.1	Diagramme de cas d'utilisation. . . . .	9
2.2	Maquette IHM et navigation . . . . .	10
2.3	Modèle du domaine. . . . .	11
2.4	Diagramme d'interaction «Ajout événement ». . . . .	13
2.5	Diagramme d'interaction «Suppression d'un calendrier». . . . .	13
3.1	Architecture de Room. . . . .	17
3.2	Interface d'accueil . . . . .	18
3.3	Ajout d'un calendrier. . . . .	18
3.4	Ajout d'un événement. . . . .	19

# Liste des tableaux

# Liste des abréviations

<b>SE</b>	Systeme d'Exploitation
<b>UML</b>	Unified Modeling Language
<b>UP</b>	Unified Process



# Introduction Générale

La nomophobie ou adikphonia est une phobie liée à la peur excessive d'être séparé de son téléphone mobile. Un phénomène inexistant avant l'apparition des téléphones mobiles. Cette innovation technologique est à l'origine de beaucoup de changements liés à nos habitudes quotidiennes. Désormais, nos téléphones mobiles contiennent notre musique préférée, nos souvenirs de famille, nos livres préférés et bien plus encore. Ils sont même reliés à nos comptes bancaires. La sensibilité de certaines données et le besoins quasi-permanent de ces derniers explique en grande partie les raisons de cette phobie.

Cependant ce n'est pas le seul changement que l'on constate au sein de la société moderne. Elle s'est vue complètement métamorphosée. Aujourd'hui, un manque de temps se fait sentir, comme si les 24 heures d'une journée ne suffisaient plus, certains craquent d'autres s'organisent. Les gens veulent maintenant s'épanouir dans leur famille, se réaliser dans leur emploi, s'impliquer dans la vie communautaire et avoir des activités sociales, sportives et culturelles satisfaisantes. Tous ces éléments prennent du temps. Alors la question qui se pose en tant qu'étudiant, comment faire en sorte d'accomplir toutes nos obligations professionnelles tout en trouvant le temps pour nous même et notre famille et nos proches.

En vue du rôle capitale que joue désormais les périphériques mobiles aussi puissants que des ordinateurs et du problème auquel on est confronté (manque de temps). Notre projet consiste à aider les étudiants à organiser et à planifier leur quotidien en vue d'une meilleure utilisation du temps dont il dispose et lui permettre d'établir un équilibre entre les différents aspects de sa vie selon ses priorités. Et ce, grâce une application mobile que l'on souhaite développer au cours de ce projet.

Pour réaliser une application de qualité il faut un plan, une méthodologie. La première étape de cette méthodologie est l'analyse qui va nous permettre de répertorier les fonctionnalités principales de l'application, tandis que l'étape de conception va nous permettre de modéliser les solutions suite à l'analyse en ayant recours au langage de modélisation choisi. La troisième étape consiste à la réalisation et l'implémentation de l'application.

Le mémoire décrivant ce travail sera partagé en trois chapitres qui englobe l'aspect documentation de ce projet.

Le premier chapitre sera dédié à son contexte et à la problématique abordé avec de brèves définitions sur les éléments importants tel que le langage UML et les applications mobiles ainsi qu'un cahier des charges.

Le deuxième chapitre «Spécification des besoins et conception » sera partagé en deux sections.

La première «Analyse et spécifications des besoins » comportera une analyse plus approfondie des besoins fonctionnels et non fonctionnels, Un diagramme de cas d'utilisation sera ajouté pour définir les fonctionnalités. Les maquettes IHM et les liens de navigations , le modèle du domaine et un diagramme de classes conception y seront inclus. Afin de passer à la deuxième section «La conception» dans laquelle on va établir des diagrammes d'interaction dans le but de formaliser les interactions de l'utilisateur avec le système selon les cas d'utilisation ainsi que déduire un modèle relationnel pour créer notre base de données.

Le dernier chapitre sera consacré à la partie pratique de la réalisation de notre projet. Dans un premier temps, nous allons énumérer les différents outils de développement qui nous ont permis de mener à bien notre application mobile. Ensuite, nous allons présenter les différents langage de programmation utilisés, les bibliothèques et enfin les différentes interfaces de notre application.

Pour finir, une conclusion comportant un récapitulatif du projet et un rappel des résultats principaux tout en mettant la lumière sur les limitations de la solution proposée et sur les perspectives.

# Chapitre 1

## Contexte et problématique

### 1.1 Introduction

Dans ce chapitre, nous allons présenter en premier lieu le contexte du projet ainsi qu'une brève description de la problématique. Nous allons par la suite présenter le langage UML et nous aborderons quelques définitions sur les applications mobile. Enfin, on termine ce premier chapitre avec l'élaboration du cahier des charges permettant d'avoir une vue globale sur les objectifs du projet.

### 1.2 Contexte du projet

On dit toujours que le temps est la chose la plus précieuse car on peut tout acheter sauf le temps. Quoi qu'on fasse, une journée durera toujours 24 heures et le rythme de vie de la société moderne nous fait sentir que l'on est perpétuellement en manque de temps. L'étudiant est l'une des classes sociales les plus touchées par ce problème. Entre ses études, ses activités sportives, ses loisirs et éventuellement son travail à mi-temps. Ainsi, il est difficile pour lui de pouvoir gérer toutes ses tâches quotidiennes et se retrouve parfois débordé.

Les smartphones ont révolutionné le monde de par le nombre de fonctionnalités et de possibilités qu'ils offrent, il est de nos jours indispensable d'en posséder dans la société moderne, plus de 50% de la population mondial possède déjà un smartphone et l'étudiant en représente une majeure partie.

### 1.3 Problématique

Dans ce projet, nous avons décidé de réaliser une application mobile pour la gestion du temps de l'étudiant. Elle sera capable de fournir une représentation simple et compréhensible de l'emploi du temps comme les séances de cours et de travaux dirigés ainsi que ses différentes activités. Elle devra aussi l'assister au quotidien en lui rappelant toutes les tâches qu'il a prévues.

L'étudiant a tendance à arriver en retard ou carrément à rater ses rendez-vous car il a prévu

autre chose au même moment sans le savoir ? L'étudiant fait pleins d'activités et n'arrive plus à se situer ? Il ne sait pas quoi répondre quand on lui demande s'il est libre à tels moment ?. Grâce à notre application, nous espérons contribuer à résoudre ces problèmes afin que l'étudiant soit plus productif et mieux s'organiser .

Tout au long de ce travail, nous allons raisonné avec le principe du rasoir d'Ockham également appelé principe de simplicité. Vous allez voir une conception simple, des diagrammes très légers pour aboutir à un résultat fidèle à ce principe.

## 1.4 UML

UML (Unified Modeling Language) se définit comme un langage de modélisation graphique et textuel. Il est destiné à décrire des besoins, spécifier et documenter des systèmes, esquisser des architectures logicielles, concevoir des solutions et communiquer des points de vue. UML unifie également les notations nécessaires aux différentes activités d'un processus de développement d'applications et offre, par ce biais, le moyen d'établir le suivi des décisions prises, depuis l'expression des besoins jusqu'à l'étape de réalisation.[1]

En fait, et comme son nom l'indique, UML n'a pas l'ambition d'être exactement une méthode : c'est un langage. UML est donc non seulement un outil intéressant, mais une norme qui s'impose en technologie à objets et à la quelle se sont rangés tous les grands acteurs du domaine, acteurs qui ont d'ailleurs contribué à son élaboration. [2]

## 1.5 Application mobile

Une application mobile n'est rien d'autre qu'un logiciel téléchargeable que l'on installe facilement sur nos smartphones(téléphone mobile intelligent) comme on ferait sur notre ordinateur. Le téléchargement de l'application mobile se fait suivant deux options :

- Sur téléphone par le biais de connexion Internet.
- Sur ordinateur en le branchant avec le téléphone mobile.

Il existe trois types distincts selon leurs spécificités techniques qui sont :

**Application Native :** Ces applications sont liés au système d'exploitation sur le quel sont installées car elles utilisent des caractéristiques reliées à celui-ci. Elle sont écrites dans un langage adapté au système d'exploitation en question. Ce type d'application est accessible seulement sur les systèmes d'exploitation auxquelles sont destinées. Ces plate-formes retirent 25% du prix de vente pour une application native payante.

**Application Web :** Ce sont toutes les applications conçues grâce aux outils de développement web actuels (HTML, CSS, JavaScript..). Elles sont accessible sur tout les mobiles via un

navigateur Web ce qui la rend plus intéressante sur le point de vue financier car les coûts de développement sont réduits vue qu'on développe une seule application qui est compatible avec tous les smartphones quelque soit leur système.

**Application Hybride** : sont des applications qui incorporent les deux principes de développement des types précédemment cités. Les caractéristiques des applications web et celles des application native. Elles pourront être distribuées sur les plate-formes de téléchargement telles que l'Apple Store (iOS), Play Store (Android) ou encore Windows Store (Windows Phone). L'utilisateur peut donc installer ces applications et consulter leur contenu sans avoir à passer par un navigateur web. Ce type d'application mobile minimise les charges et la durée de son développement même si cela sera au détriment du perfectionnement et de la qualité qui caractérise l'application native.

La figure suivante illustre les trois types cités.

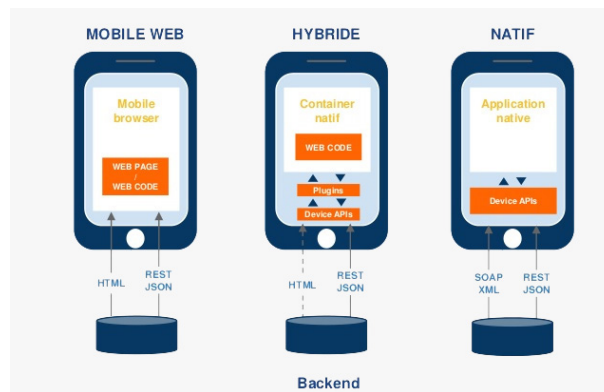


FIGURE 1.1 – Les différents types d'applications.

## 1.6 Cahier des charges

L'application à développer aura pour mission d'offrir une représentation des événements et des activités de l'utilisateur pendant les jours de la semaine. Pour cela, l'application devra répondre à ces besoins avec les fonctionnalités suivantes :

- Permettre à l'utilisateur d'organiser ses activités et les regrouper dans de différents calendriers.
- Permettre à l'utilisateur d'ajouter des activités dans le calendrier qui leur convient.
- Offrir une interface intuitive à l'utilisateur pour afficher ses activités.
- Générer des alertes/notifications pour les activités correspondantes.

## 1.7 Conclusion

Dans ce premier chapitre, nous avons présenté le contexte et la problématique abordé dans ce projet, défini le langage UML et less application mobiles et rédigé un cahier des charges incluant les besoins à satisfaire dans notre application. Dans le chapitre suivant, nous entamerons la phase d'analyse et de conception qui va nous permettre de modéliser et de proposer une solution aux besoins préalablement établies dans le cahier des charges.

# Chapitre 2

## Spécification des besoins et conception

### 2.1 Introduction

Dans ce chapitre, nous allons aborder la spécification et l'analyse des besoins de notre application (besoins fonctionnels, digramme cas d'utilisation, maquette IHM et navigation, modèle du domaine ). Suite à ça, nous allons concevoir notre application en utilisant les différents diagrammes UML et on conclut ce chapitre avec un passage au modèle relationnel qui va nous permettre d'implémenter notre base de données.

### 2.2 Spécification et analyse des besoins

Une phase décisive du processus de développement d'une application, elle permet d'identifier les acteurs ainsi que formaliser les besoins fonctionnels et non fonctionnels, déduire les différents cas d'utilisation à partir des besoins fonctionnels et les détaillés.

#### 2.2.1 Identification des besoins

**Besoins fonctionnels** Permettre à l'utilisateur d'ajouter des événements et de les administrer dans les calendriers qui leur correspondent qu'il a préalablement pris le soin de créer selon ses besoins. Il devra être rappelé et informé des événements pour lesquels il a préalablement défini les alertes, même lorsque l'application n'est pas active. Avoir une représentation détaillée sur son emploi du temps général (tous les calendriers )voire qu'un seul. Adapter la représentation pour afficher une seule journée, 3 jours, une semaine. Permettre à l'utilisateur de modifier à tout moment les informations déjà saisies.

**Besoins non fonctionnels** L'application doit remplir des critères non fonctionnels comme :

- La fiabilité : Si application crache et que l'utilisateur n'a pas reçu l'alerte, Cela peut s'avérer problématique.
- L'utilisabilité : L'utilisateur doit pouvoir maîtriser le fonctionnement de l'application facilement et rapidement.

- Performance : On est amené à vérifier notre emploi du temps plusieurs fois par jour, l'application doit être accessible rapidement pour ne pas contrarier l'utilisateur.

## 2.2.2 Diagramme de cas d'utilisation

Les diagrammes de cas d'utilisation sont des diagrammes utilisés pour donner une vision globale du comportement fonctionnel d'une application. Ils permettent de recueillir, d'analyser et d'organiser les besoins. Il s'agit donc de la première étape UML d'analyse d'un système. Les éléments d'un diagramme de cas d'utilisation sont :

**Acteur :** Un acteur est l'idéalisation d'un rôle joué par une personne externe, un processus ou une chose qui interagit avec un système. Il se représente par un petit bonhomme avec son nom inscrit dessous.

**Cas d'utilisation :** est une unité cohérente représentant une fonctionnalité visible de l'extérieur. Il réalise un service de bout en bout, avec un déclenchement, un déroulement et une fin, pour l'acteur qui l'initie. Un cas d'utilisation modélise donc un service rendu par le système, sans imposer le mode de réalisation de ce service. Il se représente par une ellipse contenant le nom du cas.

**Relations d'association :** Une relation d'association est un chemin de communication entre un acteur et un cas d'utilisation et est représenté par un trait continu [3].

Dans ce qui suit, nous décrivons le diagramme de cas d'utilisation associé à l'utilisateur de l'application à réaliser. L'application est supposé fournir une représentation simplifiée de l'emploi du temps d'une personne (ex :étudiant). Suite à ça, nous avons identifié un seul acteur qui est l'utilisateur lui-même.



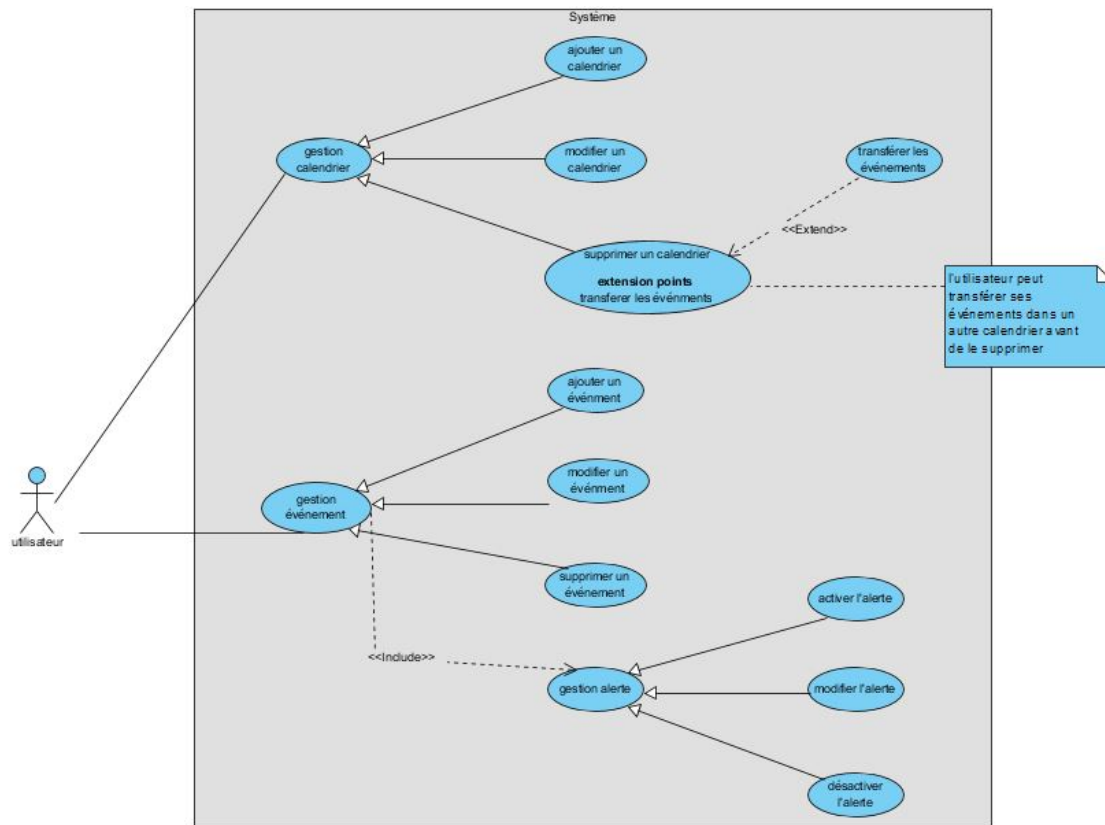


FIGURE 2.1 – Diagramme de cas d'utilisation.

Le diagramme de la figure 2.1 représente les différents cas d'utilisation associés à l'utilisateur. Ce dernier peut librement gérer ses calendriers et ses événements. La suppression d'un calendrier peut être étendue par « transférer les événements ». La gestion d'un événement inclut automatiquement « la gestion de l'alerte ».

### 2.2.3 Maquettes IHM et navigation

Une maquette est un produit jetable permettant aux utilisateurs d'avoir une vue concrète mais non définitive des interfaces de la future application. On parle aussi de prototypage d'IHM (Interface Homme Machine). Une maquette permet de bien cerner les fonctionnalités de l'application à réaliser.[4]

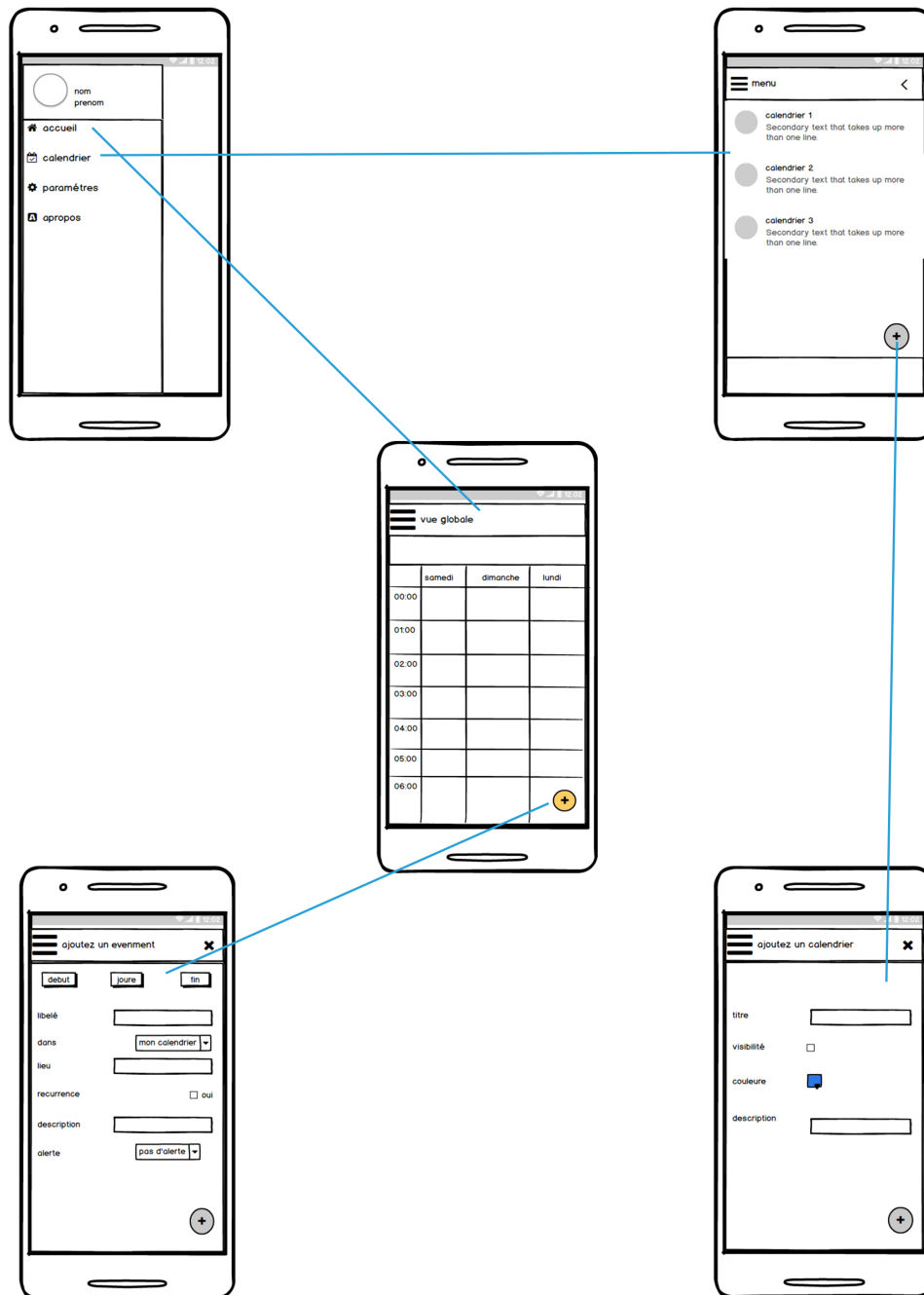


FIGURE 2.2 – Maquette IHM et navigation

Dans cette figure 2.2, on présente les interfaces principales et les liens qui les relient, vous remarquerez que nous avons essayé de rendre les interfaces aussi simple, minimaliste et intuitive

car nous estimons que ceci est capitale pour une utilisation fluide et rapide.

## 2.2.4 Le modèle du domaine

La modélisation des besoins par des cas d'utilisation s'apparente à une analyse fonctionnelle classique. L'élaboration du modèle des classes du domaine permet d'opérer une transition vers une véritable modélisation objet. L'analyse du domaine est une étape totalement dissociée de l'analyse des besoins (sections . Elle peut être menée avant, en parallèle ou après cette dernière.[2]) En analysant le domaine on peut concevoir le modèle du domaine qui représente les objets du monde réel dans ce domaine. Ces objets doivent contenir des attributs. Il est important de ne pas modéliser un concept par un attribut au lieu d'une classe et vice versa .

Dans la figure 2.3, on a établi le modèle du domaine de l'application à réaliser.

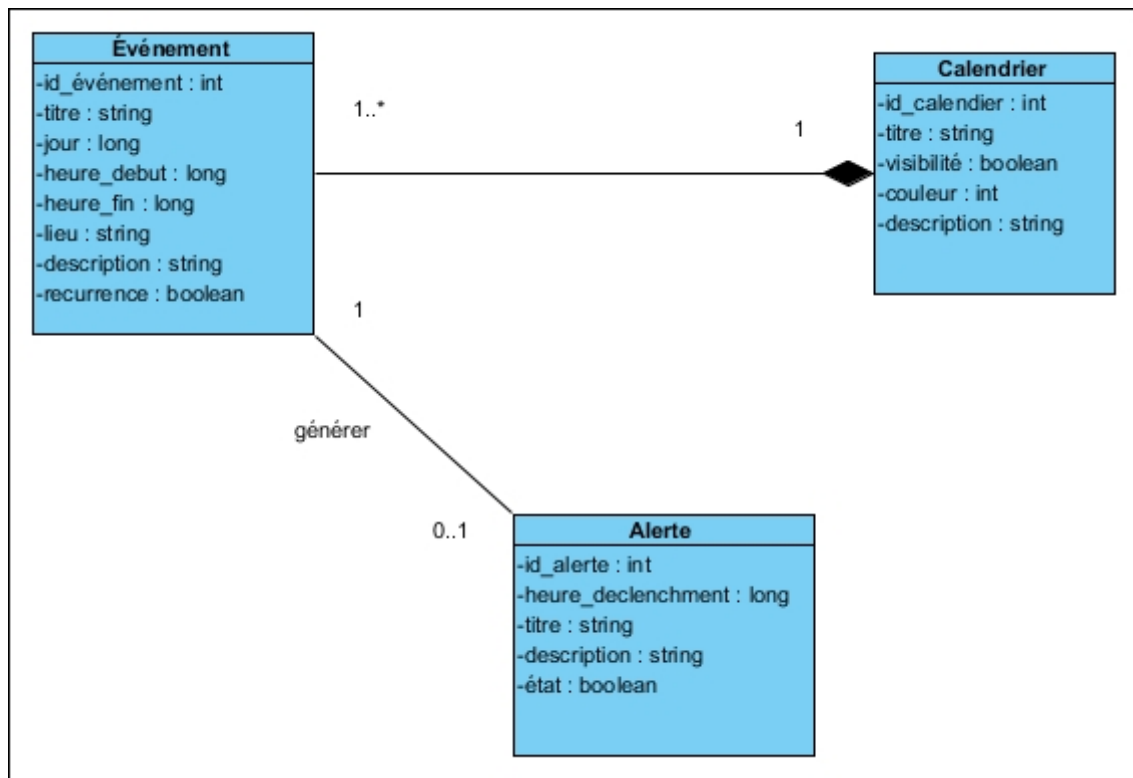


FIGURE 2.3 – Modèle du domaine.

## 2.2.5 Diagramme des classes participantes

Autrement dit les DCP, ils décrivent chaque cas d'utilisation c'est confrontation des diagrammes de cas d'utilisation, des interfaces graphiques et du modèle du domaine, les trois principales classes d'analyse et leurs relations.

Les règles : Pour compléter ce travail d'identification, nous allons ajouter des attributs et des opérations dans les classes d'analyse, ainsi que des associations entre elles.

- Les contrôles vont seulement posséder des opérations. Ces opérations montrent la logique de l'application, les règles transverses à plusieurs entités, bref les comportements du système informatique.
- Les dialogues vont posséder des attributs et des opérations. Les attributs représenteront des champs de saisie ou des résultats. Les résultats seront distingués en utilisant la notation de l'attribut dérivé. Les opérations représenteront des actions de l'utilisateur sur l'IHM.
- les classes entités réalisées à partir du modèle du domaine.

## 2.3 Conception

Après avoir tracé les grandes lignes de phase de spécification de besoins, mettons l'accent maintenant sur une phase fondamentale dans le cycle de vie d'un logiciel, la phase de conception. Dans cette section, nous présenter les différents diagrammes d'interaction. Ensuite élaborer le modèle relationnel .

### 2.3.1 Diagramme d'interactions

Ces diagrammes permettent de modéliser comment les objets communiquent entre eux cite5. Pour cela, nous utiliserons encore les trois types de classes d'analyse, on représente les interactions dans un format où chaque nouvel objet est ajouté en haut à droite. On représente la ligne de vie de chaque objet par un trait pointillé vertical. Cette ligne de vie sert de point de départ ou d'arrivée à des messages représentés eux-mêmes par des flèches horizontales. Par convention, le temps coule de haut en bas. Il indique ainsi visuellement la séquence relative des envois et réceptions de messages.

nous avons décidé d'inclure deux diagrammes seulement qui concerne l'ajout d'un événement et la suppression d'un calendrier.

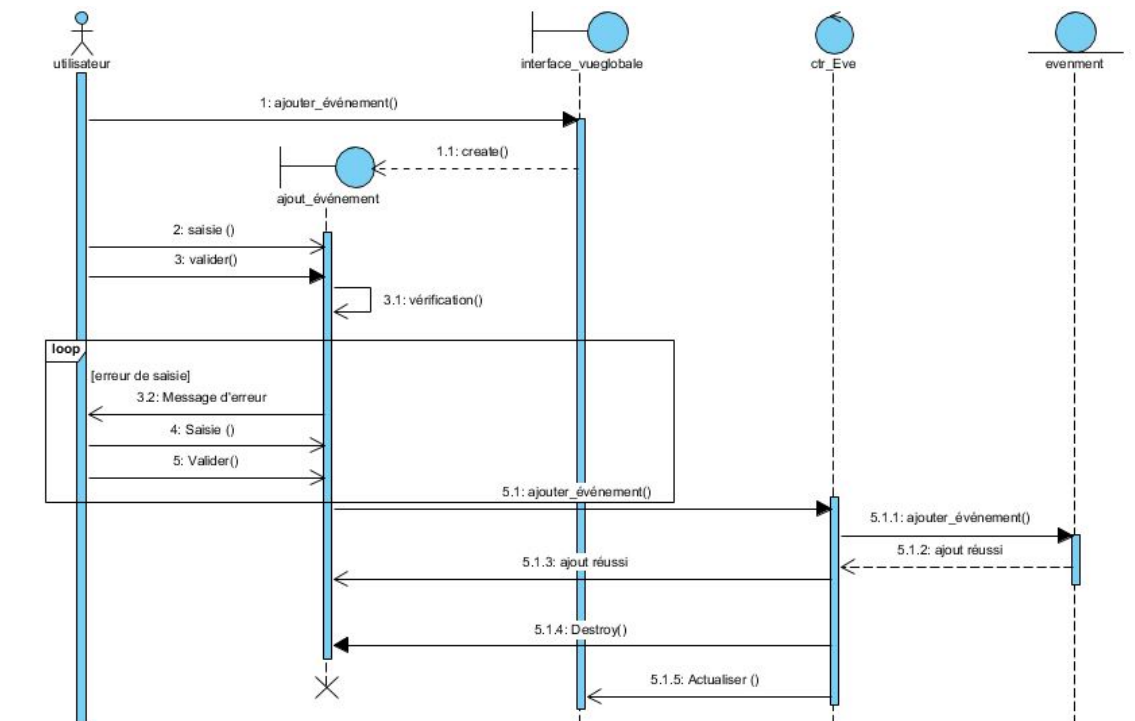


FIGURE 2.4 – Diagramme d'interaction «Ajout événement».

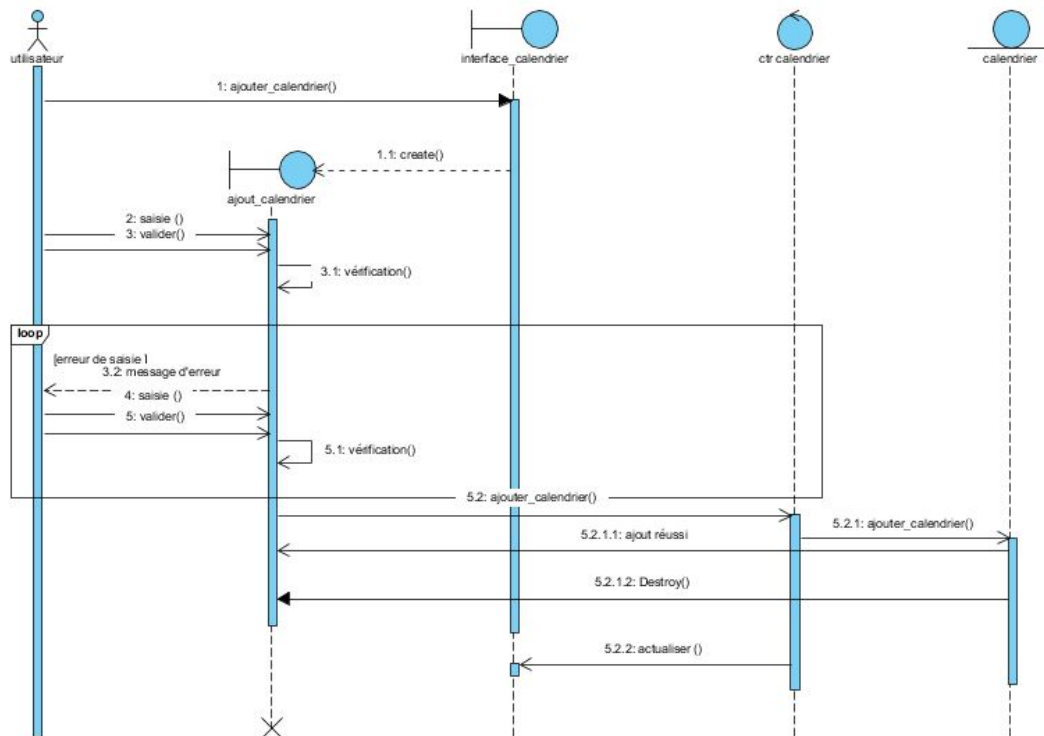


FIGURE 2.5 – Diagramme d'interaction «Suppression d'un calendrier».

### 2.3.2 Modèle relationnel

Le modèle relationnel représente la base de données comme un ensemble de tables, sans préjuger de la façon dont les informations sont stockées dans la machine. Les tables constituent donc la structure logique du modèle relationnel. Au niveau physique, le système est libre d'utiliser n'importe quelle technique de stockage dès lors qu'il est possible de relier ces structures à des tables au niveau logique. Les tables ne représentent donc qu'une abstraction de l'enregistrement physique des données en mémoire [5].

Dans notre contexte, nous avons établi ce schéma :

- Calendrier(id\_calendrier, titre, visibilité, couleur, description).
- Événement(id\_evenement, titre, jour, heure\_debut, heure\_fin, lieu, description, récurrence, #id\_calendrier, #id\_alerte).
- Alerte(id\_alerte, heure\_declenchement, titre, description, état).

## 2.4 Conclusion

Dans ce présent chapitre, nous avons modélisé et concrètement défini les solutions satisfaisant les exigences établies. Ceci est fait par une spécification et une analyse des besoins puis la réalisation des diagrammes d'interaction et diagramme de classes conception dans la phase de conception. Dans le chapitre suivant, nous allons entamer la réalisation de notre application en implémentant les solutions proposées dans ce chapitre.

# Chapitre 3

## Implémentation

### 3.1 Introduction

Ce dernier chapitre est consacré à la partie pratique de la réalisation de notre projet. Dans un premier temps, nous allons énumérer les différents outils de développement qui nous ont permis de mener à bien notre application mobile. Ensuite, nous allons présenter les différents langages de programmation utilisés, les bibliothèques et enfin les différentes interfaces de notre application.

### 3.2 Environnement de développement

#### 3.2.1 Android Studio

Android Studio est un environnement de développement intégré(EDI) permettant de développer des applications sous Android. Développé par Google, il se base sur l'EDI IntelliJ de JetBrains. Il offre les outils nécessaires pour développer des applications mobiles natives destinées à Android. Ainsi, il permet d'éditer des fichiers Java/Kotlin pour la partie programmation et des fichiers XML pour la partie graphique.

#### 3.2.2 Git et GitHub

Git est un logiciel libre de gestion de versions, sous licence publique générale GNU 2. GitHub est un service web de gestion et d'hébergement de projet de développement logiciel utilisant le logiciel Git.

### 3.3 Outils de développement

#### 3.3.1 SDK de Android

Le SDK (Software Development Kit) de Android est un ensemble d'outils de développement essentiel au développement d'application mobile sous Android, il inclut ainsi de différents outils

tel qu'un débogueur, de la documentation, un émulateur basé sur QEMU et un ensemble de bibliothèques logicielles.

### 3.3.2 JDK

Le JDK (Java Development Kit) est un ensemble d'outils et de bibliothèques logicielles destinées à la programmation Java. Il est nécessaire notamment pour la compilation du code Java qui sera transformé en bytecode pour être exécuté par la Java Virtual Machine(JVM).

## 3.4 Langage de programmation

**Java** est un langage de programmation orienté objet, puissant, il a la particularité d'être portable c'est-à-dire avoir la possibilité d'exécuter les programmes écrits en Java sous n'importe quel système d'exploitation grâce à la JVM incluse dans le JDK.

**XML** eXtensible Markup Language (Langage de balisage extensible en français) est un métalangage informatique de balisage générique. Il permet ainsi de structurer des données grâce à des balises.

## 3.5 Persistance des données

Comme nous l'avons vu durant les chapitres précédents, il est primordial de stocker en permanence les données. Ceci dit, une base de données locale est suffisante dans notre contexte.

**SQLite** est une bibliothèque de Android qui propose un moteur de base de données relationnelles accessible par SQL.

**Room** est une librairie de base de données développée par Google. Elle est une couche d'abstraction à SQLite. En effet, Room facilite la gestion de la base de données, de sa création à la lecture des données en passant par leur mise à jour de manière fluide pour exploiter toute la puissance de SQLite. Son principal atout est de détecter les erreurs de SQL à la compilation du code. Elle offre aussi la possibilité d'exécuter les requêtes SQL dans différents Threads évitant ainsi de se surcharger le Thread principal, Elle permet aussi de mettre en cache des données lors de l'absence d'une connexion Internet. Elle est composée de :

**Entités** : Les entités est l'ensemble de classes dont chacune d'elle représente une table dans la base de données.

**DAOs(Data Access Objects)** : Ce sont des interfaces qui ont pour rôle de gérer toutes les requêtes SQL, elle agit comme un intermédiaire entre la base de données et le reste de l'application. Chaque entité doit avoir son propre DAO.

**Base de données** : Elle contient toutes les tables et toutes les données stockées.



La figure suivante résume l'architecture de Room

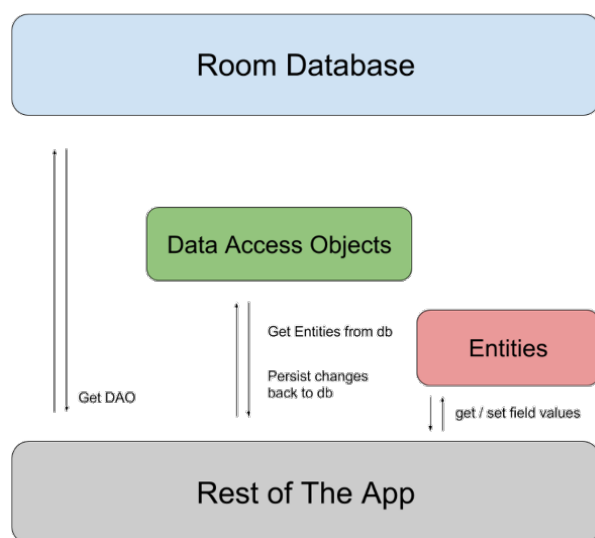


FIGURE 3.1 – Architecture de Room.

## 3.6 Bibliothèques utilisées

**WeekView** est une bibliothèque qui affiche la vue d'un calendrier. Elle a été utile pour l'affichage des événements sur un calendrier, elle implémente également plusieurs fonctionnalités rendant ainsi l'application plus intuitive.

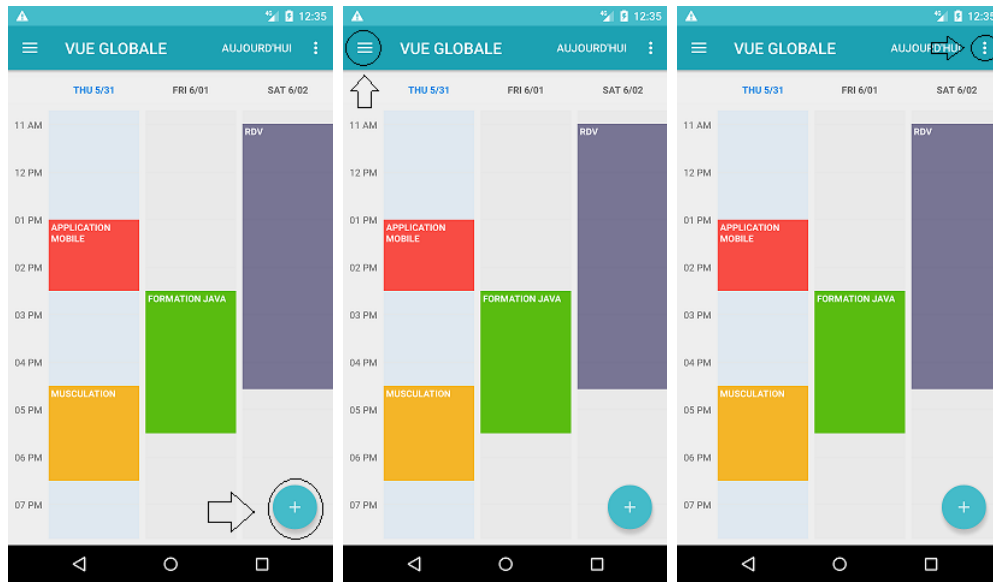
**ColorPicker** est comme son nom l'indique, une bibliothèque qui permet de choisir une ou plusieurs couleurs.

Il est à noter que les bibliothèques utilisées étaient libres, open-source et disponibles sur GitHub.

## 3.7 Présentation des interfaces

### 3.7.1 Interface d'accueil

Cette première figure présente l'interface d'accueil, ceci est la première interface affichée à l'utilisateur au lancement de l'application. Son principal composant est une vue calendrier, il regroupe les différents événements préalablement ajoutés par l'utilisateur. Chaque événement est affiché avec la couleur du calendrier auquel il appartient. Aussi, nous avons un bouton flottant (figure 3.2(a)) qui permet d'ajouter des événements, un menu latéral pour naviguer (figure 3.2(b)) entre les différentes activités de l'application et enfin un menu qui ouvre une liste déroulante pour changer la vue du calendrier (figure 3.2(c)).



(a) Bouton flottant

(b) Bouton menu

(c) Bouton option

FIGURE 3.2 – Interface d'accueil

### 3.7.2 Interface d'ajout d'un calendrier

La figure 3.3 représente l'interface d'ajout d'un calendrier, elle nous permet d'ajouter un calendrier. Pour cela, il suffit de remplir les champs et de faire le choix de la couleur avec laquelle seront affichés les événements de appartenant au calendrier.

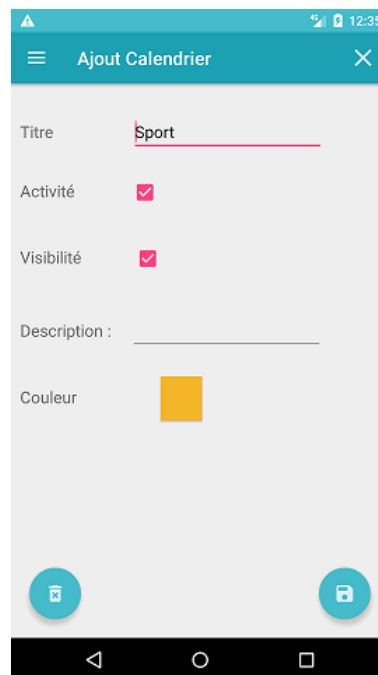


FIGURE 3.3 – Ajout d'un calendrier.

### 3.7.3 Interface d'ajout d'événement

La figure ci-dessous représente l'interface d'ajout d'un événement. L'utilisateur est invité à remplir les champs, à choisir à quel calendrier appartient-il et enfin décider à quel moment il souhaite être alerté voire désactiver complètement l'alerte.

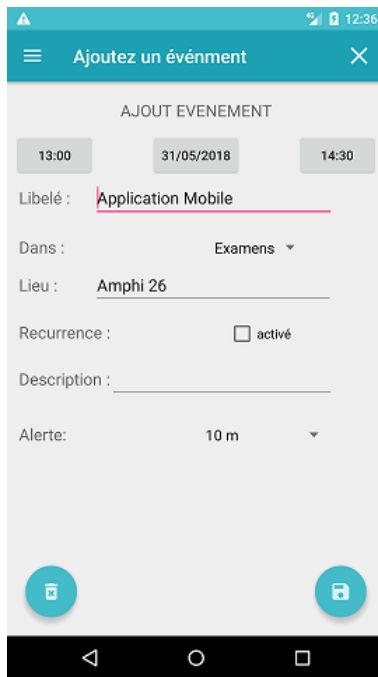


FIGURE 3.4 – Ajout d'un événement.

## 3.8 Conclusion

Dans ce dernier chapitre, nous avons vu les différents environnements de développement et décrit les outils utilisés notamment GitHub qui a été un outil clé pour notre travail collaboratif. Nous avons vu également les différents langage de programmation. Concernant la base de données, Room a été un véritable plus avec sa facilité à gérer et implémenter la base de données. Aussi, les différentes librairies qui nous ont permis de gagner un temps précieux. Enfin, nous avons vu les différentes interfaces principales qui composent notre application.

# Conclusion générale et perspectives

Texte ...

En guise de perspectives, nous envisageons de ...

# Bibliographie

- [1] PASCAL ROQUES, *UML 2 Modéliser une Application Web*, 4<sup>e</sup> édition, Eyrolles, Paris, 2008.
- [2] LAURENT AUDIBERT, *UML 2 de l'apprentissage à la pratique*, Publié le 31 octobre 2006 - Mis à jour le 12 janvier 2009 ,sur :[https ://laurent-audibert.developpez.com/Cours-UML/](https://laurent-audibert.developpez.com/Cours-UML/)
- [3] LAURENT AUDIBERT, *UML 2 : de l'apprentissage à la pratique*, Ellipse, 2<sup>e</sup>, 2009
- [4] [http ://niedercorn.free.fr/iris/iris1/uml/uml10.pdf](http://niedercorn.free.fr/iris/iris1/uml/uml10.pdf).
- [5] [http ://www.ai.univ-paris8.fr/lysop/bd/seance4-ModeleRel.pdf](http://www.ai.univ-paris8.fr/lysop/bd/seance4-ModeleRel.pdf).
- [6] [http ://tvaira.free.fr/dev/fiches/fiche-c16-diagramme\\_classes\\_conception.pdf](http://tvaira.free.fr/dev/fiches/fiche-c16-diagramme_classes_conception.pdf).

# Annexe

# Résumé

Texte en français ...

***Mots clés :*** *mot clé 1, mot clé 2, mot clé 3, mot clé 4, mot clé 5*

# Abstract

Texte en anglais ...

***Keywords :*** *mot clé 1, mot clé 2, mot clé 3, mot clé 4, mot clé 5*