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

Conception et réalisation d'une application Android pour la gestion du temps de l'étudiant

Réalisé par

M. TAYEB CHERIF Mohand Said
M. YAYADENE Abderzak
M. SALIMI Salim
M. ZADIR Azzedine

Devant le jury composé de

Présidente :	M^{me} F. BOULAHROUZ	Université de Béjaïa
Examinatrice :	M^{me} A. TIAB	Université de Béjaïa
Encadrant :	M. K. AKILAL	Université de Béjaïa

Promotion 2017 - 2018

Remerciements

Pour commencer, nous tenons à remercier notre encadrant M. K. AKILAL, pour l'orientation, la confiance, la patience qui ont constitué un apport considérable sans lequel ce travail n'aurait pas pu être mené à bon port.

Nos remerciements s'étendent également à M^{lle} S. BOUCHELAGHEM pour son chaleureux accueil et le temps qu'elle nous a consacré, aux personnes qui nous ont apporté leur aide précieuse et qui ont contribué à l'élaboration de ce travail.

Nous remercions également chacun des membres du jury pour l'intérêt qu'ils ont porté pour notre travail, pour les remarques et les conseils contribuant ainsi à l'enrichissement ce modeste travail.

Enfin, nous adressons nos plus sincères remerciement à nos familles, à nos proches et à nos amis, qui nous ont accompagné, aidé, soutenu et encouragé tout au long de la réalisation de ce mémoire.

Table des matières

Table des figures

Liste des abréviations

CSS	Cascading Style Sheets
DAO	Data Access Objects
DCP	Diagramme de Classe Participante
HTML	HyperText Markup Language
IHM	Interactions homme-machine
iOS	iPhone Operating System
JDK	Java Developement Kit
JVM	Java Virtual Machine
SDK	Software Developpement Kit
SE	Système d'Exploitation
SGBD	Système de Gestion de Base de Données
SQL	Structured Query Language
UML	Unified Modeling Language
XML	Extensible Markup Language

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 dernières expliquent 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. Les gens veulent s'épanouir dans leur familles, évoluer dans leurs carrières, 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 pour accomplir toutes nos obligations pédagogiques tout en trouvant le temps pour nous même et nos familles et nos proches ?

Vu le rôle capital que jouent 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 pour une meilleure utilisation du temps dont ils disposent et leur permettre d'établir un équilibre entre les différents aspects de leur vie selon leurs priorités. Et ce, grâce à une application mobile que nous comptons développer.

Pour réaliser une application de qualité il faut un plan ; une méthodologie. La première étape de cette méthodologie est l'analyse qui nous permettra 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 en la réalisation et l'implémentation de l'application.

Ce mémoire sera divisé en trois chapitres.

Le premier sera dédié au contexte du projet et à la problématique que l'on abordera avec de brèves définitions sur les éléments importants tels que le langage UML et les applications mobiles. L'on conclura ce chapitre par un cahier des charges clair et précis.

Le deuxième chapitre 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 présenté pour définir les fonctionnalités. Les maquettes IHM et les liens de navigation ; le modèle du domaine ; et un diagramme de classes conception y seront

aussi inclus. La deuxième section « Conception » comportera : 1) des diagrammes d'interaction qui décrivent formellement les interactions de l'utilisateur avec le système. 2) un modèle relationnel pour implémenter notre base de données.

Le dernier chapitre sera consacré à la partie pratique, ou à 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, l'on présentera les différents langages de programmation utilisés, les librairies, et enfin les différentes interfaces de notre application.

Nous achèverons ce mémoire par 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 d'amélioration.

Chapitre 1

Contexte et problématique

1.1 Introduction

Dans ce chapitre, nous présenterons en premier lieu le contexte et la problématique du projet. Nous présenterons brièvement le langage UML et nous aborderons quelques définitions sur les applications mobiles. Nous terminerons avec l'élaboration d'un cahier des charges permettant d'avoir une idée plus précise sur les objectifs du projet.

1.2 Contexte du projet

L'on dit toujours que le temps est la chose la plus précieuse car on peut tout acheter hormis le temps. Quoi qu'on fasse, une journée durera toujours 24 heures bien que 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, l'étudiant a du mal à 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 quasi-indispensable d'en posséder. Plus de 50% de la population mondiale possède déjà un smartphone et les étudiants en présentent une majeure partie [1].

1.3 Problématique

Nous avons, dans ce projet, décidé de réaliser une application mobile pour la gestion du temps de l'étudiant. Elle devrait être 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 devrait 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. Il n'arrive pas à se situer. C'est dans ce cadre que s'inscrit notre application : à assister l'étudiant et à l'aider à être plus productif et mieux organisé.

1.4 UML

UML (Unified Modeling Language) est 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 [2].

En effet, 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 [3].

1.5 Application mobile

Une application mobile n'est rien d'autre qu'un logiciel téléchargeable que l'on installe facilement sur nos smartphones(téléphones mobiles intelligents) comme on ferait sur nos ordinateurs.

Il existe trois types d'applications mobiles selon leurs spécificités techniques qui sont :

Applications Natives : Ces applications sont liées au système d'exploitation sur lequel elles sont installées car elles utilisent des caractéristiques reliées à celui-ci. Elle sont écrites dans un langage adapté au système d'exploitation en question.

Applications Web : Ce sont toutes les applications conçues grâce aux outils de développement web actuels (HTML, CSS, JavaScript, etc.). Elles sont accessible sur tous les mobiles via un navigateur Web ce qui les rend plus intéressante du point de vue financier, car les coûts de développement sont réduits vu qu'on développe une seule application qui est compatible avec tous les smartphones quelque soit leurs systèmes.

Applications Hybrides : sont des applications qui incorporent les deux principes de développement précédemment cités. Les caractéristiques des applications web et celles des applications natives. 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.

La figure suivante illustre les trois types cités.

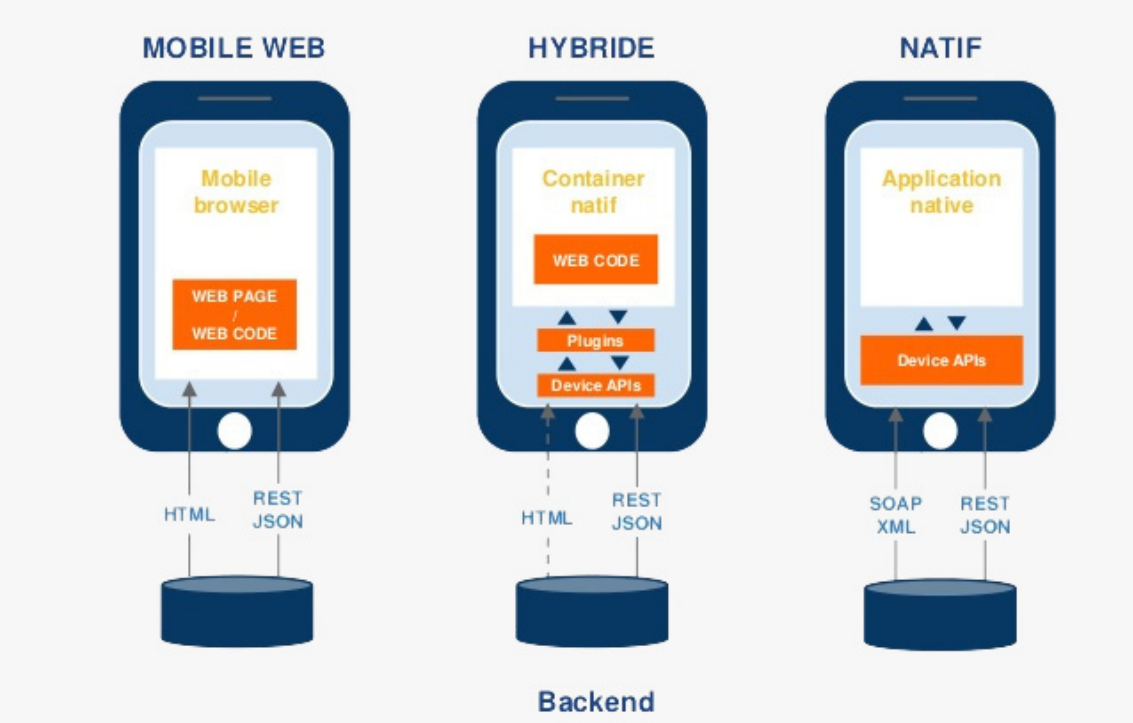


FIGURE 1.1 – Les différents types d'applications [4].

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 offrir les fonctionnalités suivantes :

- Permettre à l'utilisateur d'organiser ses activités et de les regrouper dans de différents calendriers.
- Permettre à l'utilisateur d'ajouter des activités dans les calendriers qui leur conviennent.
- Offrir une interface intuitive à l'utilisateur pour afficher ses activités.
- Générer des alertes/notifications pour les activités préalablement saisies et pour lesquelles l'utilisateur souhaite être rappelé par l'application.

1.7 Conclusion

Dans ce premier chapitre, nous avons présenté le contexte et la problématique abordés dans ce projet. Nous avons défini brièvement le langage UML et les applications mobiles et rédigé un cahier des charges incluant les besoins à satisfaire les besoins auxquels notre application est censée répondre. 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 étudions la spécification et l'analyse des besoins de notre application (besoins fonctionnels, digramme cas d'utilisation, maquette IHM et navigation, modèle du domaine). Et nous concevrons notre application en utilisant les différents diagrammes UML et nous terminerons par le passage au modèle relationnel qui nous permettra d'implémenter notre base de données.

2.2 Spécification et analyse des besoins

C'est une phase décisive du processus de développement d'une application. Elle permet d'identifier les acteurs, de formaliser les besoins fonctionnels et non fonctionnels, et de déduire les différents cas d'utilisation à partir des besoins fonctionnels.

2.2.1 Identification des besoins

Besoins fonctionnels Permettre à l'utilisateur d'ajouter des événements et de les administrer dans des calendriers qu'il aurait préalablement pris le soin de créer selon ses besoins. Il devra être rappelé et informé des événements pour lesquels il aurait défini les alertes, et ce même lorsque l'application n'est pas active. Avoir une représentation détaillée de son emploi du temps général (tous les calendriers) ou d'un seul calendrier à la fois (Exemple : limiter l'affichage des activités sportives). Adapter la représentation pour afficher une seule journée, 3 jours, une semaine, et finalement 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 : L'utilisateur devra recevoir des les alertes programmées même si l'application cesse de fonctionner.
- 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 rapidement accessible 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 [3]. 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. On le représente par un petit bonhomme avec son nom inscrit en 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. On 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.

Dans ce qui suit, nous décrivons le diagramme de cas d'utilisation de l'utilisateur (unique acteur) de l'application.

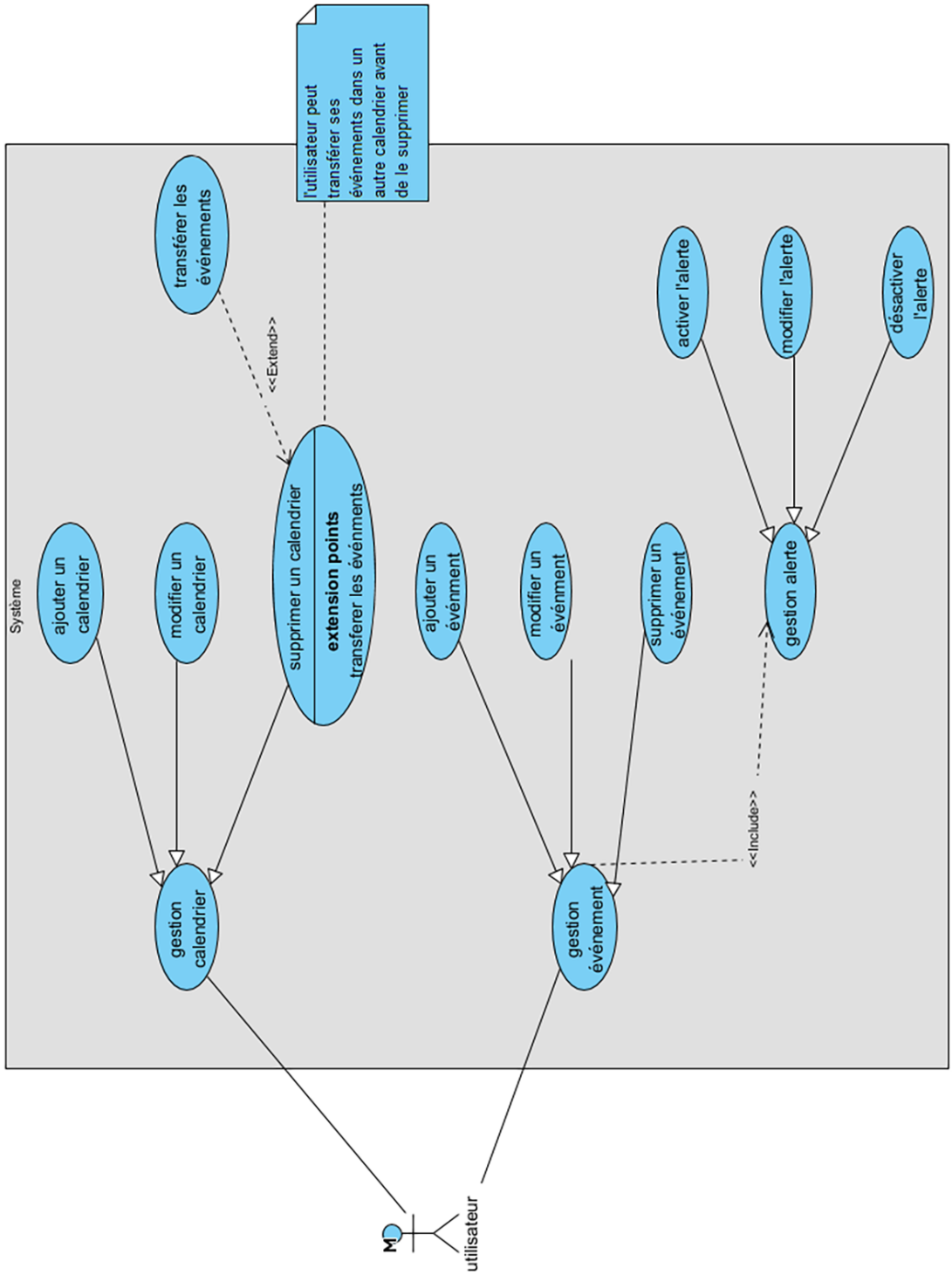


FIGURE 2.1 – Diagramme de cas d'utilisation.

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 un « 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 [5].

La Figure 2.2 présente les interfaces principales et les liens qui les relient. Nous remarquons que ces interfaces sont simples, minimalistes, et intuitives, ce qui à notre sens, est capital pour une utilisation fluide et rapide.

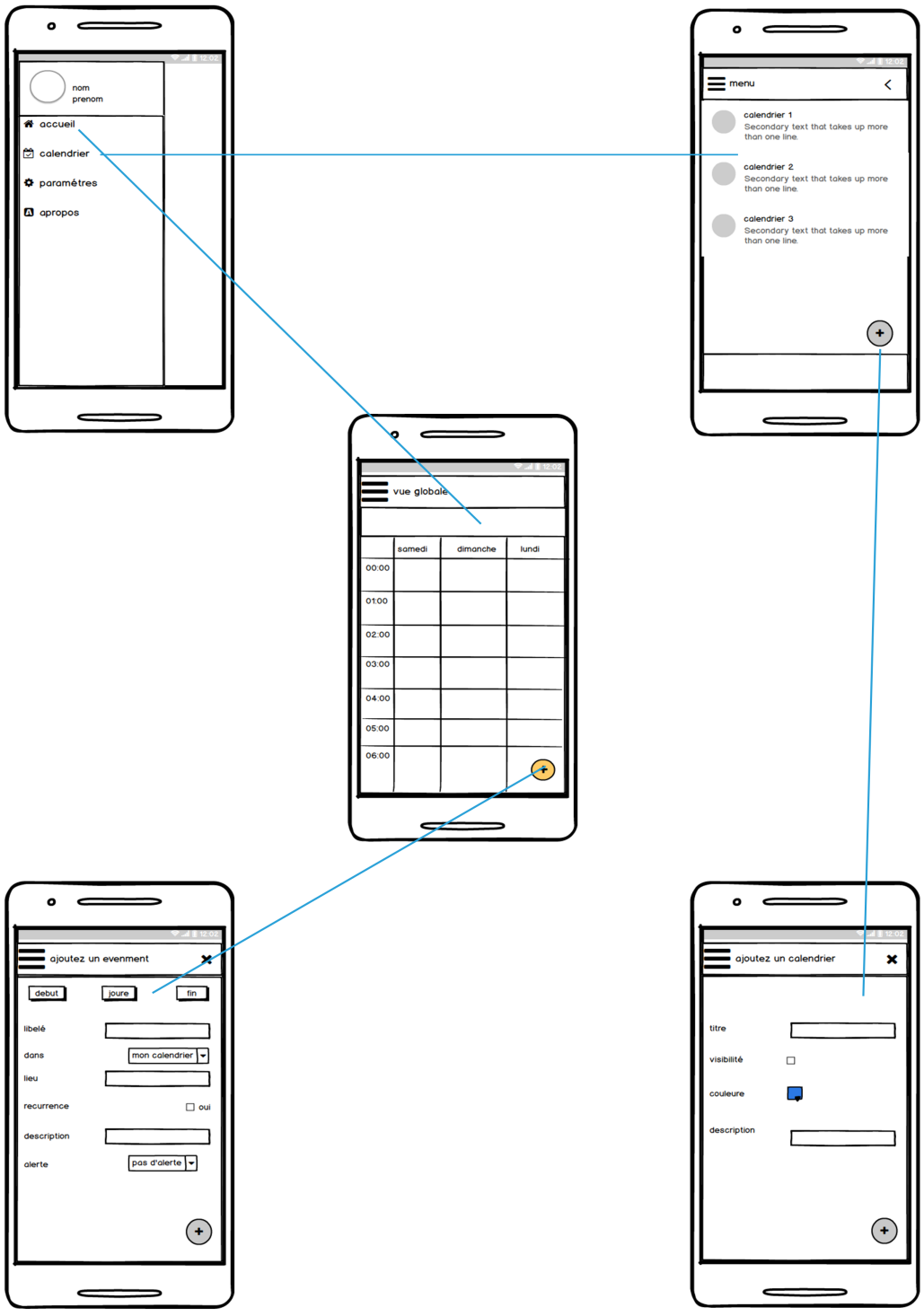


FIGURE 2.2 – Maquette IHM et navigation

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. Elle peut être menée avant, en parallèle ou après cette dernière [3]. En analysant le domaine, on peut concevoir ce modèle qui représente les objets du monde réel. Ces objets doivent contenir des attributs. La Figure 2.3 détaille le modèle du domaine de l'application à réaliser.

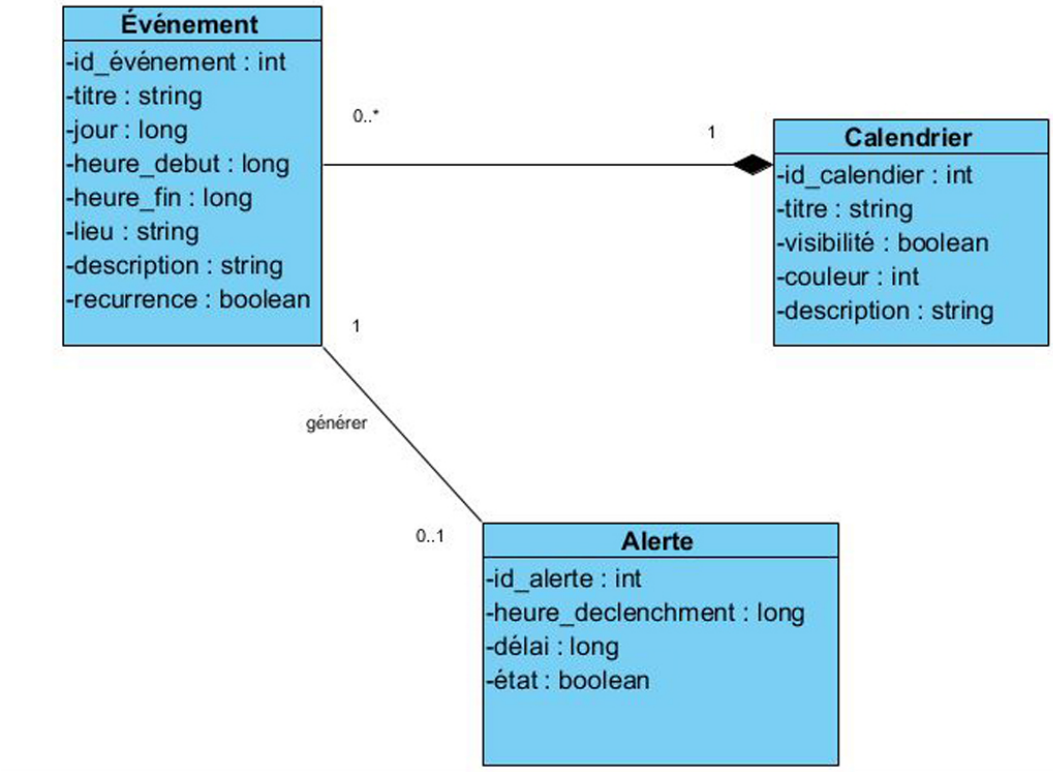


FIGURE 2.3 – Modèle du domaine.

2.2.5 Diagramme des classes participantes (DCP)

Le diagramme de classes participantes est particulièrement important puisqu'il effectue la jonction entre, d'une part, les cas d'utilisation, le modèle du domaine et la maquette , et d'autre part, les diagrammes de conception logicielle.

Il n'est pas souhaitable que les utilisateurs interagissent directement avec les instances des classes du domaine par le biais de l'interface graphique. En effet, le modèle du domaine doit être indépendant des utilisateurs et de l'interface graphique. De même, l'interface graphique du logiciel doit pouvoir évoluer sans répercussion sur le cœur de l'application. C'est le principe fondamental du découpage en couches d'une application. Ainsi, le diagramme de classes participantes modélise trois types de classes d'analyse, les dialogues, les contrôles et les entités ainsi que leurs relations [3].

Pour compléter ce travail d'identification, nous ajouterons des attributs et des opérations dans les classes d'analyse, ainsi que des associations entre elles.

- Les contrôles ne possèdent que des opérations. Ces opérations montrent la logique de l'application, les règles transverses à plusieurs entités, simplement dit les comportements du système informatique.
- Les dialogues possèdent 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.

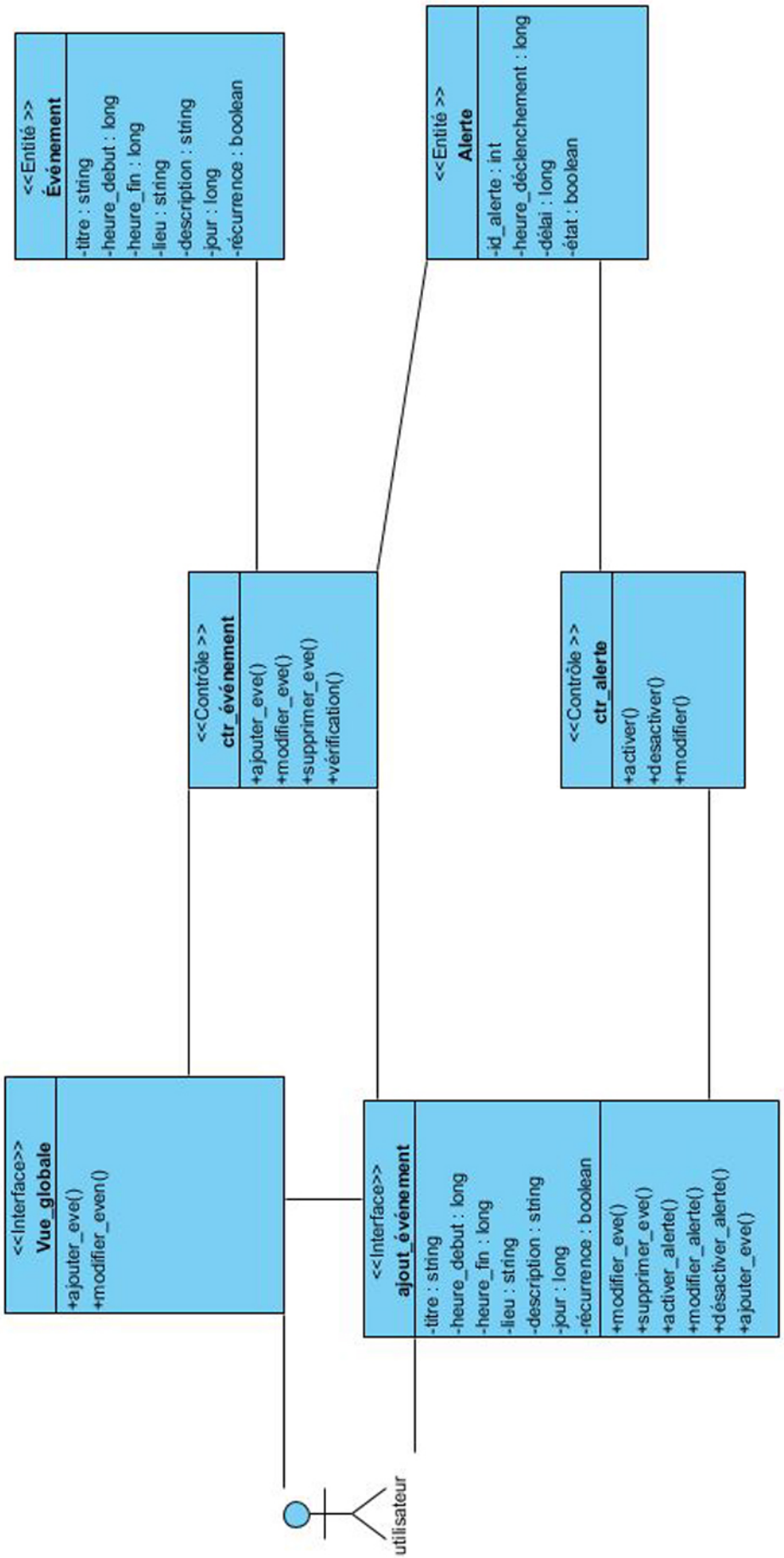


FIGURE 2.4 – Diagramme de classes participantes « Ajout événement ».

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 présenterons les différents diagrammes d'interaction. Et élaborerons le modèle relationnel.

2.3.1 Diagramme d'interactions

Ces diagrammes permettent de modéliser comment les objets communiquent entre eux [6]. Pour cela, nous utiliserons encore les trois types de classes d'analyse. L'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.

Par soucis de brièveté, nous n'incluons deux diagrammes : Celui de l'ajout d'un événement, et celui de la suppression d'un calendrier.

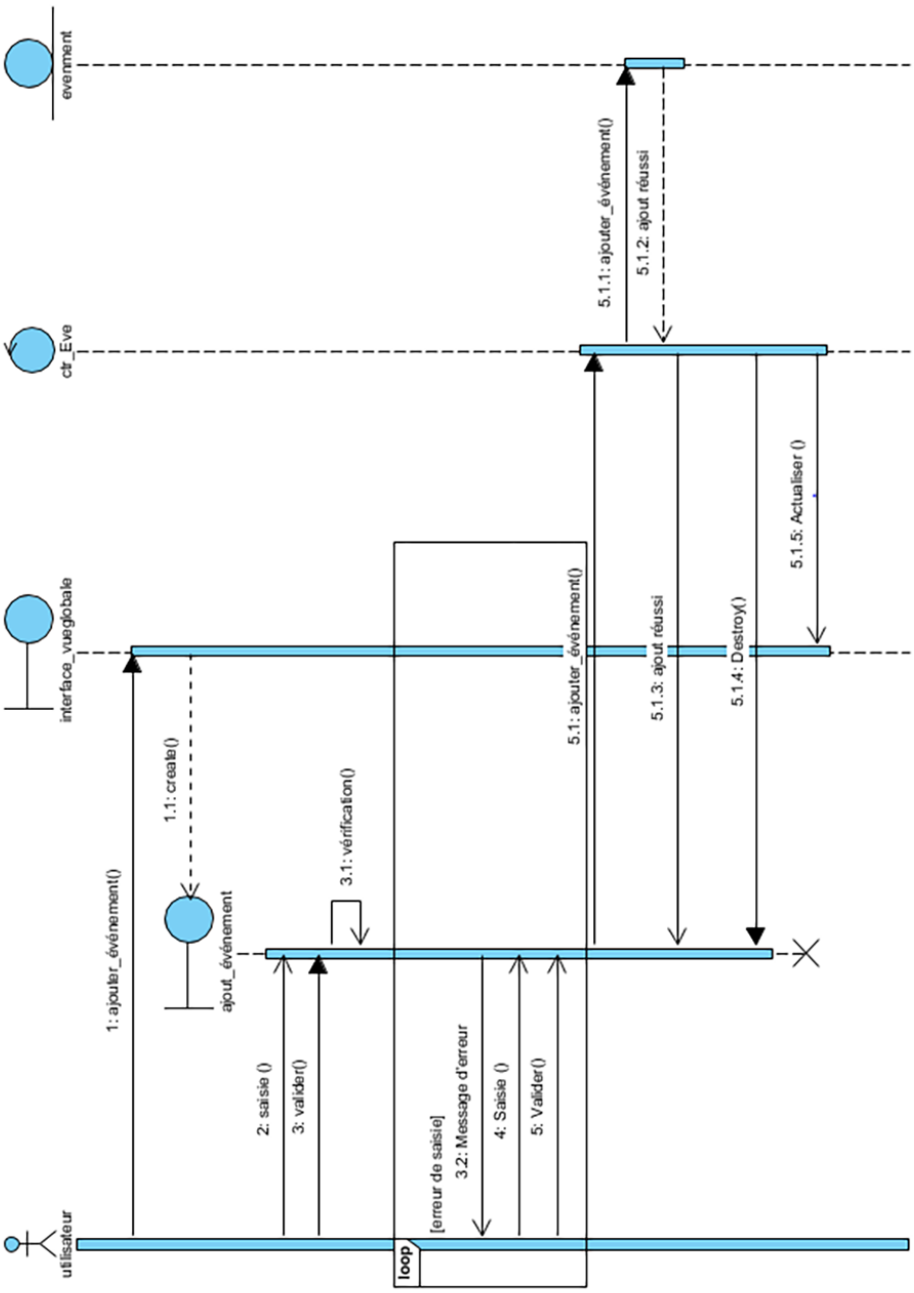


FIGURE 2.5 – Diagramme d'interaction « Ajout événement ».

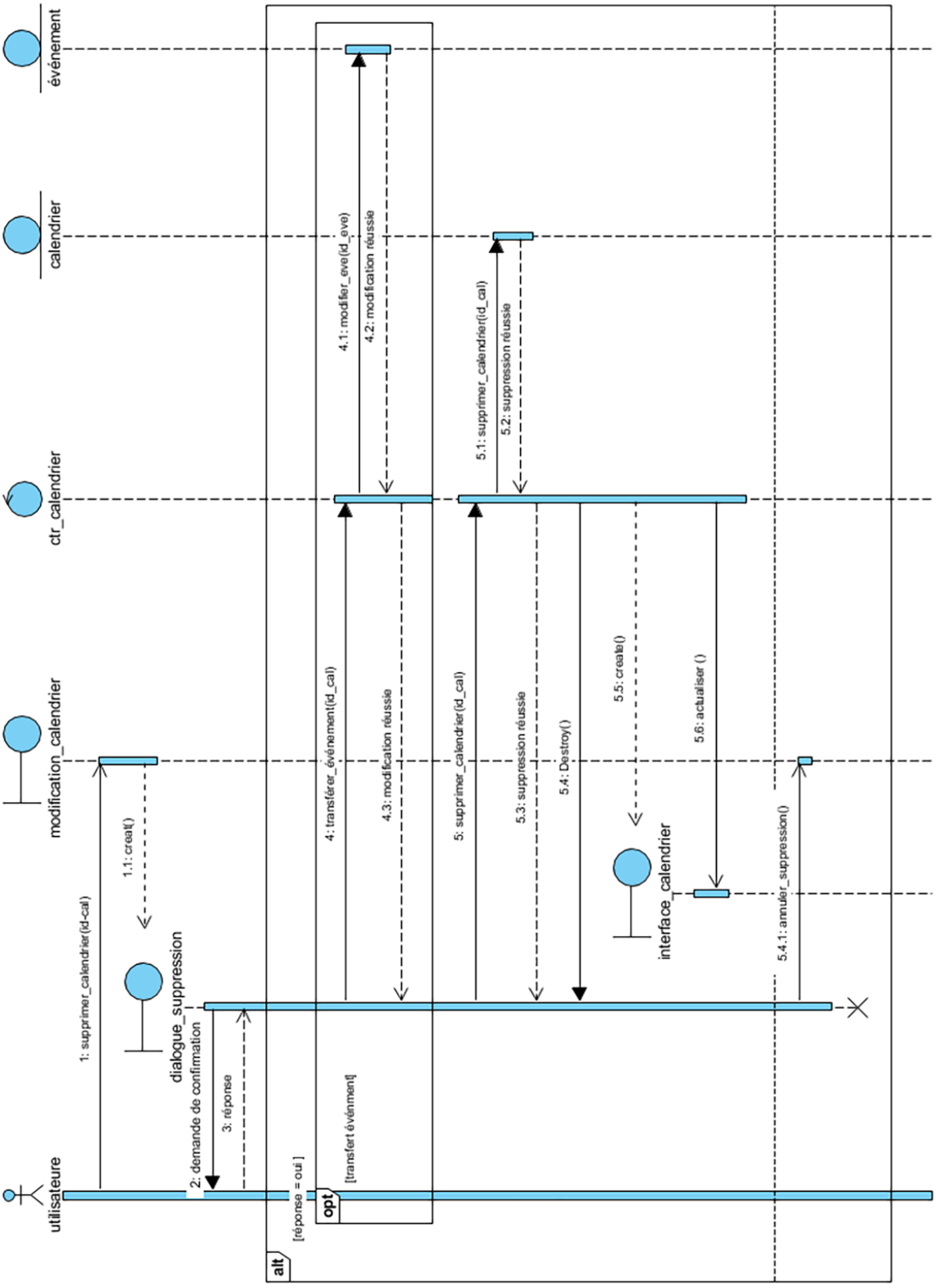


FIGURE 2.6 – Diagramme d'interaction « Suppression d'un calendrier ».

2.3.2 Modèle relationnel

Le modèle relationnel représente la base de données comme un ensemble de tables, sans préjugés 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 [7].

Les règles utilisées pour le passage du modèle du domaine vers le modèle relationnel sont [8] :

— Composition : La clé primaire des relations déduites des classes composantes doit contenir l'identifiant de la classe composite (quelles que soient les multiplicités).

— Association un-à-un : Cette règle consiste à ajouter un attribut clé étrangère dans la relation dérivée de l'entité ayant la cardinalité minimale à zéro.

Dans notre contexte, nous avons obtenu le schéma relationnel suivant :

— Calendrier(**id_calendrier**,titre,visibilité,couleur,description).

— Événement(**id_evenement**, titre, jour, heure_debut, heure_fin, lieu, description, récurrence,*#id_calendrier*).

— Alerte (**id_alerte**,heure_déclenchement, délai,*#id_evenement*) ——————————————————————————————

— Calendrier(**id_calendrier**,titre,visibilité,couleur,description).

— Événement(**id_evenement**, titre, jour, heure_debut, heure_fin, lieu, description, récurrence,*#id_calendrier*,heure_déclenchement,délai).

2.4 Conclusion

Dans le présent chapitre, nous avons modélisé et définit des solutions satisfaisants aux 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. Le prochain chapitre sera consacré à la réalisation de notre application e l'implémentant les solutions proposées.

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 énumérons les différents outils de développement qui nous ont permis de mener à bien notre application mobile. Ensuite, nous présenterons les différents langages de programmation utilisés, les librairies, 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 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. 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 GPL2. GitHub est un service web de gestion et d'hébergement de projet de développement logiciel utilisant le logiciel Git(qui, pour l'anecdote, a été racheté le 4 de ce mois par Microsoft).

Cet outil a été un véritable atout pour notre projet. Il nous a permis de travailler de manière collaborative, sur un code source unique, et de suivre l'évolution du travail de chacun en temps réel. Le lecteur peut accéder et suivre l'évolution de notre dépôt en visitant le lien ci-dessous : `https://github.com/silvermoon06/prototypapp`

3.3 Outils de développement

3.3.1 SDK de Android

Le SDK (Software Developpement Kit) d'Android est un ensemble d'outils de développement essentiel au développement d'application mobile Android, il inclut de différents outils tel qu'un débogueur, un émulateur basé sur QEMU et un ensemble de bibliothèques logicielles auxquels vient s'ajouter une documentation des plus riches.

3.3.2 JDK

Le JDK (Java Developement 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 puisannt langage de programmation orienté objet. Il a la particularité d'être portable c'est-à-dire qu'il est possible d'exécuter les programmes écrits en Java sous n'importe quel système d'exploitation grâce a 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 de structurer des données grâce à des balises et est utilisé essentiellement pour décrire les interfaces graphiques Android et autres fichiers de données globales.

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 un SGBD local qui propose un moteur de base de données relationnelles accessible par SQL. À la différence d'autres SGBD, il ne reproduit pas le schéma habituel client-serveur, il est directement intégré aux programmes.

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 en exploitant 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 qui, au niveau SQLite, correspondent à des tables SQL dans la base de données.

DAOs (Data Access Objects) : Ce sont des interfaces qui ont pour rôle de gérer toutes les requêtes SQL, elles agissent 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.

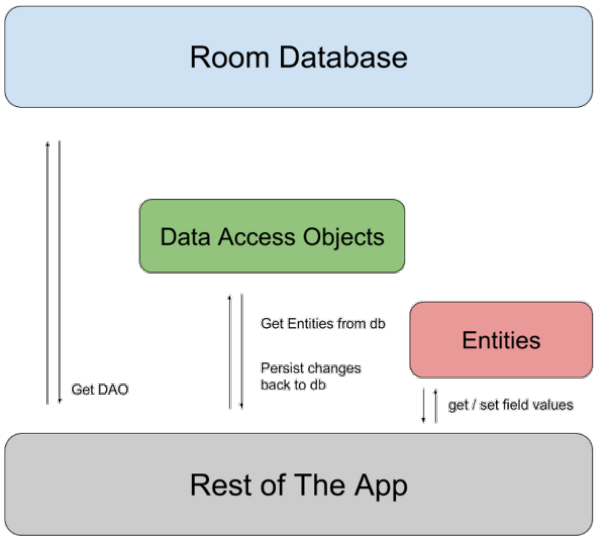


FIGURE 3.1 – Architecture de Room [9].

3.6 Librairies utilisées

WeekView est une librairie 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.

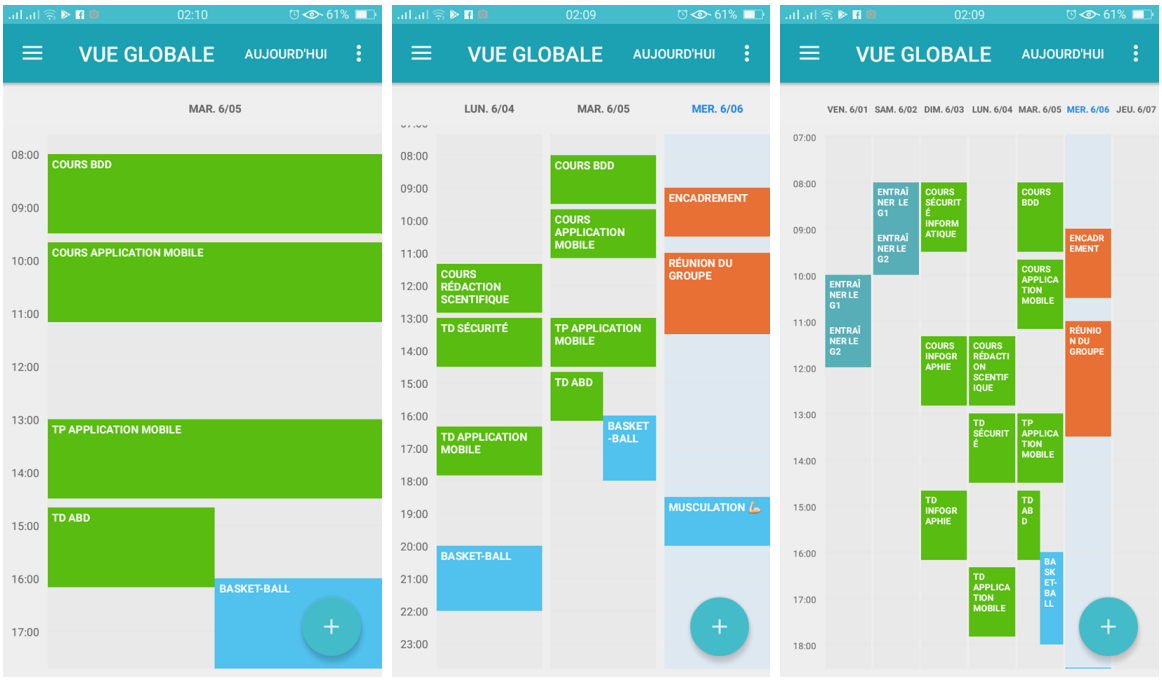
ColorPicker est comme son nom l'indique, une librairie qui permet de choisir une ou plusieurs couleurs.

Il est à noter que les librairies utilisées sont 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. C'est la première interface affichée à l'utilisateur au lancement de l'application. Son principal composant est une vue calendrier qui 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 '+' qui permet d'ajouter des événements, un menu latéral pour naviguer entre les différentes activités de l'application, et enfin un menu qui ouvre une liste déroulante pour changer le type du calendrier.



(a) Vue Aujourd'hui (b) Vue 3 jours (c) Vue une semaine

FIGURE 3.2 – Interface d'accueil

3.7.2 Interface Calendrier

La figure 3.3 représente la liste des calendriers dont un par défaut (Mon calendrier) qui est créé automatiquement lors de l'installation de l'application. Nous avons également d'autres calendriers qui ont été ajoutés par l'utilisateur, ces derniers peuvent être modifiés ou supprimés en appuyant sur le bouton qui lui est associé. En appuyant sur l'un des calendriers, l'utilisateur est redirigé vers la vue globale contenant seulement les événements qui lui sont affectés. Pour ajouter un calendrier, il suffit d'appuyer sur le bouton '+' pour être redirigé vers l'interface d'ajout d'un calendrier.

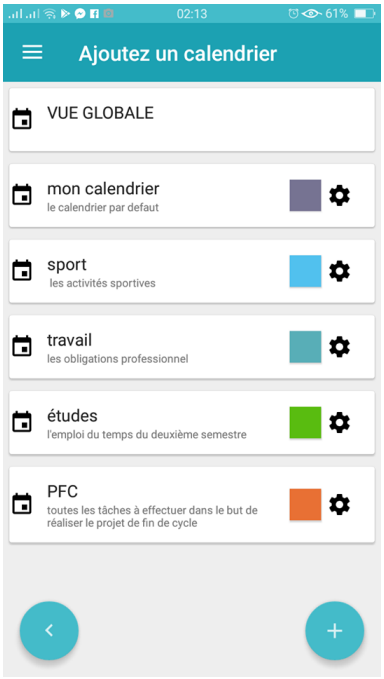


FIGURE 3.3 – Interface calendrier.

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 à décider à quel moment il souhaite être alerté, désactiver complètement l'alerte, sinon.

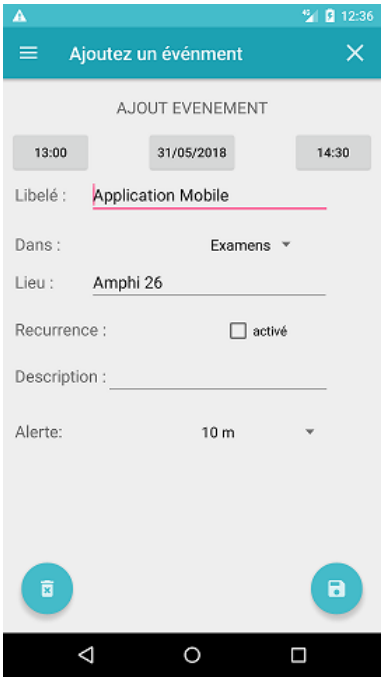


FIGURE 3.4 – Ajout d'un événement.

3.8 Conclusion

Dans ce dernier chapitre, nous avons revu notre environnement de développement et décrit les outils utilisés notamment GitHub qui a été un outil clé pour notre travail collaboratif. Nous avons également vu les différents langages de programmation. Concernant la base de données, Room a été un véritable atout avec sa facilité à gérer et à implémenter la base de données. Un gain de temps que l'on attribue, aussi, aux différentes librairies. Enfin, nous avons vu les différentes interfaces principales qui composent notre application.

Conclusion générale et perspectives

Au cours des chapitres précédents, nous avons défini les aspects principaux de la méthodologie de conception suivie, dans le but de concevoir une application Android permettant une gestion optimale du temps de l'étudiant, et ce grâce à une étude du contexte et de la problématique. Nous avons rédigé un cahier des charges, analyser les différentes fonctionnalités de l'application puis les modéliser grâce à UML. Nous avons, également, implémenter ces dernières en ayant recours à l'environnent de développement Android Studio, au langage Java et XML, GitHub ainsi que Git comme outils de travail collaboratif, et exploiter des librairies Open Source comme «ColorPicker» et «WeekView».

Nous avons réussi à concevoir une application qui donne la possibilité à l'étudiant d'introduire ses activités dans de différents calendriers et de recevoir des notifications, d'avoir une vue détaillée sur un ou plusieurs jours de son emploi du temps tout en ayant la possibilité de modifier à tout moment. L'application obtenue peut être utilisée par tout le monde ayant besoin de gérer son temps et de mieux s'organiser et ne demande pas un grand temps d'adaptation.

Néanmoins, la solution proposée a ses limites comme l'impossibilité en son état actuel de gérer des événements qui s'étalent de plus d'une journée.

En guise de perspective, nous travaillons sur la possibilité d'inclure une note vocale dans l'événement afin d'offrir à l'utilisateur un moyen autre que la saisie pour enregistrer des informations complémentaires. Aussi, relier l'application à un compte Gmail afin d'avoir accès à tout moment à l'emploi du temps sur tous les appareils mobiles utilisant ce même compte (Synchronisation des calendriers).

Ce projet a été pour nous une occasion de mettre en pratique toutes les connaissances acquises au cours de ce cycle et d'avoir un aperçu sur le métier de développeur tout en acquérant de nouvelles compétences et informations, car à l'issue de ce travail, nous avons appris à travailler de manière collaborative et à utiliser Android Studio, Visual Paradigm, Git/GitHub, etc.

Bibliographie

[1] Digitalsobe, *Statistiques d'utilisation mobile ?*, https://fr.digitalsobe.com/statistiques-dutilisation-mobiles-, consulté le 25/04/2018.

[2] Pascal Roques, *UML 2 Modéliser une Application Web*, 4^{e} édition, Eyrolles, Paris, 2008.

[3] Laurent Audibert, *UML 2 de l'apprentissage a la pratique* , https://laurent-audibert.developpez.com/Cours-UML/, 2009.

[4] Taktil Communication, *Application mobile définition et typologie*, https://www.taktilcommunication.com/blog/applications-mobile/definition-typologie-applications-mobiles.html, 2016, consulté le 27/04/2018.

[5] Laurent Audibert, *UML 2 : de l'apprentissage à la pratique*, Ellipses, Paris, 2^{e} édition, 2009.

[6] Niedercorn, *UML : Cas d'utilisation*, Lycée technique «La Briquerie» 57100 Thionville, http://niedercorn.free.fr/iris/iris1/uml/uml10.pdf, Année !!!!!!!, consulté le 05/05/2018.

[7] Rim Chaabane, *Le modèle de données relationnel*, http://www.ai.univ-paris8.fr/~lysop/bd/seance4-ModeleRel.pdf, consulté le 01/05/2018.

[8] Christian Soutou, *UML 2 pour les bases de données.*, Eyrolles, Paris, 2007.

[9] Android Developers, *Save data in a local database using Room*, https://developer.android.com/training/data-storage/room/, consulté le 17/05/2018.

Annexe

Diagramme d'interaction « Modifier événement »

Dans la Figure 5, nous présentons le diagramme d'interaction du cas d'utilisation « Modifier événement ».

Depuis la vue globale, l'utilisateur choisit quel événement veut-il modifié. Par la suite, il est redirigé vers l'interface "Modifier événement" pour saisir les nouvelles informations. En validant ses choix, une vérification de la saisie est effectuée. Si l'utilisateur se trompe, il sera invité à ressaisir les données introduites. Au cas contraire, le système envoie les nouvelles informations à la classe «contrôle événement» qui se chargera de modifier l'événement concerné. Une notification est affichée à l'utilisateur que la modification a été effectuée avec succés. Suite à ça, la vue "Modifier événement" sera détruite. L'utilisateur sera redirigé vers la "Vue Globale" qui sera, par conséquant, actualisée.

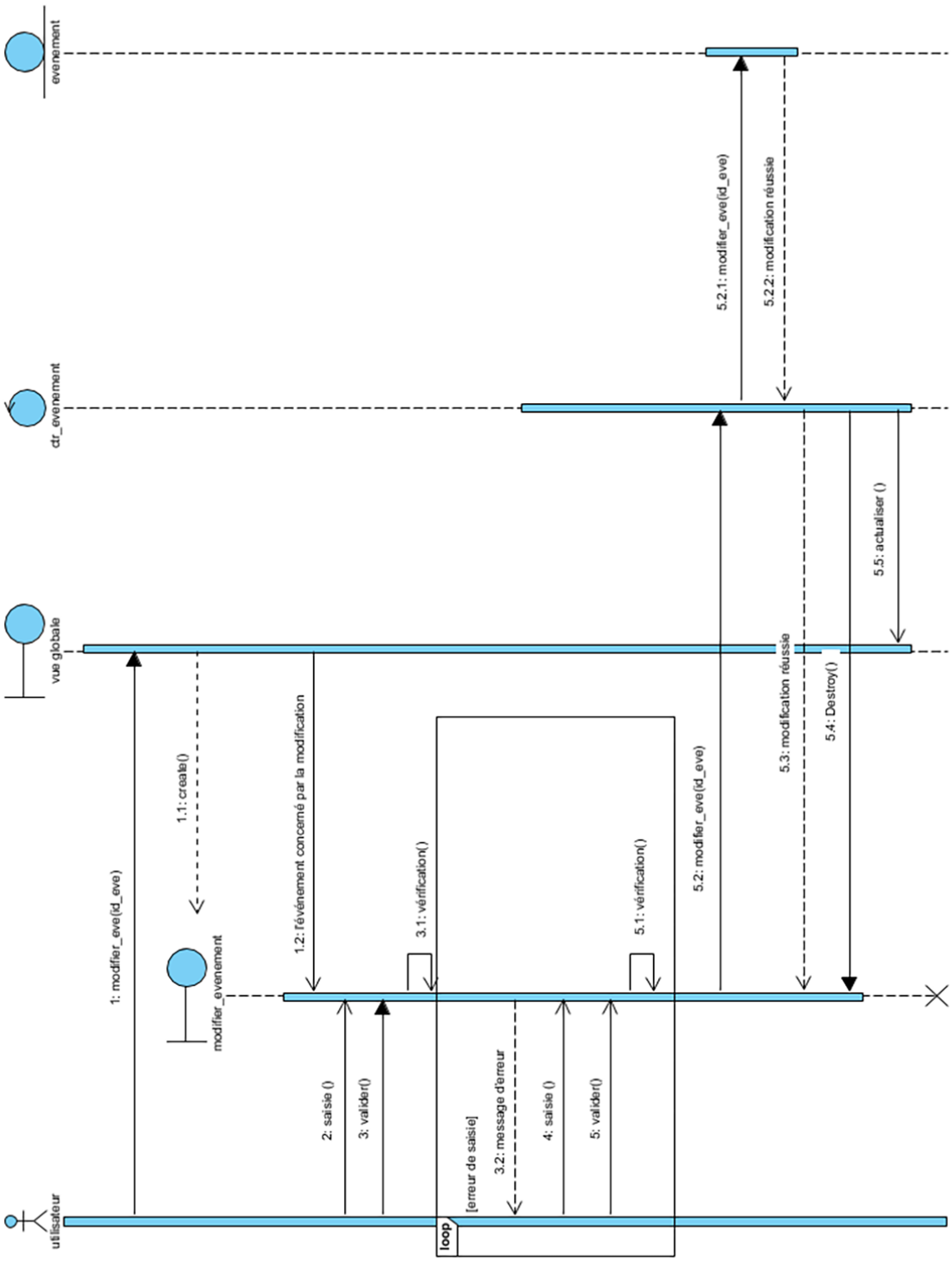
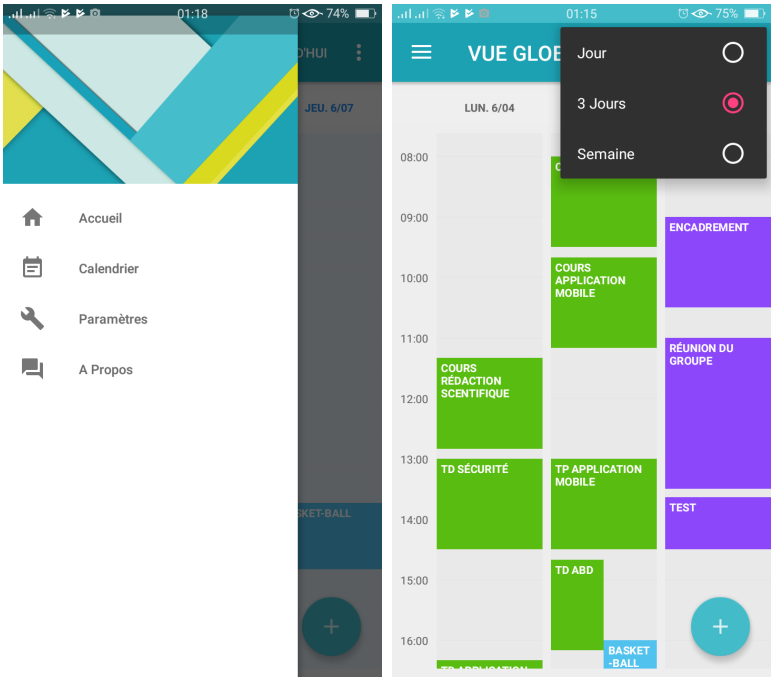


FIGURE 5 – Diagramme d'interaction « Modifier événement ».

Interfaces de l'application

La Figure 6(a) représente le menu latéral qui permet de naviguer entre les différentes activités de l'application.

La Figure 6(b) représente le menu d'option, il permet à l'utilisateur de choisir le nombre de jours à afficher dans la vue globale.



(a) Menu latéral (b) Menu Option

FIGURE 6 – Les menus latéraux

Résumé

Au cours de cette décennie, l'industrie de la téléphonie mobile a connu une progression fulgurante suite à l'apparition des smartphones. Voulant aider l'étudiant à mieux gérer son temps et à être plus productif, nous avons réalisé ce projet. Ceci en implémentant les solutions modélisées lors de la conception suite à l'analyse des besoins en utilisant UML comme langage de conception, Android Studio, Java, XML, SQLite, Git, GitHub, et Room.

Mots clés : *Gestion du temps, Application mobile, Android, Room, SQLite.*

Abstract

At the end of this decade, the mobile phone industry has experienced a huge rise, following the appearance of smartphones. Wanting to help the student to better organize their schedule and to be more productive. We realized this project. Implementing the modeled solution during design follows needs analysis. Using UML as the modeling language, Android Studio, Java, XML, SQLite, Git, GitHub, and Room.

Keywords : *Time scheduling, Mobile application, Android, Room, SQLite.*